VLSI DESIGN 2009

VLSI DESIGN 2009

IMPROVING PRODUCTIVITY THROUGH HIGHER ABSTRACTION

THE 22nd INTERNATIONAL CONFERENCE ON VLSI DESIGN

Held jointly with

THE 7th INTERNATIONAL CONFERENCE ON EMBEDDED SYSTEMS

New Delhi, India
5–9 January 2009

Sponsored by

VLSI SOCIETY OF INDIA (VSI)

In Technical Co-Sponsorship with

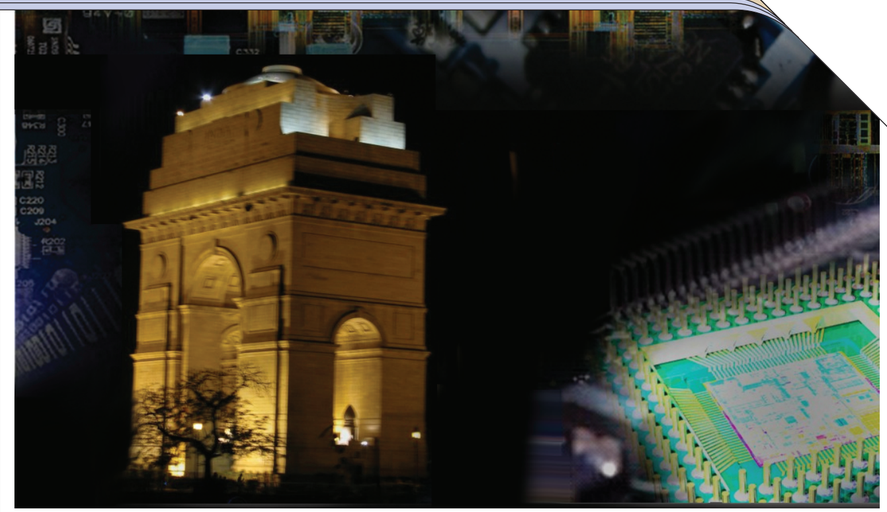
IEEE CIRCUITS AND SYSTEMS SOCIETY

IEEE ELECTRON DEVICES SOCIETY

IEEE SOLID STATE CIRCUITS SOCIETY

Sister Conference

IEEE/ACM DESIGN AUTOMATION CONFERENCE



Published by the IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

IEEE Computer Society Order Number P3506
BMS Part Number: CFP09041-PRT
ISSN 1063-9667
ISBN 978-0-7695-3506-7

ISBN 978-0-7695-3506-7
90000
9 780769 535067

VLSI DESIGN 2009 • *IMPROVING PRODUCTIVITY THROUGH HIGHER ABSTRACTION*
PROCEEDINGS OF THE 22nd INTERNATIONAL CONFERENCE ON VLSI DESIGN 5-9 January 2009
HELD JOINTLY WITH THE 7th INTERNATIONAL CONFERENCE ON EMBEDDED SYSTEMS

IEEE COMPUTER SOCIETY

IEEE

Table of Contents

VLSI Design 2009

Invited Talks/Special Sessions
Chair: Rajiv Kapur

Made For India Forum

VLSI Design Conference 2009 – Panels
Chair: Rajiv Kapur

Tutorials

Session 1A: Low Power Design for Wireless Communication

Session 1B: SoC Verification

Session 1C: Fault Diagnosis

Session 2A: Analog and Mixed Signal I

Session 2B: Floorplanning and Analog Layout

Session 2C: Network on Chip

Session 3A: Low Power Device Technology

Session 3B: System Synthesis

Session 3C: Test Generation

Session 4A: Advanced Device Modeling

Session 4B: Application-Specific Architectures and Reconfigurable Computing

Invited Talk

Session 4C: Embedded Systems I

Session 5A: SRAM and Random Number Generation

Session 5B: Secure VLSI Design

Session 5C: Embedded Systems II

Session 6A: Analog and Mixed Signal II

Session 6B: Routing, Power Optimization

Session 6C: Low Power Design

Session 7A: Analog and Mixed Signal III

Session 7B: Reliability and Design Space Exploration

Session 7C: BIST, Error Modeling

Session 8A: Advanced Nanodevice Modeling

Session 8B: Timing Analysis and Optimization

Invited Talk

Session 8C: Processor Design and Scheduling

Session 9A: VLSI Education

Session 9B: Invited Paper-Phase Locked Loops

Session 9C: Invited Paper-Design for Variations

Proceedings

VLSI Design 2009

Proceedings

22nd International Conference on VLSI Design

Held jointly with
8th International Conference on Embedded Systems

New Delhi, India *5-9 January 2009*

Technical Co-Sponsorship
IEEE Circuits and Systems Society
IEEE Solid State Circuits Society
IEEE Electron Devices Society

Sponsored by
VLSI Society of India

Sister Conference
IEEE/ACM Design Automation Conference

CPS Conference Publishing Services

Los Alamitos, California

Washington • Tokyo

IEEE computer society

IEEE Computer Society Order Number P3506
BMS Part Number CFP09041-PRT
ISBN 978-0-7695-3506-7
ISSN Number 1063-9667

Additional copies may be ordered from:

IEEE Computer Society
Customer Service Center
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314
Tel: + 1 800 272 6657
Fax: + 1 714 821 4641
http://computer.org/cspress
csbooks@computer.org

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
Tel: + 1 732 981 0060
Fax: + 1 732 981 9667
http://shop.ieee.org/store/
customer-service@ieee.org

IEEE Computer Society
Asia/Pacific Office
Watanabe Bldg., 1-4-2
Minami-Aoyama
Minato-ku, Tokyo 107-0062
JAPAN
Tel: + 81 3 3408 3118
Fax: + 81 3 3408 3553
tokyo.ofc@computer.org

Individual paper REPRINTS may be ordered at: <reprints@computer.org>

Editorial production by Lisa O'Conner
Cover art production by Alex Torres
Printed in the United States of America by The Printing House

IEEE Computer Society
***Conference Publishing Services* (CPS)**
http://www.computer.org/cps

Message from the General Chairs


Anshul Kumar


Anurag Seth

Welcome to the joint 22nd International Conference on VLSI Design and 8th International Conference on Embedded Systems being held from 5th to 9th January, 2009 in New Delhi.

With a history of more than two decades, this conference has now become a prestigious annual event with participation from industry leaders, technical experts, practicing engineers, academicians and students from India and abroad. The conference theme this year is Improving Productivity Through Higher Abstraction, which reflects a growing concern about design productivity as the VLSI chips and systems are becoming increasingly complex while the time available for design is shrinking.

Like the past years, the conference brings three days of exciting technical sessions with papers selected through a world-wide peer review process and two days of informative tutorials by experts on topics of current interest. While the technical papers will present new results and discuss the intricacies of VLSI Design, Electronic Design Automation and Embedded Systems, there will be keynotes and invited talks by the leaders in the industry and academia, who would present a broader picture of the state of the art and do a bit of crystal gazing into the future. Keynote speakers in this conference include Justin Rattner, Intel Senior Fellow, Chief Technology Officer, Abhi Talwalkar, President and CEO, LSI Corp., Neil Henderson, GM, Mentor Graphics, Thomas Williams, Synopsys Fellow, Dr. Ajoy Bose, Founder, Atrenta and Jacob Abraham, Professor, University of Texas, Austin.

The conference has been constantly striving for improving its quality in order to attract the best of the people in the field. This year there has been a very good response to our call for papers. There were as many as 320 paper submissions, out of which the Program Chairs, Preeti Ranjan Panda and Rajendran Panda, along with their team of program committee members and reviewers, have selected the best 79 through a rigorous review process. This is a testimony to the growing stature of the conference and the selectivity it can afford now. The Tutorial Chairs, Sanjiv Narayan and Atul Jain, with their team of reviewers, have organized 7 full day and 2 half day tutorials on a variety of topics including Power-conscious Tests, Security & Dependability of Embedded Systems and Power Management for Mobile Multimedia. Again, the response for tutorial submissions has been overwhelming with a record 40+ submissions and the review team had a tough time selecting the 9.

The Industry Forum is now a regular feature of this conference. The Special Sessions Chair Rajiv Kapur has put together a session on Made for India, which would try to capture thoughts about the growing economy of India as an agent in shaping the electronic products. Among the special sessions, besides the mentioned keynotes, we have various high quality technical talks ranging from platform based design to computational lithography to Analog Design to Low Power issues and standards. Moreover, we have

four panel discussions this year in the areas of IP, EDA, Embedded software, and Automotive/Made for India.

We continue to encourage and facilitate participation of students. In the 2008 conference, a new feature Students Track was introduced with a view to initiate them in the field. The Student Track chairs Gurudutt Bansal and M. Jagadesh Kumar have organized a track for students that will run parallel to technical sessions on the 3rd day of the conference. The regular participants from the academia, both students and faculty, are supported through the fellowship program, which is being coordinated by the Fellowships Chair Vineet Sahula. The promotional activities of the conference, which aim at bringing out innovative ideas from the budding professionals, include a Design Contest and the new EDA Software Contest introduced this year. The Design Contest Chairs Aloknath De and Subind Kumar and the EDA Contest Chairs Manu Lauria and Shabbir Batterywala have organized these contests.

A conference of this magnitude cannot materialize without the financial support from the industry as well as organizational efforts from the individuals. The Sponsorship and Exhibits Chairs Sanjeev Aggarwal and S. Uma Mahesh have taken care of mobilizing the required financial resources. The sponsors this year include Cadence, ARM, Magma, Atrenta, Broadcom, and Conexant. You can expect to see state of the art exhibits from these companies as well as several others. The Organizing Chair Rajeev Sehgal and Organizing Coordinator Harish Chauhan have put in tremendous efforts to bring the conference to this shape. As expected, without the proactive drive from the Finance Chair Prem Nivasa (especially in the current economy) with a good tab on inflows and outflows and compliance issues etc, things would have been not as smooth.

We would also like to acknowledge the proactive efforts from our publicity chairs – Ricky Bedi, Sapan Garg and Yatin Trivedi to make the conference a grand affair.

The role of the Publication Chair Nagi Naganathan and Lisa O'Conner, Production Editor, Conference Publishing Services of IEEE has been very crucial in bringing out the conference proceedings in a timely manner.

The conference has been enjoying association with several reputed professional societies, that adds to its prestige. We acknowledge the contributions of CP Ravikumar (VLSI Society of India), N. Ranganathan (IEEE Circuits and Systems Society), Anantha Chandrakasan (IEEE Solid-State Circuits Society), Nikil Dutt (ACM/SIGDA) and Poornima Shenoy (Indian Semiconductor Association) in making this association continue.

Finally we would like to thank the Steering Committee Chair Vishwani Agrawal and other members of the Steering Committee for giving us this opportunity to organize the conference. Hope you will enjoy the conference as well as the host city of Delhi.

Anshul Kumar and Anurag Seth

Message from the Program Chairs

Preeti Ranjan Panda

Rajendran Panda

It is our pleasure to welcome you all to 2009 edition of the VLSI Design Conference! A strong technical program has been the hallmark of VLSI Design Conferences in the past, and we have endeavored to ensure that you will witness an outstanding technical program this year also.

In keeping with the international character of the conference, this year we received about 320 paper submissions from 21 countries. While India and USA accounted for a large fraction of the submissions, we also received a significant response from Iran, China, Japan, Germany, and UK. Getting the papers reviewed in the two months we had set for ourselves was a challenging task and would not have been possible without the hard and diligent work put in by the 90 member Technical Program Committee consisting of leading researchers from across the world, from both academia and industry. The papers were divided into 9 tracks based on subject areas, and were reviewed by a total of more than 300 reviewers. We are very much indebted to the team of reviewers for doing an outstanding job in making a thorough and fair evaluation of the submissions, that resulted in an average of more than 4 reviews per paper. Email discussions were initiated by the track chairs even before the actual program committee meeting, and approximate decisions were identified. In keeping with the conference tradition, two program committee meetings were organised - one in IIT Delhi and the other in Rutgers University, USA. A total of 57 Regular papers and 22 Short papers were selected by the program committee for inclusion in the technical program. These excellent papers, on a wide-ranging set of topics related to VLSI Design and Embedded Systems, have been organized into three parallel tracks for presentation. The selection process was highly competitive and many good papers could not make it to the final list. We sincerely hope that the feedback from our expert reviewers was helpful to every author. This being also an embedded systems conference, we have one running track on 6th January titled “Embedded Systems Day”.

We would like to thank all authors for considering VLSI Conference as a venue for publishing their work. We would like to place on record our gratefulness to the Technical Program Committee members and the volunteer reviewers who worked very hard to make the technical program happen. The program committee list and reviewer list appears elsewhere in these proceedings, but we would like to take the opportunity to convey our special thanks to the track chairs who managed the review procedure within the nine tracks: Srivaths Ravi (Test), Prabhat Mishra (Synthesis and Verification), Nitin Chandrachoodan and Sudeep Pasricha (Application Specific Architectures), Puneet Gupta and Susmita Sur-Kolay (Physical Design), Praveen Elakkumanan and Nagi Naganathan (Low Power Electronics), Shouri Chatterjee (Analog), Kolin Paul and Tulika Mitra (Embedded Systems), Rajiv Joshi (TCAD), and Vijay Degalahal (Architecture). We also like to convey our deep appreciation to Vishwani Agrawal, Srimat Chakradhar, N. Ranganathan, and Mike Bushnell for their valuable help, feedback, and support throughout the planning and execution of the review process.

Special thanks are due to Nagi Naganathan, the Publications Chair, for undertaking the crucial and difficult task of co-ordinating with everyone else for compiling the contents of the conference proceedings and overall management of the iterative process, and to Lisa O'Conner, Production Editor at the Conference Publishing Services of IEEE for her expert handling of the proceedings production and for patiently and instantly accommodating the numerous update requests in spite of ill health. The polished final product in your hands is the result of their untiring efforts.

We sincerely believe we have assembled for you an outstanding technical program, and we hope you enjoy the experience. The charming capital city of Delhi has ancient history rubbing shoulders with high technology; it will dazzle you with its breathtaking sights, and has much to offer to both the first time visitor as well as the experienced traveller. We invite you to lose yourself amidst the tantalizing environs in and around Delhi.

Preeti Ranjan Panda, *IIT Delhi*
Rajendran Panda, *Freescale Semiconductor, Austin*
VLSI09 Program Chairs
New Delhi, January 2009

Message from the Organizing Chair

Welcome to the 22nd VLSI Design and 8th Embedded Systems conference to be held from 5th to 9th January 2009.

In the past years India has seen a lot of growth in the VLSI and Embedded Systems industry. The leading educational institutes in India have also been actively introducing curriculum with focus on this industry. Not only that, there has been increasing collaboration between the two on focused research in solving some of the extremely complex problems faced by the industry. The VLSI and Embedded systems conference is more relevant than other time in the past.

In these extremely difficult economic times to stage a conference of this size was a challenge. Anurag Seth and Dr. Anshul Kumar, the General chairs have led from the front in achieving the high goals set for the conference and I would like to acknowledge their efforts for the same.

The conference comes to New Delhi and National Capital Region (NCR) after a gap of 6 years. Significant changes have occurred during this period. There has been the introduction of Metro, building of more flyovers and still a lot of construction is going all around in preparation for the 2010 Commonwealth games to be held here. Similarly there has been tremendous growth in VLSI Design and EDA companies in last six years, NCR is now home to major VLSI Design centers such as ST and Freescale. It is also a home for many EDA companies, such as Cadence, Mentor, Sequence, Magma, Calypto, Co-Ware, Atrenta, Interra Systems, and so on.

I am sure you will enjoy the conference, tutorials, panel discussions, keynote speeches, industry forum and the special "Made for India" track, that the organizing committee has put together this year. Capital of India, New Delhi houses finest museums, galleries, shopping, dining and entertainment inviting you to take a plunge into the history and modernity of the city.

Once again, I welcome you to Delhi and the conference and have a memorable 5 days.

Rajeev Sehgal
Organizing Chair

Conference Steering Committee

Vishwani D. Agrawal, **Chair**

Jaswinder Ahuja
M. Balakrishnan
Srimat T. Chakradhar
Dasaradha Gude
Apurva Kalia
Bobby Mitra
A. Prabhakar
N. Ranganathan
C.P. Ravikumar

VLSI Design 2009 Committee Members

Steering Committee Chair

Vishwani Agrawal
Auburn University

General Chair

Anshul Kumar
IIT Delhi

General Chair

Anurag Seth
Cadence

Program Chair

Preeti Ranjan Panda
IIT, Delhi

Program Chair

Rajendran Panda
Freescale

Organizing Chair

Rajeev Sehgal
Mentor Graphics

Organizing Coordinator

Harish Chauhan
Cadence

Publication Chair

Nagi Naganathan
LSI Corp.

Publicity Chair

Ricky Bedi
Magma

Publicity Chair

Sapan Garg
Atrenta

Publicity Chair

Yatin Trivedi
Magma

Tutorial Chair

Sanjiv Narayan
Calypto

Tutorial Chair

Atul Jain
TI

Special Session Chair

Rajiv Kapur
Broadcom

Sponsorship / Exhibit Chair

Sanjeev Aggarwal
Cadence

Sponsorship / Exhibit Chair

S. Uma Mahesh
Indrion Technologies

Finance Chair

Prem Nivasa
Mentor Graphics

Design Contest Chair

Aloknath De
ST Microelectronics

Design Contest Chair

Subind Kumar
Freescale

Fellowship Chair

Vineet Sahula
MNIT, Jaipur

EDA Contest Chair

Manu Lauria
Cadence

EDA Contest Chair

Shabbir Batterywala
Synopsys

Student Track Chair

Gurudutt Bansal
Cadence

Student Track Chair

M Jagadesh Kumar
IIT, Delhi

IEEE Liaison

Nagarajan Ranganathan
University of South Florida

SSCS Liaison

Anantha Chandrakasan
MIT

ISA Liaison

Poornima Shenoy

VSI Liaison

CP Ravi Kumar
TI

ACM/SIGDA Liaison

Nikil Dutt
University of California, Irvine

VLSI Design 2009
Technical Program Committee

Program Co-Chairs
Preeti Ranjan Panda, *IIT Delhi, India*
Rajendran Panda, *Freescale Semiconductor, USA*

Tracks (Names of Track Chairs or Co-Chairs are in BOLD)

AMS: Analog, RF, Mixed Signals
Shouri Chatterjee, *IIT Delhi, India*
Debapriya Sahu, *Texas Instruments, India*
G.S. Visweswaran, *IIT Delhi, India*
Nagendra Krishnapura, *IIT Madras, India*
Pavan K. Hanumolu, *Oregon State University, USA*
Prakash Easwaran, *Cosmic Circuits, India*
Sambuddha Bhattacharya, *Synopsys, India*
Shanthi Pavan, *IIT Madras, India*
Srinivasan C, *Cosmic Circuits, India*
Vivek G. Pawar, *Sankalp Semiconductor, India*

ASA: Application Specific Architecutures, Security
Nitin Chandrachoodan, *IIT Madras, India*
Sudeep Pasricha, *Colorado State University, USA*
Ashish Mathur, *Freescale Semiconductor, India*
Mona Mathur, *ST Microelectronics, India*
Sivakumar Sri, *Wipro, India*
Sri Parameswaran, *UNSW, Australia*
Sri Chandra, *Freescale Semiconductor, India*
Sriram R. Vangal, *Intel, USA,*

PHY: Physical Design, DFM, Power and Signal Integrity, Interconnect and Timing Analysis/Optimization, Reliability
Puneet Gupta, *UCLA, USA*
Susmita Sur-Kolay, *ISI Kolkata, USA*
Chul-Hong Park, *Samsung, S. Korea*
Elaheh Bozorgzadeh, *University of California, Irvine, USA*
Florin Balasa, *Southern Utah University, USA*
Min Zhao, *Magma Design Automation, USA*
Murat Becer, *CLK-DA, USA*
Puneet Sharma, *Freescale Semiconductor, USA*
Sao-Jie Chen, *National Taiwan University, Taiwan*
Savithri Sundareswaran, *Freescale Semiconductor, USA*
Shabbir Batterywala, *Synopsys, India*
Vishal Khandelwal, *Synopsys, USA*
Vladimir Zolotov, *IBM, USA*

EMB: Embedded Systems, Fault Tolerance, Sensor Networks, Ubiquitous Computing, Asynch. Design, Low Power Systems
Kolin Paul, *IIT Delhi, India*
Tulika Mitra, *National University of Singapore, Singapore*
Anup Gangwar, *AMD, India*
Basant Dwivedi, *Calypto Design Systems, India*
Javier Resano, *UCM Madrid, Spain*
David Atienza, *UCM Madrid, Spain*
Madhu Mutyam, *IIT Madras, India*
Mahesh Mehendale, *Texas Instruments, India*
Nikil Dutt, *UC Irvine, USA*
Niraj Jha, *Princeton University, USA*
Paolo Ienne, *EPFL, Switzerland*
Pradip Jha, *Xilinx, USA*
Samarjit Chakrabarty, *National University of Singapore, Singapore*
Sarma Vrudhula, *Arizona State University, USA*

LPE: Low Power Electronics
Nagi Naganathan, *LSI Corp, USA*
Praveen Elakkumanan, *IBM, USA*
Amit Patra, *IIT Kharagpur, India*
Bharadwaj Amrutur, *IISc, India*
Ram Krishnamurty, *Intel, USA*
Tezaswi Raja, *LSI Corp, USA*
Xin Li, *CMU, USA*
Aditya Bansal, *IBM, USA*
Saibal Mukhopadhyay, *Georgia Tech., USA*

TECH: Technology, Device Modeling and Simulation, MEMs, Nanoelectronics, and Biological Systems
Rajiv Joshi, *IBM, USA*
Bipul Paul, *Toshiba, USA*
Durgamadhab Misra, *NJIT, USA*
Josef Watts, *IBM, USA*
M. Jagadesh Kumar, *IIT Delhi, India*
Madabusi Govindarajan, *IBM, India*
Sukumar Jairam, *Texas Instruments, India*

ARCH: Processor Architecture, Multi-core Systems
Vijay Degalahal, *Intel, India*
Ashok Jagannathan, *Intel, India*
Maurizio Palesi, *UNICT, Italy*
Petru Eles, *Linkoping Universitet, Sweden*
Nagarajan Ranganathan, *University of South Florida, USA*
S.K. Nandy, *IISc, India*
Sourav Roy, *Freescale Semiconductor, India*

SYN: Design Specification, Modeling, and Synthesis, Hardware/Software Co-design, Simulation, Emulation and Formal Verification
Prabhat Mishra, *University of Florida, USA*
Indira Iyer, *Synfora, India*
Jayanta Bhadra, *Freescale Semiconductor, USA*
Logie Ramachandran, *Synopsys, USA*
Malay Haldar, *Calypto Design Systems, India*
Pallab Dasgupta, *IIT Kharagpur, India*
Saraju Mohanty, *University of North Texas, USA*
Shankar Hemmady, *Synopsys, USA*
Sandeep Shukla, *Virginia Tech, USA*

TEST: Testing, DFT
Srivaths Ravi, *Texas Instruments, India*
Adit Singh, *Auburn University, USA*
Bernard Courtois, *CMP, France*
Bhargav Bhattacharya, *ISI Kolkata, India*
Kartik Mohanram, *Rice University, USA*
Kewal Saluja, *University of Wisconsin, USA*
Nagesh Tamarapalli, *AMD, India*
Pradip Thaker, *Analog Devices, India*
C. P. Ravikumar, *Texas Instruments, India*
Rubin Parekhji, *Texas Instruments, India*
Srimat Chakradhar, *NEC Labs, USA*
Sudhakar Reddy, *University of Iowa, USA*
Vishwani Agrawal, *Auburn University, USA*

Reviewers

External Reviewers

Aashish Pant	*UCLA*
Abhijit Das	*TI*
Adam Donlin	*Xilinx*
Ajit Pal	*Indian Institute of Technology-Kharagpur*
Ajit Gupte	*TI*
Alexander Fell	*Indian Institute of Science*
Alok Pugalia	*Sankalp Semiconductor P. Ltd.*
Aman Kokrady	*TI*
Amin Khajeh	*University of California, Irvine*
Amit Badole	*Freescale Semiconductor*
Amit D	*TI*
Amol Bhinge	*Freescale Semiconductor*
Amrit Singh	*Freescale Semiconductor*
Anshoo Tandon	*Freescale Semiconductor*
Anshuman Chandra	*Synopsys*
Ansuman Banerjee	*Interra Systems*
Anthony Chun	*Intel*
Anupam Singal	*Freescale Semiconductor*
Apu Sivadas	*TI*
Aravindh Anantaraman	*Intel*
Ari Valero-Lopez	*LSI*
Arup Chakraborty	*University of California, Irvine*
Aseem Gupta	*University of California, Irvine*
Ashok Balivada	*Analog*
Bernd Becker	*University of Freiburg, Germany*
Bhaskar Pal	*Synopsys*
Bhasker Karmakar	*TI*
Bijoy Jose	*Virginia Tech*
Bo Hu	
Bo Wan	
Chandra Tirumurti	*Intel*
Chandramouli Gopalakrishnan	*Synopsys*
Chao Huang	*Virginia Tech*
Charles Wen	*National Chiao Tung University, Taiwan*
Chittaranjan Mandal	*Indian Institute of Technology-Kharagpur*
Chouki Aktouf	*DeFacTo Technologies*
Chrysostomos Nicopoulos	*EPFL*
Chungchun Wan	*Stanford University*
Chunhua Yao	*University of Wisconsin*
Colin Tan	*National University of Singapore*
Craig Gleason	*Synfora*
Daniel Chaver	*Universidad Complutense de Madrid*
Daniel Mozos	*Universidad Complutense de Madrid*

Debasri Saha	*Indian Statistical Institute*
Deepak C.	*Indian Institute of Science*
Deepak Mathaikutty	*Intel*
Deji Akinwanda	*Stanford University*
Devadutt K	*Synfora*
Devanathan Varadarajan	*TI*
Dhruva Ghai	*University of North Texas*
Dong Ha	*Virginia Tech*
Dwarka Prasad	*Freescale Semiconductor*
Ekaterina Trofimova	*Freescale Semiconductor*
Elias Kougianos	*University of North Texas*
Erika Cota	*Universidade Federal do Rio Grande do Sul*
Fan Wang	*Juniper Network*
Gabor Madl	*University of California, Irvine*
Gaurav Singh	*Virginia Tech*
Gireesh Rajendran	*TI*
Grady Giles	*AMD*
Haihua Yan	*Synopsys*
Hangsheng Wang	*Freescale Semiconductor*
Hans-Joachim Wunderlich	*University of Stuttgart, Germany*
Harish P	*Indian Institute of Science*
Hillary Grimes	*Auburn University*
Himanshu Thapliyal	*University of South Florida*
Himyanshu Anand	*Freescale Semiconductor*
Hiren Patel	*University of California, Berkeley*
Huynh Phung Huynh	*National University of Singapore*
Ilia Polian	
Indradeep Ghosh	*Fujitsu*
Irith Pomeranz	*Purdue University*
Iwao Yamazaki	*Hitachi*
Jais Abraham	*AMD*
James Tschanz	*Intel*
Janakiraman V	
Jawar Singh	*Bristol University*
Jiang Hu	*Texas A&M University*
Jie Qin	*Auburn University*
Jim Holt	*Freescale Semiconductor*
Jim Plusquellic	*University of New Mexico*
Jimson Mathew	*Bristol University*
Jins Davis Alexander	*Auburn University*
Jong Eun Lee	*Arizona State University*
Jooheung Lee	*University of Central Florida*
José Ayala	*Universidad Complutense de Madrid*
Jose Ignacio Gomez	*Universidad Complutense de Madrid*
Jose Luis Risco	*Universidad Complutense de Madrid*
Juan Antonio Clemente	*Universidad Complutense de Madrid*
Kalyana Chakravarty	*Freescale Semiconductor*
Kanishka Lahiri	*Intel*
Kausar Banoo	*LSI*
Kausik Datta	*Interra Systems*

Kedarnath Balakrishnan	*AMD*
Keshavan Varadarajan	*Indian Institute of Science*
Koushik M	*Indian Institute of Science*
Koustav Bhattacharya	*University of South Florida*
Krishnaiah Gummidipudi	*Indian Institute of Technology-Delhi*
Krishnaswamy T	*TI*
L. Begin	*Université Libre de Bruxelles*
Lerong Chen	*University of California, Los Angeles*
Lin Li	*Intel*
Lin Zhong	*Rice University*
Loganathan Lingappan	*Intel*
Lokesh Gupta	*Magma*
Luis Bathen	*University of California, Irvine*
M. Balakrishnan	*Indian Institute of Technology-Delhi*
Mahalingam Venkataraman	*University of South Florida*
Mainak Banga	*Virginia Tech*
Manish Pillai	*TI*
Manish Ratnani	*University of North Texas*
Manish Pillai	*TI*
Manvi Agarwal	*NXP*
Marcos Sanchez-Elez	*Universidad Complutense de Madrid*
Michael Bushnell	*Rutgers University*
Michael Hsiao	*Virginia Tech*
Michel Renovell	*LIRMM, France*
Mihir Choudhury	*Rice University*
Mike Burns	*Freescale Semiconductor*
Mingsong Chen	*University of Florida*
Minyoung Kim	*University of California, Irvine*
Mohammed Ashfaq Shukoor	*Auburn University*
Mrinal Bose	*Freescale Semiconductor*
Mukesh Mishra	*Intel*
Mukund Sivaraman	*Synfora*
Narasimhaswamy Bharath	*AMD*
Naveen Raina	*Freescale Semiconductor*
Neeraj Goel	*Indian Institute of Technology-Delhi*
Nisar Ahmed	*TI*
Nishad P	*University of Minnesota*
Nithiyanandan Bashyam	*Intel*
Nitin Yogi	*Auburn University*
Nur Touba	*University of Texas-Austin*
Pablo García	*Universidad Complutense de Madrid*
Palkesh J	*TI*
Patrick Schaumont	*Virginia Tech*
Paul Schumacher	*Xilinx*
PJ Joseph	*Freescale Semiconductor*
Prajit Nandi	*Sankalp Semiconductor P. Ltd.*
Prakash Venkitaraman	*AMD*
Prasanta Basu	*Calcutta University*
Pratap Das	*Indian Institute of Science*
Praveen Sanjeev	*Analog*

Preetam T	*TI*
R Datta	*TI*
Rahul Jain	*Calypto Design Systems*
Rajamani Sethuram	
Rajaraman Ramanarayanan	*Intel*
Rajat Bhatia	*Freescale Semiconductor*
Rajesh S	*Cosmic Circuits*
Rakesh Kumar	*TI*
Rakesh Gnana David	
Ramkumar Jayaseelan	*National University of Singapore*
Rani Ghaida	*University of California, Los Angeles*
Ranjan Bose	*Indian Institute of Technology-Delhi*
Ravi Venkatesan	*Intel*
Rohit Kapur	*Synopsys*
Rubin Parekhji	*TI*
Saibal Mukhopadhyay	*Georgia Tech*
Sanatan Chattopadhyay	
Sandeep Oswal	*TI*
Sandip Kundu	*University of Massachusetts*
Sandip Ray	*University of Texas-Austin*
Sanjay Kumar	*Broadcom*
Sanjukta Bhanja	*University of South Florida*
Sankaran Aniruddhan	
Santosh Salunkhe	*Analog*
Sanu Mathew	*Intel*
Saravana Ganeshan	*TI*
Satish Anand	
Satish Yada	*Intel*
Satishkumar Udayanarayanan	*Wipro*
Saurabh Tiwari	*Intel*
Seiji Kajihara	
Shail Aditya	*Synfora*
Shailendra Jain	*Intel*
Shakti R	*TI*
Sharad Seth	*University of Nebraska-Lincoln*
Shashi Kumar	*Jonkoping University, Sweden*
Shiva Kasiviswanathan	*Penn State*
Shreya Dasgupta	*Analog*
Shyam Sundar	*Analog*
Silpa BVN	*Indian Institute of Technology-Delhi*
Sitaraman Iyer	
Sivaramalinga, Reddy	*Analog*
Soonte Kim	*Information and Communication University, S. Korea*
Soujanna Sarkar	*TI*
Soumyaroop Roy	*University of South Florida*
Sravan Kumar	*AMD*
Srinivas Patil	*Intel*
Srinivas V	*TI*
Srinivas Katkoori	*University of South Florida*
Srinivasulu Alampally	*TI*

Sriparna Saha	*Indian Statistical Institute*
Subhasis Banerjee	*Intel*
Subir Roy	*TI*
Subodh Sharma	*University of Utah, Salt Lake City*
Subramanian Rajagopalan	
Subrat Panda	*Indian Institute of Technology-Kharagpur*
Sudheer Prasad	*TI*
Sudip Shekhar	
Sujoy Chakravarty	*TI*
Suman Mandal	
Sumant Kale	*TI*
Sumit Ahuja	*Virginia Tech*
Surendra Guntur	*NXP*
Suresh PR	*TI*
Sutirtha Sanyal	*Barcelona Supercomputing Center*
Swarup Bhunia	*Case Western*
T Pramod	*Magma*
Tania Mishra	*University of Florida*
Tej Rai	*Freescale Semiconductor*
Theocharis Theocharides	*University of Cyprus*
Tomokazu Yoneda	*NAIST, Japan*
Tuck-Boon Chan	*University of California, Los Angeles*
Udaykumar H	*TI*
Umit Ogras	*Intel*
Upavan Gupta	*University of South Florida*
V. Nagarajan	
Vaideeswaran Sethuraman	*AMD*
Veezhinathan Kamakoti	*Indian Institute of Technology-Chennai*
Venkatesh C	*Indian Institute of Science*
Venugopal Puvvada	
Vijay Sundaresan	*University of Cincinatti*
Vikram K.N.	*George Washington University*
Vincenzo Catania	*University of Catania, Italy*
Virendra Singh	*Indian Institute of Science*
Wei Jiang	*Auburn University*
Wei Qin	*Boston University*
Weixun Wang	*University of Florida*
Wen Xiaoqing	*Kyushu Institute of Technology, Japan*
Xiaofang Wang	*Villanova University*
Xiaoke Qin	*University of Florida*
Xijiang Lin	*Mentor Graphics*
Yang Yi	*Freescale Semiconductor*
Yi-Shing Chang	*Intel*
Yong Wang	
Yoshinobu Higami	*Ehime University, Japan*
Yu Huang	*Mentor Graphics*
Yuhong Fu	*Freescale Semiconductor*
Zhan Guo	*Lund University, Sweeden*
Zhao Li	

Tutorial Committee

Tutorial Co-Chairs

Atul Jain *Texas Instruments*
Sanjiv Narayan *Calypto Design Systems*

Reviewers

Abhishek Ranjan
Alok Jain
Anand K.
Anil Kumar
Anindya Saha
Anup Gangwar
Anurag Midha
Apu Sivadas
Arun Chandrashekhar
Arun Ramakrishnan
Ashish Mathur
Basant Dwivedi
Bhaskar Karmakar
D. R. Bhaskar
Dwayne Lee
Frank Vahid
Ganesan Thiagarajan
Jagdish Rao
Jai Balakrishnan
Jairam Sukumar
Jayashree Saxena
Kaushik De
Ken Butler
Kolin Paul
M. Balakrishnan
Mahesh Mehendale
Manu Chopra
Nikil Dutt
Nitin Chandrachoodan
Nitin Chawla
Pallab Dasgupta
Pradeep Kumar
Pravin Jain
Raj Mitra
Raj Senani
Rajan Arora
Rajeev Sehgal
Rajendra Pratap
Rajesh Gupta
Ranga Vemuri
Ranjit Dash
Reene Tayal
Ricky Bedi
Rubin Parekji
S. Dharamrajan
Sandeep Bhatia
Sandeep Kumar
Sandeep Pagey
Sanjeev Saluja
Saraju Mohanty
Shabbir Batterywala
Subhahshish Mukherjee
Sushil Gupta
Udaykumar H.
Venugopal Puvvada
Vinod Kathail
Vivek Pawar

VLSI Design 2008
Fellowship Recipients

Ajith Kumar Panda	*NIS & T, Berhampur*
Anilkumar V. Nandi	*B.V.B. College of Engg. & Tech.*
Arunachalam.V	*SES, VIT University*
Babita Roslind Jose	*Cochin University*
Bhanu Murthy Bhaskara	*Nalla Malla Reddy Coll. Of Engg.*
Bhavesh	*NIT, Silchar*
B.Karthikeyan	*SES, VIT University*
Bishnu Prasad Das	*IISC*
C.D.Naidu	*VNR Institute of Engg.& Tech.*
C.N. Shariff	*Bellary Engg.College*
Surajit Das	*Vidya Sagar College, Kolkata*
Dipak Kumar Kole	*St. Thomas College of Engg.& Tech.*
Darvinder singh Oberoi	*University of Jammu*
Ahsan Raja Chowdhury	*University of Dhaka*
Ghanshyam Singh	*MNIT*
H.Mangalam	*S.K. Collge of Engg. &Tech, Coimbatore*
B.P. Harish	*Visveswaraya College of Engg.*
Devasarmahiren Kumar	*SMIT, Sikkim*
Indranil Hatai	*IIT, Kharagpur*
Jayarajan. R	*Anna University, Chennai*
K. Sivani	*Kakatiya Institute of Technology*
Chitrasena Bhat	*KIT, Tiptur*
Krishnendu Choudhury	*NIT, Durga pur*
K. Sivasankaran	*SES, VIT University*
Lakshmi Prabha Viswanathan	*Govt. College of Technology*
T. Laxmikandan	*Anna University, Chennai*
Prof. Kamaraju	*GEC, Gudlavalleru*
Ms.Madhusmita Panda	*CV Raman College of Engg.*
Manu. T.M	*Bellary Engg.College*
D. Meghanathan	*Anna University, Chennai*
Nachiketa Das	*MERI*
N.B. Balamurugan	*Thiagarajar College of Engg.*
M. Nirmala Devi	*Amrita Vishwa VidyaPeetham*
N. Rajkumar	*NHCE*
N. Siva Sankar Reddy	*Vasavi College of Engg.*
Prof. N.S. Murthy	*NIT, Warangal*
Sreehari Rao Patri	*NIT, Warangal*
Pradyut Sarkar	*MCKV Institute of Engg.*
Priyabrata Pattananaik	*Silcon Institute of Technology*
S.Rajaram	*Thiagarajar College of Engg.*
Rajib Kar	*NIT, Durga pur*
K. Venkata Ramanaiah	*Narayana Engg.College, Nellore*
Ramesh Bhakthavatchalu	*Amrita Vishwa VidyaPeetham*

Aytha Ramesh Kumar	*VNR Institute of Engg.& Tech.*
C.V. Krishna Reddy	*Kakatiya Institute of Technology*
Rockey Gupta	*University of Jammu J & K*
Pranab Roy	*BES University, Shibpur*
R.Sakthivel	*V IT University*
M.Sreenivasulu	*Nalla Malla Reddy Coll. of Engg.*
Saroja V Siddamal	*JNTU, Kukatpally*
Saurabh Choudhury	*NIT, Silchar*
S.Sivanantham	*VIT University*
Samrat L. Sabat	*Central University*
S.Muthukumar	*VIT University*
Sriyankar Acharyya	*West Bengal University of Tech.*
S. Moorthi	*NIT, Tiruchirapalli*
M.Srinivasa Rao	*VNR Institute of Engg.& Tech.*
Susheel Kumar Sharma	*University of Jammu J & K*
S.Vasu Krishna	*VNR Institute of Engg.& Tech.*
P. Vijay Kumar	*PSG College of Tech., Coimbatore*
Vinayak	*VNIT, Nagpur*
V Vinoth Thyagarajan	*Thiagarajar College of Engg.*
Madhavi latha	*JNTU*
Rajeshwari M Bankar	*BVB College, Engg.& Tech.*
S.P. Venu MadhavaRao	*Osmania University*
Mohammed Abid Hussain	*IIIT, Hyderabad*
Amit Harode	*VNIT, Nagpur*
Aseem Gupta	*California*
Rahul M Badghare	*VNIT, Nagpur*
Chandan Karfa	*IIT, Kharagpur*
Rathna	*Anna University, Chennai*
Falguni Sinhababu	*ViT University*
Prashanth Jnanendra	*IIT, Madras*
Arun Janarthanan	*University of Cincinnati*
Jimson Mathew	*University of Bristol*
Janakiraman V	*IISc, Bangalore*
K. Navaram Kumar	*VNIT, Nagpur*
Kameswara Rao Chepette	*IIT, Delhi*
Kaushik Bhattacharya	*IIT, Kharagpur*
G. Krishnaiah	*IIT, Delhi*
Rasmita Panda	*Berhampur University*
Mehdi Fazeli	*Sharif University of Tech., Azadi*
Susmit Maiti	*BES University, Shibpur*
Mamatha.S, Mrs.	*IIIT, Hyderabad*
G. Muneeswari	*IIT, Madras*
Nagaraju Pothineni	*IIT, Delhi*
Neeraj Goel	*IIT, Delhi*
Noor Mahammad Sk	*IIT, Madras*
Pejman Lotfi Kamran	*CAD Research Group*
K. Praveen Kumar Reddy	*VNIT, Nagpur*
K. Shyamala	*IIT, Madras*
S. Ramasamy	*NIT, Trichy*
R.V. Kshirsagar	*VNIT, Nagpur*

Ravindra Jayanthi	*IIIT, Hyderabad*
Ravindra Mukhiya	*IIT, Kharagpur*
Rupam Mukherjee	*IIT, Kharagpur*
Vikas. G	*IISc, Bangalore*
S.D. Thimmappa	*IIT, Delhi*
Chandrashekhar T Kukade	*VNIT, Nagpur*
B.V.N. Silpa	*IIT, Delhi*
Narasimham K	*C.C.C.M / B.A.R.C*
Soumyajit Dey	*IIT, Kharagpur*
Vijay Sundaresan	*University of Cincinnati*
Supata Ditti	*BES University, Shibpur*
Uma Rajaram	*Anna University, College of engg.,*
P.V. Sankara Rao	*IIT, Kharagpur*
Falguni Sinhababu	*BES University, Shibpur*
Rabindra Kumar Pradhan	*IIT, Kharagpur*
Rangababu	*University of Hyderabad*
Abhay S Kochhar	*VNIT, Nagpur*
Abhishek Mittal	*ViT University*
Alladi. Mahish	*VNIT, Nagpur*
Amitava Banerjee	*IIT, Kharagpur*
Amey M. Walke	*VNIT, Nagpur*
Anurag Sharma	*ViT University*
Archana Devi .B.R.	*ViT University*
Aishwarya Bharathi Sankara	*University of California, Irvine*
V. Ashis Kumar	*N.I.T.K. Surathkal*
Ashis Maity	*IIT, Kharagpur*
Lingamneni Avinash	*IIIT, Hyderabad*
Biswajit Ray	*IISc, Bangalore*
Chester Rebeiro	*IIT, Madras*
Deepa Kannan	*Arizona State University*
Nithin B.Dev	*N.I.T.K. Surathkal*
Vikas Singh	*IIT, Madras*
Ganpat anant parulekar	*VNIT, Nagpur*
Hassan Raza	*VNIT, Nagpur*
Hitesh Kumar Gupta	*IISc Bangalore*
Jayalakshmi Periyasamy	*VNIT, Nagpur*
Prakash.J	*ViT University*
Jyotirmoy Ghosh	*IIT, Kharagpur*
T.V. Kalyan	*IIIT, Hyderabad*
Kamalika Datta Ms	*IIT, Kharagpur*
Karthik T J	*IIT, Madras*
M Kiran Kumar Reddy	*IISc, Bangalore*
Kranthi Kumar.Guduru	*N.I.T.K. Surathkal*
J.S. Krishnam Raja Reddy	*ViT University*
J. Lavanya	*IIT, Madras*
Manas Kumar Lenka	*IISc Bangalore*
Shashidhar L	*IISc, Bangalore*
Manikumar Ravula	*N.I.T.K. Surathkal*
Shyam Shroff	*IIT, Madras*
M. Vijaya Raju	*University of Hyderabad*

Partha Mukhopadhyay	*IIT, Kharagpur*
Narendra Konidala	*N.I.T.K. Surathkal*
Sreeram N S	*IISc, Bangalore*
N S Vinay	*IISc, Bangalore*
Arun.K	*Anna Univ, College of Engg.,*
Omprakash Parida	*IISc Bangalore*
Lalit Patnaik	*IISc Bangalore*
Prabhu Natarajan	*Anna University*
Pradeep Nair	*IISc, Bangalore*
T. Prashanth Reddy	*VNIT, Nagpur*
Raja Shekar Guntupally	*N.I.T.K. Surathkal*
Rajesh Kumar Dasari	*IIT, Madras*
Prasenjit Ray	*BES University, Shibpur*
Rooha Razmid Ahamed	*VNIT, Nagpur*
Reiley Jeyapaul	*Arizona State University*
Rajashekar n	*IISc, Bangalore*
B. Sandeep	*VNIT, Nagpur*
Bhukya Hariprasad	*VNIT, Nagpur*
R. Rajasekhar	*Anna University, College of Engg.,*
Mohammed Shoaib	*IIT, Madras*
Sivakumar Bondada	*IISc, Bangalore*
Sreekanth Reddy P	*IIIT, Hyderabad*
Sreehari Veeramachaneni	*IIIT, Hyderabad*
K. Srimannarayana Karthik	*Sathyabama University*
Santanu Sarkar	*IIT, Kharagpur*
Subho Chatterjee	*IIT, Kharagpur*
Subir Kumar mandal	*VNIT, Nagpur*
Sudip Roy	*IIT, Kharagpur*
Sumit Samajpati	*IIT, Madras*
Sunil shah	*IIT, Delhi*
Virendra Thakur	*IIT, Madras*
Mangaiarkkarasi.K	*ViT University*
Umakant Goyal	*IISc Bangalore*
Vani S	*ViT University*
P. Venkateswarlu	*NIT, Trichy*
Virendra Kumar Patidar	*IIT, Madras*
Anurag Avinash Zope	*VNIT, Nagpur*
G.Palakshi	*IIT, Madras*
S.V. Mohanasundaram	*NIT, Trichy*
Suraj Sindia	*IISC, CEDT*
Deepa Devendran	*IIT, Madras*
V. Raghavendra	*IIIT, Pune*
Saurav Bandyopadhyay	*IIT, Kharagpur*
Amal Kumar Kundu	*IIT, Kharagpur*
Ajaykumar Devarapalli	*University of Hyderabad*
M. Kirthi Krishna	*IIIT, Hyderabad*
Mohit Sharma	*Malaviya Institute of Technology, Jaipur*
Rahul Choudary	*Malaviya Institute of Technology*
Aranab Ray	*Malaviya Institute of Technology*
Vysakh	*Amrita University, Kerala*

Navneet Menon	*Amrita University*
Ashwin Menon	*Amrita University*
Ranjith C	*National Institute of Technology, Warangal*
Ms Gayatri	*National Institute of Technology*
Venugopal. M	*National Institute of Technology*
Swagat Nanda	*VIT*
Anusha Kakarala	*VIT*

VLSI Design 2008
Best Paper Awards

Arun Kumar Choudhury Best Paper Award
"Integrated TIA-Equalizer for High Speed Optical Link"

Saurav Bandyopadhyay (*Indian Institute of Technology, Kharagpur*)
Stephen E. Ralph (*Georgia Institute of Technology, USA*)
Pradip Mandal (*Indian Institute of Technology, Kharagpur*)
Kenneth Pedrotti (*University of California, Santa Cruz, USA*)

Nripendra Nath Biswas Best Student Paper Award
"Recursive Statistical Blockade: An Enhanced Technique for Rare Event Simulation with Application to SRAM Circuit"

Amith Singhee (*Dept. of ECE, Carnegie Mellon University, Pittsburgh, USA*)
Jiajing Wang (*ECE Dept., University of Virginia, Charlottesville, USA*)
Benton H. Calhoun (*ECE Dept., University of Virginia, Charlottesville, USA*)
Rob A. Rutenbar (*Dept. of ECE, Carnegie Mellon University, Pittsburgh, USA*)

Honorable Mention Award
"Compact Modeling of Suspended Gate FET"

Y.S. Chauhan (*Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland*)
D. Tsamados (*Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland*)
N. Abelé (*Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*)
and (*ST Microelectronics, Crolles, France*)
C. Eggimann (*Center for Integrated Systems, Stanford University, Stanford, CA USA*)
M. Declercq (*Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*)
A.M. Ionescu (*Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland*)

Honorable Mention Award
"Optimal Dual-VT Design in Sub-100 Nanometer PDSOI and Double-Gate Technologies"

Aditya Bansal (S*chool of Electrical and Computer Engineering, Purdue University, West Lafayette, USA*)
Jae-Joon Kim (*IBM T. J. Watson Research Center, Yorktown Heights, USA*)
Keunwoo Kim (*IBM T. J. Watson Research Center, Yorktown Heights, USA*)
Saibal Mukhopadhyay (*IBM T. J. Watson Research Center, Yorktown Heights, USA*)
Ching-Te Chuang, (*IBM T. J. Watson Research Center, Yorktown Heights, USA*)
Kaushik Roy (*School of Electrical and Computer Engineering, Purdue University, West Lafayette, USA*)

VLSI Design Conference History

Meeting Sequence	Place	Dates	Number of Papers	Number of Posters	Number of Tutorials	Proceedings Pages
First	Madras, India	Dec. 26-28, 1985	29	0	1	193
Second	Bangalore, India	Dec. 15-18, 1988	26	21	4	496
Third	Bangalore, India	Jan. 6-9, 1990	30	22	4	390
Fourth	New Delhi, India	Jan. 4-8, 1991	45	16	9	315
Fifth	Bangalore, India	Jan. 4-7, 1992	57	24	4	378
Sixth	Bombay, India	Jan. 3-6, 1993	70	9	6	371
Seventh	Calcutta, India	Jan. 5-8, 1994	87	0	6	448
Eighth	New Delhi, India	Jan. 4-7, 1995	77	6	6	456
Ninth	Bangalore, India	Jan. 3-6, 1996	75	16	6	480
Tenth	Hyderabad, India	Jan. 4-7, 1997	84	18	6	608
Eleventh	Chennai, India	Jan. 4-7, 1998	98	0	6	624
Twelfth	Goa, India	Jan. 7-10, 1999	103	0	6	682
Thirteenth	Calcutta, India	Jan. 3-7, 2000	93	0	6	590
Fourteenth	Bangalore, India	Jan. 3-7, 2001	77	0	9	592
Fifteenth	Bangalore, India	Jan. 7-11, 2002	109	0	8	834
Sixteenth	New Delhi, India	Jan. 4-8, 2003	84	0	6	622
Seventeenth	Mumbai, India	Jan. 5-9, 2004	120	44	8	1132
Eighteenth	Kolkata, India	Jan. 3-7, 2005	113	23	9	922
Nineteenth	Hyderabad, India	Jan. 3-7, 2006	136	0	11	880
Twentieth	Bangalore, India	Jan. 6-10, 2007	147	0	15	990
Twenty First	Hyderabad, India	Jan. 4-8, 2008	108	0	10	780
Twenty Second	New Delhi, India	Jan. 5-9, 2009	79	0	9	632

Embedded Systems Conference: History

Meeting Sequence	Place	Dates	Number of Papers	Proceedings Pages
First	New Delhi, India	Jan. 2-4, 2002	8	70
Second	New Delhi, India	Jan. 4-8, 2003	84	622
Third	Mumbai, India	Jan. 5-9, 2004	120	1132
Fourth	Kolkata, India	Jan. 3-7, 2005	113	922
Fifth	Hyderabad, India	Jan. 3-7, 2006	136	880
Sixth	Bangalore, India	Jan. 6-10, 2007	147	990
Seventh	Hyderabad, India	Jan. 4-8, 2008	108	780
Eighth	New Delhi, India	Jan. 5-9, 2009	79	632

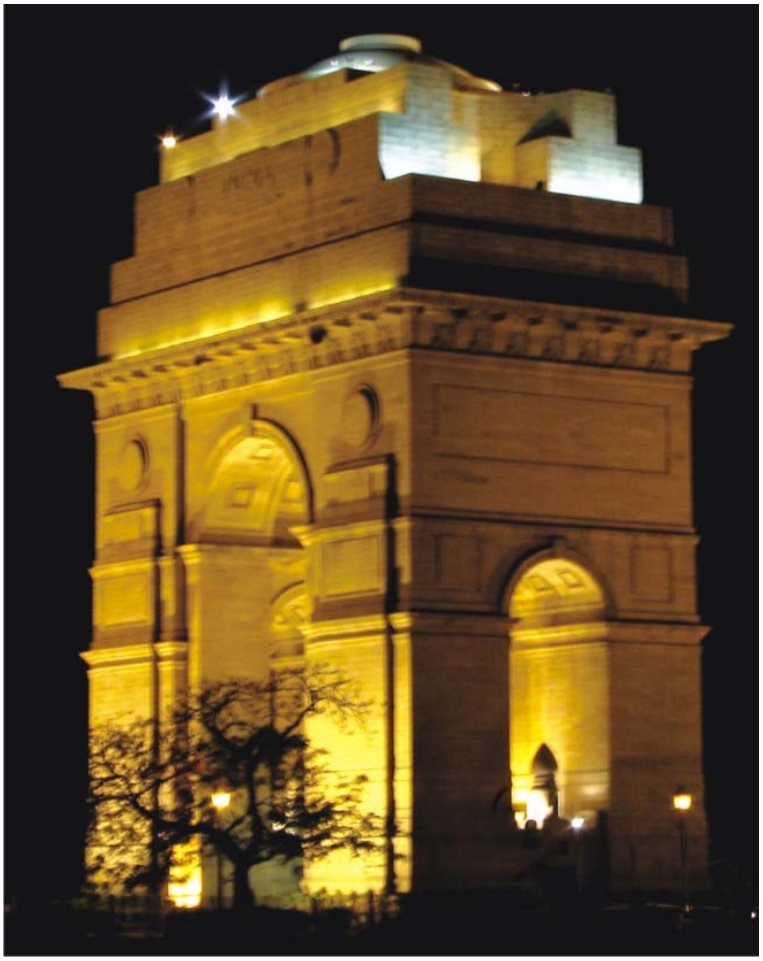
About the Cover

India Gate, an important monument of the city, is a memorial built in commemoration of more than 80,000 Indian soldiers who were killed during World War I. At the base of the India gate there is another memorial, the Amar Jawan Jyoti (Flame of the immortal warrior), a never-ceasing flame under the humongous arch to remind the nation of soldiers who laid their lives to serve this nation. India Gate was designed to convey to the people of India the values of "**Duty, discipline, unity, fraternity, loyalty, service and sacrifice**".

Delhi is the capital of India, a country that blends an amazing history and a dynamic future creating an enchanting experience for all of its visitors, a vibrant dynamic masterpiece, the largest democracy and second most populous nation on earth. It offers a unique cultural mix blending ancient and noble cultures, modern and dynamic societies.

New Delhi is the only city in the world with 3 World Heritage Sites – The Red Fort, The Qutub Minar and The Humayun's Tomb.

Delhi has had the honour to host many magnificent events in the past. In 1951, the inaugural edition of the Asian Games was organized in Delhi and again in 1982. Now the Capital city of Delhi is preparing to host the most prestigious Commonwealth Games in 2010.



VLSI Design 2009

Invited Keynote Speakers

Justin R. Rattner
Intel Senior Fellow
Vice President
Director, Corporate Technology Group and
Intel Chief Technology Officer

Abhi Talwalkar
President and Chief Executive Officer
LSI Corporation

Neil Henderson
General Manager,
Embedded Systems Division (ESD) of Mentor Graphics

Thomas W. Williams
Synopsys Fellow, Synopsys, Inc.

Dr. Ajoy K. Bose
Chairman of the Board, Founder,
President and Chief Executive Officer Atrenta Inc.

Prof. Jacob A. Abraham
Electrical and Computer Engineering
University of Texas, Austin

Invited Talks/Special Sessions Abstracts

A Decade of Platform-Based Design: A look backwards, a look forwards

Grant Martin, Chief Scientist, *Tensilica*

Abstract:

It has been 10 years since a group of us wrote the book "***Surviving the SoC Revolution**: **A Guide to Platform-Based Design***", and almost a decade since I gave a talk at VLSI 2000 in Kolkata about this theme. The intervening time has seen considerable development in the platform based design approach. It has become the near ubiquitous approach to the development of complex SoCs for many application areas. It has branched out from its original, mainly hardware-centric focus, to assume much more of a system and software focus complementing hardware. And the nature of platform architectures have changed: we now see many more embedded processors of all kinds in SoC platforms, from application-specific processors (ASIPs) to clusters of homogeneous or heterogeneous processing engines and many integrated subsystems each including one or more ASIPs or general purpose cores.

This talk will look back at the past decade in platform based design and describe the evolution of architectures, design approaches and tools, and also look forward at the next decade or two and try to paint some possible scenarios for the future evolution of the platform-based approach. As we move towards new generations of design tools and higher level design approaches, what will be the main forms of platforms in future and how will designers use them?

Speaker Bio:

Grant Martin is a Chief Scientist at Tensilica, Inc. in Santa Clara, California. Before that, Grant worked for Burroughs in Scotland for 6 years; Nortel/BNR in Canada for 10 years; and Cadence Design Systems for 9 years, eventually becoming a Cadence Fellow in their Labs. He received his Bachelor's and Master's degrees in Mathematics (Combinatorics and Optimisation) from the University of Waterloo, Canada, in 1977 and 1978.

Grant is a co-author or co-editor of nine books dealing with SoC design, SystemC, UML, modelling, EDA for integrated circuits and system-level design, including the first book on SoC design published in Russian. His most recent book, *ESL Design and Verification*, written with Brian Bailey and Andrew Piziali, was published by Elsevier Morgan Kaufmann in February, 2007.

He was co-chair of the DAC Technical Program Committee for Methods for 2005 and 2006. His particular areas of interest include system-level design, IP-based design of system-on-chip, platform-based design, and embedded software. Grant is a Senior Member of the IEEE.

DOI 10.1109/VLSI.Design.2009.100

Analog IC Design in Nanometer CMOS Technologies

Prof. Willy Sansen, *K.U.Leuven Belgium*

Abstract:

In nanometer CMOS technologies, several new effects emerge, such as velocity saturation and gate leakage currents. As a result the transconductance and speed are both limited by velocity saturation. Also noise and mismatch are affected as a result of the thinner gate oxides used. Moreover the supply voltage is reduced to values below 1 Volt, creating new challenges for analog circuit design.

This presentation provides a review of the modifications in model parameters, including noise and distortion. It is followed by an exploration of the noise/power compromise in existing circuit blocks such as Miller operational amplifiers and Gm-C filters. An overview is the given of low-voltage amplifiers/filters configurations with both Gate and Bulk drives. Several sub-1 Volt circuits are finally discussed for different applications.

Speaker Bio:

Prof. Willy Sansen has the PhD degree from the University of California, Berkeley in 1972. Since 1980 he has been full professor at the Catholic University of Leuven, in Belgium, where he has headed the ESAT-MICAS laboratory on analog design since 1984. He has been supervisor of sixtyfive PhD theses and has authored and coauthored more than 625 publications and sixteen books, among which is ***Analog Design Essentials***. He is a Fellow of the IEEE. He was program chair of the ISSCC-2002 conference and is now President of the IEEE SSCS.

DOI 10.1109/VLSI.Design.2009.101

Common Power Format: A User-driven Ecosystem for Proven Low Power Design Flows

Dr. Sumit DasGupta, Senior Vice President, *Si2*

Abstract:

Low power design has emerged as one of the urgent needs in IC design. The International Technology Roadmap for Semiconductors (ITRS) has identified the challenges surrounding low power design as one of the fundamental bottlenecks in exploiting the full capabilities of some of the advanced technology nodes. In fact, data from major chip design houses have underscored this need.

Much attention has been focused world-wide on the three existing formats for expressing low power constraints and intent: Common Power Format (CPF) from Silicon Integration Initiative (Si2), UPF 1.0 from Accellera, and UPF 2.0/P1801 from IEEE. However, the real challenge lies in the development of design flows and tools that exploit the content expressed by designers in these formats to solve real-life, power-related issues in design. Therefore, it should come as no surprise that at Si2 the focus has been on both developing and standardizing CPF in a coalition of both users and EDA suppliers, and in creating an ecosystem that provides training and adoption aids for CPF to support its adoption by chip designers and tool developers alike and proliferation of CPF into design flows in IC companies around the world.

This presentation begins with a brief introduction on Si2 and the Low Power Coalition (LPC) and the processes used in LPC to drive the development of CPF. There will be a discussion on the CPF roadmap with an introduction of the current standard CPF version 1.1 identifying the key enhancements over the previous version 1.0, and the roadmap leading to version 1.2 where interoperability with P1801 is one of the focus items. Next, we will describe some of the enablers provided by Si2 to support adoption, such as, training materials, a parser, a reference guide and a relational analyzer which can be used both to train in CPF as well as to analyze the contents of multiple CPF files used across the design. The talk will include examples of adoption by EDA companies and will conclude with results achieved to-date among IC design companies with references to some real-life success stories in low power design.

Speaker Bio:

Sumit DasGupta is the Senior Vice-President of Engineering at Silicon Integration Initiative (Si2) an electronics industry consortium based in Austin, TX. Prior to joining Si2, Sumit was at Motorola Semiconductors, now Freescale, where he served as Director of Design Systems and was responsible for developing and integrating tools for the design of PowerPC microprocessors and SoC designs. Before Motorola, Sumit was at IBM where he served in senior management and technical leadership positions in the EDA field and was responsible for developing design tools and methods for physical design, and design for test, many of which are still in use today. Sumit has a PhD in Computer Science from Syracuse University. He has 8 patents issued and over 25 papers and publications. He is a Senior Member of the IEEE and has served in several leadership positions in IEEE events and activities.

DOI 10.1109/VLSI.Design.2009.102

The Future of Low Power Design is Here: IEEE P1801, aka, UPF 2.0

Stephen Bailey, Director Product Marketing, *Mentor Graphics*

Abstract:

Industry adoption of Accellera's Unified Power Format (UPF) has been broad and swift. And why shouldn't it be? For the first time, UPF made it possible to specify the power design intent in combination with the HDL specification of the design for use throughout the design, verification and implementation flows. UPF's portability and feature set opened the door for more efficient design of low power systems. Now, UPF 2.0 is just around the corner. The IEEE P1801 working group, by the time of this conference, will have completed the sequel and it will be well on its way to IEEE standardization. Rumors are that there are significant changes to UPF 2.0. Why has been UPF been enhanced? What value will the new capabilities deliver? Do the changes obsolete UPF 1.0? This presentation will provide an overview of the major changes in UPF 2.0, its relationship with UPF 1.0 and the value that everyone doing low power designs will want to know.

Speaker Bio:

Stephen Bailey is Director of Product Marketing for Functional Verification at Mentor Graphics. He has been active in EDA standards activities including chair of IEEE P1801 and Accellera UPF, past chair of IEEE VHDL 1076 working group and participating in the PSL 1850 working group. He is the past technical program chair, vice chair and general chair for the DVCon Conference (2004-2008). With over 15 years of EDA and electronics industry experience, he has served in R&D, applications, consulting, and technical and product marketing roles. Stephen has a BS and MS in Computer Science from Chapman University.

DOI 10.1109/VLSI.Design.2009.103

Making Sense Out of the Potential Babble of Low Power Standards

Dr. Gary Delp, Distinguished Engineer, *LSI Corp*

Abstract:

For decades designers have worked with the digital abstraction, signals are either logical true or logical false. As with all abstractions, this one had great utility, allowed optimizations in analysis, and separated two areas of difficult analysis, making the design task achievable. In 2009, this abstraction becomes more valuable, and more complex. Parts of digital circuits will be turned off relative to other parts, parts will enjoy low-power slow-down modes, and parts will scream with performance and energy. The good news is that there is a simple way to express the relationships, boundaries, activities, and side effects of many power domains without having to give up most of the simplifications that the digital abstraction allow us. The bad news is that there are currently two ways to do it.

Using examples from a number of design flows and design problems, the speaker will show how to use both UPF/P1801 and CPF to express the power constraints and characteristics of designs. As work is ongoing in both the Si2 Low Power Coalition, and the IEEE P1801 groups, the January state of interoperability will be greater than it is currently, and much quicker and cleaner to hear about than it has been to develop.

Speaker Bio:

Dr. Gary Delp is a Distinguished Engineer working out of the CTO office at LSI. As one of three architects of the Low Power Coalition, and vice-chair of the IEEE P1801 working group, he is in a unique position to provide insight into interoperability needs and potentials. Gary spends his time working on design and IP reuse, inside of a design, across designs, and across the economic eco-system. Some of this reuse is in the form of bundles and IP functions, some is in the form of formats, methodologies, and exploratory work. Standards Setting bodies, Industry Alliances and University research programs support this work of technology transfer.

He is the Technical Director of The SPIRIT Consortium, and the CTO of the VSI Alliance. He is also the vice-chair of the IEEE study group on common power formats. LSI has a keen interest in power reduction in service of the needs of their customers in the storage and consumer industries.

His work has always been in the area of system optimization, but the systems have varied. As a VLSI Designer at the IBM AS/400 Division, he led teams in the optimization of hardware/software tradeoffs for network interconnect and the provision of network services. He holds patents in scheduling and shaping algorithms, circuit design, chip product structures, and video editing among others. He works collaboratively; the bulk of his 40+ patents are joint with others. At the University of Delaware, his PhD. Dissertation was MemNet: a distributed shared memory network implementation and architecture. IBM Watson work included FDDI and ATM operating system/hardware interfaces.

He holds a Bachelor of Arts degree in Theatre from Oberlin College and a Master of Fine Arts in Technical Theatre from the University of Memphis, Tennessee. Delp received his PhD in Electrical Engineering from the University of Delaware and has taught in several institutions of higher education including Rhode Island College, Memphis State University, and the University of Delaware. Once, as technical director, he led a team in building 2 mountains for an outdoor historical drama in Chillicothe, Ohio.

DOI 10.1109/VLSI.Design.2009.104

DFX and Productivity

Dr. Robert C Aitken, R&D Fellow, *ARM*

Abstract:

CMOS scaling has led to ever-increasing numbers of potentially available transistors on chips. At the same time, design productivity has also continued to improve, but has not been able to keep up, resulting in increasing design effort. Many factors contribute to this situation, but one key element is the complexity involved in ensuring that yield targets will be met. (DFY). This talk outlines the basics of design-for-yield (DFY) and shows how it relates to design-for-manufacturability, test, and variability (DFM, DFT, and DFV respectively). It is shown how a comprehensive approach to all of the problems, known as DFX, can lead to improved design methodology and hence improved productivity.

Speaker Bio:

Robert C. Aitken is an R&D Fellow at ARM. His areas of responsibility include low power design, library architecture, and design for manufacturability. He has worked on design, manufacturing and variability issues for many years at ARM, Artisan, Agilent and HP. He has given tutorials and short courses on a variety of subjects at conferences and universities worldwide. He has published over 70 technical papers, and holds a Ph.D. degree from McGill University in Canada.

DOI 10.1109/VLSI.Design.2009.105

Computational Lithography – Moore Bang for your Buck

Dr. Vivek Singh, Senior Principal Engineer, *Intel*

Abstract:

There have been many pronouncements about the slowing down of Moore's Law. Human enterprise, however, has managed to disprove these dim prophecies by producing ingenious solutions on a regular basis, to allow Moore's Law to continue its unabated march. Many of these solutions are coming from the growing field of Computational Lithography. Generally speaking, Computational Lithography comprises a broad set of techniques that use physics-based calculations to eke out more lithographic performance from today's steppers than they were originally designed for. Given the extraordinary cost of lithography tools and the fact that economics drives Moore's Law as much as physics, this boost in IC affordability is a key driver of innovations in Computational Lithography.

One such innovation is Pixelated Phase Mask technology. This technology was created to address the problem caused by the growing gap between the lithography wavelength and the feature sizes patterned with it. As this gap has increased, the quality of the image has deteriorated. About a decade ago, Optical Proximity Correction was introduced to bridge this gap, but as this gap continued to increase, one could not rely on the same basic set of techniques to maintain image quality. We sought to alleviate this problem by introducing additional degrees of freedom within the mask. The resulting Pixelated Phase Mask technology will be described in this paper, as an example of how Computational Lithography can contribute to affordable scaling and design productivity.

Speaker Bio:

Vivek Singh obtained his Ph.D. in 1993 from Stanford University, where he worked on simulations and experiments in plasma etching and deposition. He joined the TCAD department at Intel, where he worked on many different aspects of lithography technology development and optimization. Currently, he is a Senior Principal Engineer, and the Manager of Computational Lithography Group at Intel, responsible for all tool development in the area of OPC, rigorous lithography simulation, double patterning, and inverse lithography. Vivek also represents Intel on several DFM forums, and is currently Chair of the SPIE DFM Conference.

DOI 10.1109/VLSI.Design.2009.106

Made For India Forum

Made For India Forum

Chair: Rajiv Kapur
Organizers: Chitra Giridhar, Sanjeev Mehta, and Anurag Gupta

India is an emerging market that is being noticed and taken seriously the world over. The >1B population is very young (35% under the age of 15) with an ever-expanding middle class. Sustained 8% GDP growth, rapid urbanization, increasing dispensable income, changing lifestyles, opening economic policy, and democratic society are fuelling 'explosive' consumerism in India resulting in persistent demand for latest goods and services.

India as a market encourages/requires creative product design or business models to meet its needs. This uniqueness comes from its demographics, its economics, and its inherent diversity. Hence this transmogrification of the India story: *from "**Made in India**" of yester years - "**Made for India" today** -* **"Made by India" tomorrow.**

In this context we have created a platform titled ***"Made For India"***. This is a platform to help identify and promote products that are centered around semiconductors, electronics and embedded SW, designed for the Indian market. We have created this platform for eminent speakers to start with the end market and showcase use of technology, business models/services that were established to meet its needs.

Speakers:

1. Rajeev Kaushal, Head - *Pune Division, eInfochips*
2. Manish Gupta, *Associate Director, IBM India Research Laboratory*
3. Rajeev Agarwal, *CEO Innoviti*
4. C. Damodaran, *General Manager - Embedded & Networking Solutions at Network Systems and Technologies (NeST)*
5. Prasad Mantri, *Member ITRS, SUN*
6. Ramendra S. Baoni, *Managing Director, BiSquare Systems Pvt Ltd*
7. Dr. G. Venkatesh, *CTO, Sasken*

DOI 10.1109/VLSI.Design.2009.98

VLSI Design Conference 2009
Panels

IP Panel Topic:

Why is design automation and reuse of analog designs increasingly trailing the digital world?

Summary:

Demand for high performance in today's systems requires some key IP components to be designed in analog, whereas quick turn around time requires significant use of digital IPs. While there has been significant progress in design automation and design reuse of digital circuits in the last couple of decades, much has not changed for analog design. Design capture in low abstraction level, poor automation of implementation and verification steps, worsening silicon variability etc. combined with several technology nodes/flavors put a lot of pressure on the limited analog design expertise available today.

This panel discussion focuses on these issues and possible solutions. Can some of these design functions be moved from analog domain to digital? What is involved in deploying some of the digital design paradigms and reuse concepts in analog design? What can EDA vendors do to address these issues? What can Fabs do to control process variability to enable reuse of Analog IPs?

Organizers:

Ghasi Agarwal, Prakash Bare

DOI 10.1109/VLSI.Design.2009.107

EDA Panel Topic:

EDA Made-in-India: Fact or Fiction?

Summary:

Advances in EDA technology have barely kept pace with increasing complexity of IC designs and growth in the semiconductor industry. Over the last 20 years, India has been playing an increasing role in the design evolution. Semiconductor companies in India are steadily increasing their contributions in leading-edge design work. Can the EDA industry in India keep pace with that trend? Is there enough momentum in innovation by EDA companies in India? Will we ever see a "Made-in-India" tag for EDA products and services?

EDA industry in India taps into a large talent pool that is proficient in electronic design, algorithms and software development. Can this be a key driving factor for the EDA industry?

On the other hand, recent focus in EDA industry has been in topics that affect manufacturing and yield, such as DFM, SSTA etc. Such topics warrant a deeper understanding of the semiconductor process technologies and physics of deep-submicron devices. Given that Indian semiconductor industry is mostly design oriented, with minimal experience in IC fabrication, will the EDA industry be curtailed in it's growth? Can Indian universities step up to play a significant role to groom the EDA industry, similar to the roles played by Stanford and UC Berkeley during the EDA industry's infancy.

Or, is it even worth for EDA companies in India to pursue the "Made-in-India" tag? Given the amount of evolutionary knowledge that needs to be gained as well as the focus needed on advanced research, isn't Indian EDA industry better off not pursuing the "Made-in-India" tag on EDA products?

The panelists will debate the topic from various angles and share their personal insight of the history, challenges and vision for EDA and semiconductor industry in India and world-wide.

Organizers:
Raman Santhanakrishnan, Yatin Trivedi

DOI 10.1109/VLSI.Design.2009.108

Automotive Sector/Made For India Panel Topic:

Solutions for a small car – Made for India and Made in India

Summary:

Majority of the Indian and overseas auto manufacturers have plans of developing a small car in India for the Indian consumers. According to industry sources, the key to success of the small car will be a combination of parameters such as the design, features, specifications, manufacturing efficiency, performance, marketing network and after sales service and the final competitive price. Keeping low cost pricing as a crucial aspect of the game, how can solutions developed for such a car be optimized in the most cost efficient manner?

What is the scope of optimization for a small car for the semiconductor industry? How much impact do the manufacturing processes along with the marketing activities have on the final success of the small car?

Organizers:

India Semiconductor Association

DOI 10.1109/VLSI.Design.2009.109

Embedded SW Panel Topic:

Accelerating Embedded System Design

Summary:

From rack-mounted data or telecom computer systems to mobile devices, embedded systems may take many forms but face very similar design issues. Power, cost, performance and time to market are critical, as always, but so too is the user interface, as shown by the highly successful Apple iPhone and the G1 Google Phone. That the user interface is so software dependent only adds to the on-going debate as to whether hardware or software is more conducive to a effective, fast-turnaround design. This panel will discuss where the emphasis should lie and also discuss the various forms of system design abstraction and virtualization techniques available to engineers to accelerate embedded system design.

Organizers:

Techonline

DOI 10.1109/VLSI.Design.2009.110

Tutorials

Tutorial T1

Defect Aware to Power Conscious Tests – The New DFT Landscape

Nilanjan Mukherjee & Janusz Rajski, Mentor Graphics Corporation, Wilsonville, OR, USA, *{nilanjan_mukherjee, janusz_rajski}@mentor.com*
Jerzy Tyszer, Poznań University of Technology, Poznań, Poland, *tyszer@et.put.poznan.pl*

Abstract

The rapid scaling of semiconductor devices along with technological innovations including material and process changes such as high-k gate dielectric, metal gate electrodes, etc., are making conventional fault models inadequate. In addition, the evolution of interconnects from single to multiple-levels, the use of new materials to meet the wire conductivity requirements and reduce dielectric permittivity, and the scaling of conventional metal/dielectric system have had significant impact on the performance and power dissipation of devices. Multi-core designs, heterogeneous component integration, and sophisticated packaging techniques further aggravate the challenge of testing such devices effectively. This tutorial will focus on some of the advances shaping the test industry today to address the above mentioned design and process changes. New fault models that are termed as "defect aware" are being proposed and there is a demand for test vectors targeting such defects. Bridging (static and dynamic), n-detect, stuck-open, inline-resistance, propagation delay, etc., are some examples of new fault models that are being used to various extent in the industry today. At the same time, at-speed testing is becoming the norm as the industry moves towards smaller technology nodes. Methods to handle false and multi-cycle paths effectively are common in practice to prevent unnecessary yield losses. For the first time, timing information is being considered during Automatic Test Pattern Generation (ATPG) targeting small-delay defects. Each one of the fault models will be discussed and the current trends in the industry along with some preliminary silicon experiments will be presented.

Another industry trend posing a serious challenge to manufacturing test is the advent of power-aware designs. Various techniques such as architecture driven voltage reduction, switching activity minimization, switched capacitance minimization, and dynamic power management are being deployed in designing low power devices. New DFT techniques are required as well to limit power dissipation during test (preferably matching the power dissipation in functional mode of operation) thereby preventing IR drops, voltage droop, or hot spots. Test pattern generation needs to be tweaked to consider power dissipation during the key steps of the algorithm. In this tutorial, methods to control power dissipation during test, both from DFT as well as test generation perspectives will be discussed.

As numerous fault models are being proposed, all the different pattern sets along with power-aware test pattern generation results in a significant increase in the number of patterns. This directly impacts test cost as both test application time as well as the tester memory needed to store the vectors are increasing. Compression schemes with not only aggressive compression ratios are being adopted; they need to fit into the low power DFT methodology. This tutorial will highlight some methods to handle low power designs with compression. Additionally, advanced compression schemes to handle very large compression ratios will be discussed.

In summary, the trend towards "zero DPM" manufacturing, especially in certain applications such as automotive, is pushing the envelope for high quality test sets as well as new DFT flows and methodologies. All the above factors that have direct impact on test have to be addressed to

DOI 10.1109/VLSI.Design.2009.111

handle complex and heterogeneous devices that are being manufactured with new material and technologies today. This tutorial will help in developing an understanding on how to address some of these issues and highlight DFT techniques and methodologies that are being adopted to achieve the desired results.

Speaker Biographies

Nilanjan Mukherjee received the B.Tech (Hons) degree in Electronics Engineering from IIT, Kharagpur, India, and a Ph.D. degree from McGill University, Montreal, Canada, in 1996. Dr. Mukherjee currently leads a technical group in the Design to Silicon division at Mentor Graphics Corporation. At Mentor Graphics, he was a co-inventor of the EDT technology and is a lead developer for TestKompress®. His research focuses on developing next generation test methodologies for DSM designs, test data compression, test synthesis, memory testing, and fault diagnosis. Prior to joining Mentor Graphics, he worked at Lucent Bell Laboratories, NJ. Dr. Mukherjee has published more than 40 technical articles in various IEEE journals and conferences and is a co-inventor of 17 US patents. He was an invited author for the special issue of the IEEE Communications Magazine, June 1999. Dr. Mukherjee was the co-recipient of the Best Paper Award at the 1995 IEEE VLSI Test Symposium and the best student paper award at the Asian Test Symposium in November 2001. Recently, he received the prestigious 2006 IEEE Circuits and Systems Society Donald O. Pederson Outstanding Paper Award recognizing the paper on embedded deterministic test published in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. Dr. Mukherjee has presented tutorials at several conferences including ITC, DAC, VLSI Design, and have offered DFT seminars on behalf of Mentor Graphics in the US and India.

Janusz Rajski received the M.Eng degree in electrical engineering from the Technical University of Gdańsk, Poland, in 1973, and the Ph.D. degree in electrical engineering from Poznań University of Technology, Poland, in 1982. From 1973 to 1984, he was a member of the faculty of Poznań University of Technology. In June 1984, he joined McGill University, Montreal, Canada, where he became Associate Professor in 1989. In January 1995 he accepted the position of Chief Scientist at Mentor Graphics Corporation, Wilsonville, OR. His main research interests include design automation and testing of VLSI systems, design for testability, built-in self-test, and logic synthesis. He has published more than 150 research papers in these areas and is co-inventor of 30 US and international patents. He is also the principal inventor of Embedded Deterministic Test (EDT™) technology used in the first commercial test compression product TestKompress®. He is co-author of *Arithmetic Built-In Self-Test for Embedded Systems* published by Prentice Hall in 1997. He was co-recipient of the 1993 Best Paper Award for the paper on logic synthesis published in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* co-recipient of the 1995 and 1998 Best Paper Awards at the IEEE VLSI Test Symposium, co-recipient of the 1999 and 2003 Honorable Mention Awards at the IEEE International Test Conference, as well as co-recipient of the 2006 IEEE Circuits and Systems Society Donald O. Pederson Outstanding Paper Award recognizing the paper on embedded deterministic test published in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. Dr. Rajski has given presentations at all major international conferences in the area of testing. He has also presented advanced seminars and tutorials to many companies in Europe, Japan and the USA.

Jerzy Tyszer received the M.Eng (Hons.) degree in electrical engineering from Poznań University of Technology, Poland, in 1981, the Ph.D. degree in electrical engineering from the same university in 1987, and Dr. Habilis degree in telecommunications from the Technical University of Gdańsk, Poland, in 1994. From 1982 to 1990, he was a member of the faculty of Poznań University of Technology, Poland. In January 1990, he joined McGill University,

Montreal, Canada where was Research Associate and Adjunct Professor. In 1996, he assumed the position of Professor at the Institute of Electronics and Telecommunications of Poznań University of Technology, Poznań, Poland. His main research interests include design automation and testing of VLSI systems, design for testability, built-in self-test, embedded test, and computer simulation of discrete event systems. He was co-recipient of the 1995 and 1998 Best Paper Awards at the IEEE VLSI Test Symposium, the 2003 Honorable Mention Award at the IEEE International Test Conference, and the 2006 IEEE Circuits and Systems Society Donald O. Pederson Outstanding Paper Award recognizing the paper on embedded deterministic test published in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. He has published 7 books, more than 90 research papers in the above areas and is co-inventor of 21 US and international patents. Dr. Tyszer is co-author of *Arithmetic Built-In Self-Test for Embedded Systems* published by Prentice Hall in 1997 and the author of *Object-Oriented Computer Simulation of Discrete Event Systems* published by Kluwer Academic Publishers in 1999. In 1999, he was a guest co-editor of the special issue of the *IEEE Communications Magazine* devoted to testing of telecommunication hardware. He has served on technical program committees of various conferences including the IEEE VLSI Test Symposium and the IEEE European Test Symposium. He is a senior member of the IEEE.

Tutorial T2

Techniques for the Design of Low Voltage Power Efficient Analog and Mixed Signal Circuits

J. Ramirez-Angulo, New Mexico State Univ, Las Cruces NM, USA, *jramirez@nmsu.edu*
Ramon G. Carvajal, Escuela Superior de Ingenieros, Universidad de Sevilla, Sevilla, Spain, *carvajal@gte.esi.us.es*
Antonio Lopez-Martin, Universidad Publica de Navarra, Pamplona, Spain, *antonio.lopez@unavarra.es*

Abstract

Emerging applications in various fields, such as Ambient Intelligence scenarios or remote biomedical monitoring, currently demand wireless sensor networks with transceivers having extremely low power consumption requirements. This is a key issue in order to decrease battery weight and size and to increase the lifetime of the battery, which usually in these sensing nodes is not replaceable. To achieve these strict power requirements, several solutions have been proposed at various layers. At the physical layer, savings in power consumption are achieved by low-voltage operation and optimized power-to-performance ratio. Supply voltages of 1V (or less) are anyway mandatory in modern deep submicron technologies to operate reliably due to the extremely thin oxide. Furthermore reduction of the supply voltage (even of not required) strongly reduces power consumption in digital circuits since it scales with supply voltage. Although this is not so simple in analog circuits, they should operate at the same supply voltage than the digital part in mixed-mode systems to avoid the complexity involved in generating various supply voltages.

The canonic way of designing analog circuits consist in using high-gain amplifiers with passive components in negative feedback loops, both in continuous-time or discrete-time form. Sometimes amplifiers are operated in open loop (e.g. Gm-C filters, some VGAs, etc.), and in this case a large linear range is required for the amplifier at the expense of gain. In any case, amplifiers play a key role in analog design, and their power consumption directly impacts that of the overall analog system. Such amplifiers usually take the form of Operational Transconductance Amplifiers (OTAs) with high output resistance, typically driving capacitive loads, or operational amplifiers with low output resistance able to drive low resistive loads. Besides low-voltage and power-efficient operation, these amplifiers should feature a fast settling response, not limited by slew rate. Conciliating all these requirements is difficult with conventional class A topologies, since the bias current limits the maximum output current. Hence a trade-off between slew rate and power consumption do exists [1]. To overcome this issue, class AB topologies are often employed. These circuits provide well-controlled quiescent currents, which can be made very low in order to reduce drastically the static power dissipation. However, they automatically boost dynamic currents when a large differential input signal is applied, yielding maximum current levels well above the quiescent currents.

Several class AB amplifiers have been proposed. Most of them are based on adaptive biasing techniques, by including extra circuitry that increases quiescent currents (e.g. by increasing tail currents in differential stages). However, often the extra circuits included increase both power consumption and silicon area, and add significant parasitic capacitance to the internal nodes. Also positive feedback is often employed to get boosting of dynamic currents, which makes difficult to guarantee stability considering process and temperature variations. In this work we illustrate the use of new circuit design techniques to achieve low-voltage class AB amplifiers that combine

DOI 10.1109/VLSI.Design.2009.112

simplicity and power efficiency. These techniques allow introducing class AB operation at the input stage and at the active load of the amplifier with minimum penalty in other performance parameters. The tutorial is divided into several sections. Section 2 presents the concept of Super Class AB amplifiers and various circuit implementations. As an application, a Sample and Hold circuit is described in Section 3. Section 4 covers the design of class AB amplifiers using quasi-floating gate transistors. Their application in a VGA and a sigma-delta modulator for a wearable electroencephalogram monitoring system is described in Section 5.

Speaker Biographies

Jaime Ramírez-Angulo (F'00) received the degree in communications and electronic engineering (Professional degree), the M.S.E.E. degree from the National Polytechnic Institute and Director of the Mixed-Signal VLSI Lab at the Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM. He was Professor at the National Institute for Astrophysics Optics and Electronics (INAOE) and at Texas A&M University. His research is related to various aspects of design and test of analog and mixed-signal very large scale integrated circuits. He has made numerous contributions to this field which has been reported in over 400 publications. He has two high technology patents. Prof. Ramírez-Angulo received the prestigious Westhafer Award, which is the highest faculty award for research merits at New Mexico State University in Mexico City, and the Dr.-Ing. degree from the University of Stuttgart, Stuttgart, Germany, in 1974, 1976, and 1982, respectively. He is currently Klipsch Distinguished Professor.

Antonio J. Lopez-Martin (M'04) received the M.S. and Ph.D. degrees (with honors) from the Public University of Navarra, Pamplona, Spain, in 1995 and 1999, respectively. Currently, he is an Associate Professor with the Public University of Navarra, and an Adjunct Professor with the New Mexico State University. His current research interests include wireless transceivers and sensor interfaces with emphasis on low-voltage low-power implementations. He has published more than 200 technical contributions in books, journals, and conferences. He also holds two international patents, leads various research projects, and is consultant for local companies. Dr. Lopez was an Associate Editor of the *IEEE Transactions on Circuits and Systems–II: Express Briefs* during 2006–2007. He is currently serving as an Associate Editor for the *IEEE Transactions on Circuits and Systems –I: Regular Papers*, for the *IEEE Latin-American Learning Technologies Journal*, and for other technical journals. His recent awards include the Caja Navarra Research Award in 2007, the Young Investigator Award from the Complutense University of Madrid in 2006, the 2005 IEEE Transactions on Education Best Paper Award, and the European Center of Industry and Innovation (CEIN) Award in 2004 for excellence in transfer of research results to industry.

Ramón González Carvajal (M'99–SM'04) was born in Seville, Spain. He received the Electrical Engineering and Ph.D. degrees (with honors) from the University of Seville, Seville, Spain, in 1995 and 1999, respectively. Since 1996, he has been with the Department of Electronic Engineering, School of Engineering, University of Seville, where he has been a Lecturer (1996), Reader (2002) and Professor (2007). He was invited researcher at the Klipsch School of Electrical Engineering, New Mexico State University (NMSU), Las Cruces, NM, in the summers of 1999 and 2001–2004, and also at the Electrical Engineering Department of Texas A&M University in 1997. He also holds the position of Adjunct Professor at the Klipsch School of Electrical Engineering, NMSU. He has published more than 70 papers in international journals and more than 180 in international conferences. His research interests are related to low-voltage low-power analog circuit design, A/D and D/A conversion, and analog and mixed-signal processing.

Tutorial T3

Power Reduction Techniques and Flows at RTL and System Level

Anmol Mathur, Calypto Design Systems, Santa Clara, CA, USA , *amathur@calypto.com*
Qi Wang, Cadence Design Systems, San Jose, CA, USA, *qwang@cadence.com*

Abstract

Power reduction is becoming a critical design criterion for ASIC/SOC designers. Reducing both dynamic and leakage power is imperative to meet power budgets for portable devices as well as to ensure that the systems that these ASICs meet their packaging and cooling costs. In addition, the power of an ASIC has a significant impact on its reliability and manufacturing yield. Traditionally, most automated power optimization tools have focused at gate-level and physical level optimizations. However, major power reductions are only possible by addressing power at the RTL and system levels. At these levels, it is possible to make the sequential modifications needed to reduce power and energy consumption via techniques like sequential clock gating, power gating, voltage/frequency scaling and other micro-architectural techniques.

The focus of this tutorial will be on techniques for power reduction at the RTL and system level. It will also focus on expressing power intent at system and RTL levels and the flows needed to use that power intent in tools for functional verification, RTL-level optimization, logic synthesis and physical design. The following sections describe the key focus areas in the tutorial.

We will start by discussing the key trends in the semiconductor industry and in CMOS technology and relate them to the need for power-aware design flows all the way from system-level design, through micro-architecture definition and RTL design and implementation. We will then present different power and energy metrics that are used at different points in the design cycle and for different purposes such as average power of a system, peak power of a system, energy per cycle etc. We will relate these metrics to their typical use and discuss when a metric should be used and optimized. State-of–the-art techniques for estimation of power and energy metrics will be presented including those for software power estimation, energy estimation for applications on a system, RTL and gate-level power estimation etc. We will discuss both simulation-based and statistical techniques for estimating switching activity in a design.

Since, creation of system-level models is becoming a standard part of the design flow in most design teams, we will present typical flows from system-level models to RTL and the kinds of power/energy tradeoffs done at these levels such as power islands and mode identification, memory and bus architectures, voltage scaling and scheduling, and identification of clocking schemes and clock domains.

Since power of a design is a function of how it performs a computation over time, almost all the major transformations that have significant impact on the power of a design are sequential in nature – they change the sequence of values generated at key internal registers or memories in time. We will discuss the sequential optimizations like, sequential clock gating, power gating, dynamic voltage scaling and memory banking. The impact of these optimizations on verification and implementation flows will be highlighted and solutions to verification and implementation issues will be presented.

In the last few years, standards have started emerging to allow designers to express power intent such as voltage islands and power modes in a design. These are allowing for the same power intent to be seen by all the tools in the RTL flow: RTL simulation, logic synthesis, place and

DOI 10.1109/VLSI.Design.2009.113

route, logic equivalence checking and any other post-layout tools. Both CPF and UPF attempt to specify this information. We will focus on the key information that the emerging power formats need to support and how this impacts the RTL design and implementation flow. Specific power optimizations enabled by such information in implementation tools will also be discussed.

We will conclude the tutorial with a few case studies of specific designs where power optimization and power-aware design techniques discussed earlier in the tutorial were used. Practical issues in using these techniques will be highlighted. We will also discuss extensions to the power formats needed to support system-level to RTL power-aware flows and automatic optimizations to enable low power design.

Speaker Biographies

Anmol Mathur is the CTO and a co-founder at Calypto Design Systems, a company that is leveraging sequential analysis techniques for power optimization and sequential equivalence checking. Prior to Calypto, he was the architect of the datapath synthesis and optimization group at Ambit Design Systems and Cadence, where he led technology development and productization. Earlier, Anmol was part of an award-winning team at the MIPS division of SGI that developed and successfully deployed an RTL to gate-level equivalence checker and property checker within the MIPS microprocessor division. Anmol holds multiple patents and has published several papers in the areas of formal verification, logic synthesis, low power optimization and arithmetic optimization techniques. Anmol has been part of the technical program committee of DAC since 2005 and has participated in the program committees of ICCAD, MTV and MEMOCODE conferences in the past. He has presented tutorials and invited presentations on sequential equivalence checking and low power optimization at several conferences and universities over the last 4 years. Anmol participates in the Format Working Group as part of the Si2 Low Power Consortium that is defining and standardizing the CPF standard. Anmol received a Bachelor of Technology in Computer Science and Engineering from the Indian Institute of Technology, Delhi, where he was awarded a gold medal for highest academic achievement by the institute. He received an MS and PhD in Computer Science from the University of Illinois at Urbana-Champaign.

Qi Wang is a Senior Architect in the Front End Design group of Cadence Design Systems. He is the architect of the low power synthesis technology in RTL-Compiler and the Common Power Format before it was donated to Si2. Currently he is responsible for driving R&D activities in Power Forward Initiative and CPF based low power flow aross Cadence products. He is also one of the architects of the Si2 Low Power Coalition and vice Chair of the Format working group. His research interests include low power synthesis, high level synthesis and timing optimization. He had more than 20 papers published in various international conferences and journals. He also holds several US patents on synthesis and low power technologies. Dr. Wang holds a B.S. degree in Applied Electronics from Shanghai Jiao Tong University, an MSEE degree from Southern Illinois University and a Ph.D degree in Electrical and Computer Engineering from University of Arizona.

Tutorial T4

Security and Dependability of Embedded Systems: A Computer Architects' Perspective

Jörg Henkel, University of Karlsruhe, Karlsruhe, Germany, *henkel@informatik.uni-karlsruhe.de*
Vijaykrishnan Narayanan, Pennsylvania State University, USA, *vijay@cse.psu.edu*
Sri Parameswaran, University of New South Wales, Australia, *sridevan@cse.unsw.edu.au*
Roshan Ragel, University of Peradeniya, Sri Lanka**,** *roshanr@ce.pdn.ac.lk*

Abstract

Designers of embedded systems have traditionally optimized circuits for speed, size, power and time to market. Recently however, the dependability of the system is emerging as a great concern to the modern designer with the decrease in feature size and the increase in the demand for functionality. Yet another crucial concern is the security of systems used for storage of personal details and for financial transactions. A significant number of techniques that are used to overcome security and dependability are the same or have similar origins. Thus this tutorial will examine the overlapping concerns of security and dependability and the design methods used to overcome the problems and threats. This tutorial is divided into four parts: the first will examine dependability issues due to technology effects; the second will look at reliability aware designs; the third, will describe the security threats; and, the fourth part will illustrate the countermeasures to security and reliability issues

Part I: Dependability Issues due to Technology Effects and Architectural Countermeasures
Moore's law has been in place for more than four decades. Each new technology node provided advantages in basically all major design constraints (performance, power, area, etc.). When migrating to upcoming technology nodes it will become obvious that this win-win situation soon will be at an end. Or, in other words, in future it becomes far more difficult and expensive to migrate to new technology nodes. One major point is an inherent undependability which will become a challenging problem. Undependability addressed within this part of the tutorial is related to a) Fabrication and Design-Time Effects like "Yield and Process Variations" and "Complexity" as well as b) run-time effects as "Aging Effects", "Thermal Effects" and "Soft Errors". The first part of this tutorial will give the details of these effects and a prospect of how these effects might influence future architectures for embedded systems. An overview of selected state-of-the-art paradigms and approaches is given including a focus on organic computing principles as well as run-time adaptive embedded processor architectures that can deal with dependability issues.

Part II: Reliability Aware Design for Embedded Systems
Design of robust embedded systems meeting stringent quality, reliability, and availability requirements is becoming increasingly difficult in advanced technologies. The current design paradigm which assumes that no gate or interconnect will ever operate incorrectly within the lifetime of a product must change to cope with such failures. New architectural features are required for robust system design with built-in mechanisms for failure tolerance, detection and recovery during normal system operation. This part of the tutorial will focus on new design techniques required for building robust systems: concurrent error detection, recovery, and self-repair. A broad spectrum of circuit-level, logic-level, micro-architectural, hardware subsystem, and software techniques will be covered; the associated trade-offs among techniques will be presented. Implemented protection mechanisms are determined by a complex evaluation of power

DOI 10.1109/VLSI.Design.2009.114

and performance requirements and constraints, in addition to the vulnerability of specific circuits or structures to failures.

Part III: Security Attacks in Embedded Systems
Security of embedded computing systems is becoming a paramount concern as these devices become more ubiquitous, contain personal information and are increasingly used for financial transactions. Security attacks targeting embedded systems illegally gain access to the information on these devices or destroy information. Security threats in embedded systems could be classified by the means used to launch attacks. Typical launch methods are: physical, logical/software-based and side-channel/lateral attacks. Physical attacks refer to unauthorized physical access to the embedded system itself and are feasible only when the attacker has direct access to the system. Logical attacks exploit weaknesses in logical systems such as software or a cryptographic protocol to gain access to unauthorized information. Logical attacks are deployed easily against systems which are able to download and execute software and have vulnerabilities in their design. Side-channel attacks are performed by observing properties of the system (such as power consumption, electromagnetic emission, etc.) while the system performs cryptographic operations. This part of the tutorial highlights the most popular attacks on embedded computing systems.

Part IV: Countermeasures Against Security Attacks
A wide range of techniques have been proposed in the past to detect and counter security attacks in embedded devices. They could broadly be categorised into software based techniques and hardware assisted techniques. Software based techniques use software tools such as code analyzers and methods such as proof-carrying-code to overcome these attacks without changing the architecture of the processor. Hardware assisted techniques use additional hardware blocks or microarchitectural support to detect and protect against these security attacks. The talk gives an overview of countermeasures against logical and side-channel attacks. The most prominent up-to-date countermeasures are discussed in detail.

Speaker Biographies

Jörg Henkel is currently with Karlsruhe University (TH), Germany, where he is directing the Chair for Embedded Systems CES. Before, he was with NEC Laboratories in Princeton, NJ. His current research is focused on design and architectures for embedded systems with focus on low power and reliability. Dr. Henkel has organized various embedded systems and low power ACM/IEEE conferences/symposia as General Chair and Program Chair and was a Guest Editor on these topics in various Journals like the IEEE *Computer* Magazine. He is/has been a steering committee member of major conferences in the embedded systems field like at ICCAD and is also an editorial board member of various journals like the IEEE TVLSI. He has given full/half-day tutorials at leading conferences like DAC, ICCAD, DATE etc. Dr. Henkel is the Editor-in-Chief of the *ACM Transactions on Embedded Computing Systems* (ACM TECS) and holds nine US patents.

Vijaykrishnan Narayanan is currently at the Computer Science and Engineering and Electrical Engineering Departments at Penn State. He is a member of the Embedded and Mobile Computing Design Center and his research/teaching interests are in the areas of energy-aware reliable systems, embedded systems, on-chip networks, system design using emerging technologies (3D and Nano) and computer architecture. His research is supported by grants from National Science Foundation, The Technology Collaborative, and DARPA. He leads an interdisciplinary project funded by NSF that has set up a one-of-a-kind accelerated soft-error testing facility at the Penn State Nuclear reactor. Dr. Narayanan is actively involved with various technical service activities. He served as general co-chair, *ISVLSI 2002*; general-chair, *ISVLSI 2003*; vice-general chair, *Nanonets 2007*, and as program co-chair for *GLSVLSI 2006, Nanonets 2006*, and *ISLPED 2007*.

He serves on the steering committees of ISVLSI and GLSVLSI conferences. He is currently the editor-in-chief of the *ACM Journal on Emerging Technologies in Computing Systems*. He also serves on editorial boards of *IEEE Transactions on CAD* and the *Journal of Low Power Electronics*. He has offered full-day tutorials at major architectural conferences including ASPLOS, ISCA and PACT on reliability and low-power.

Sri Parameswaran is a Professor in the School of Computer Science and Engineering, University of New South Wales. His research interests are in Systems Security of SoCs and MPSoCs, System Level Synthesis, Low power systems, High Level Systems and Network on Chips. He has served on the Program Committees of numerous International Conferences, such as the Design Automation Conference (DAC), Design and Test in Europe (DATE), the International Conference on Computer Aided Design (ICCAD), the International Conference on Hardware/Software Codesign and System Synthesis (CODES-ISSS, as TPC chair), and the International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES). He is also an associate editor of the *ACM Transactions on Embedded Computing Systems,* and the *EURASIP Journal on Embedded Systems.*

Roshan Ragel is a Lecturer at the Department of Computer Engineering, University of Peradeniya since December 2002. He completed his B.Sc. Engineering (Honours I) degree in November 2001 (Peradeniya) and his PhD in June 2007 (University of New South Wales - Embedded Systems Lab). His research interests include secure embedded processors, rapid embedded hardware/software co-design, and micro-architectural support for secure and reliable computing. Mostly, he works on secure embedded systems, mainly countermeasures against code-injection attacks. Amongst his published works is the book *Microarchitectural Support for Security and Reliability: an Embedded Systems Perspective*.

Tutorial T5

Design for Manufacturability and Reliability in Nano Era

Goutam Debnath, Intel Corporation, Hillsboro, OR, USA, *goutam.debnath@intel.com*
Paul Thadikaran, Intel Corporation, Hillsboro, OR, USA, *paul.thadikaran@intel.com*

Abstract

The bottom line of any company is to maximize the profit from any given product. There are many factors influencing the product design resulting in a profitable business. One of the biggest factors is the manufacturability of the product. It is becoming more and more crucial to meet the 6+6 (6 months for the development and 6 months for qualifying the product to ship to customer) product life cycle to accommodate the rapid changing technology hungry market demand. Smooth, reliable, and efficient product ramp through manufacturing is the key of success for meeting TTM, capturing higher percentage of total available market (TAM).

This tutorial is going to address the difficulties industries are facing today in designing manufacturing friendly highly complex giga-scale products in submicron technology. As we are heavily into deep submicron era, the error margin or the tolerance guard band is getting tighter and tighter with respect to the previous generation of fabrication process. On this note, it is important to pay attention to Design For Manufacturing (DFM) related issues early in the design cycle as oppose to later in the design. These include, however not limited to, all kinds debugging hooks in the design for easy debugging of billion of transistors in a given design, paying attention to manufacturing friendly physical design rules, making sure of adequate test coverage to toggle most of the design nodes, making sure optimal guard band is implemented for transistor degradation for the lifetime of the product, and last but not least, all reliability (ESD, EM/SH, LU, etc) related issues are resolved in pre-silicon design before Tape out. In the past, manufacturing issues were not given much attention; time has changed and designers must have to be more sensitive than ever before in addressing manufacturing related issues early in the design cycle.

In a nut shell, this tutorial will capture the must have knowledge for design engineers (irrespective of front-end or back-end) who are involved in high performance VLSI design, as DFM features moving upstream in the design cycle. Audience will walk out with a good understanding on how to integrate specific manufacturing concerns into a product's design to obtain a product that is easier to manufacture with excellent overall quality in a shortest development time.

This tutorial covers the following main topics in detail:

1. **Introduction:**
    a. 6+6 product development strategy and overview
    b. Technology Trend of VLSI products in the current design environment
    c. Design trend (Power optimization, verification complexity)
    d. High volume manufacturability challenges, such as Cost of test, DPM, reliability issues
2. **DFT:**
    a. DFT strategy & planning
    b. Fault models
    c. DFT features in nanotechnology designs
    d. DFT Features for test cost optimization
    e. Special test structures

DOI 10.1109/VLSI.Design.2009.115

3. **Design for reliability:**
    a. FIT estimation for Design
    b. Design features for reliability improvement
    c. Design for Burn-in support
    d. Design for transistor degradation
    e. ESD for high speed IO
    f. EM/SH requirement and it's verification
4. **Design for manufacturability:**
    a. Process and Layout interaction and specific guidelines to reduce manufacturing defect
    b. Manufacturing friendly design rules
    c. Opportunistic usage of special cells to improve design yield
5. **Post silicon Validation process for reliability and manufacturability:**
    a. Wafer and Package Test characterization
    b. ESD/EM/SH/BI validation
    c. HVM platform for DPM measurement
    d. Test flow optimization
    e. Test hole resolution process and DPM bucketing

Speaker Biographies

Goutam Debnath received a M.S. in Physics from University of St Louis, Missouri, USA and a M.S. in Electrical Engineering from Southern Illinois University, Carbondale, USA. Goutam is in the board of Industry Advisory Council for Electrical and Computer Engineering Department of Southern Illinois University. Goutam is the Intel QPI Technology Execution lead (TXT) and Manufacturing Program Manager (MPM) in Server Platform Group (SPG). He has acted as a manager on various microprocessor designs focusing on design, design automation. He has done 15 Intel based desktop processors and 4 Xeon processor in last 17 years of Intel's professional carrier. Goutam's primary focus is on achieving higher manufacturing excellence and smooth enablement of QPI technology across the industry. Goutam has published and co-authored 9 technical papers in numerous conference including IEEE and VLSI conference. He has two USA patents on clock distribution in control logic blocks and shared power grid distribution respectively.

Paul Thadikaran received his Ph.D. in Computer Science from State University of New-York, Buffalo. He is currently a Principal Engineer at the Client Platform Architecture Group in Intel Corporation and focused on development of next generation client platform architecture. He has been involved in various aspects of design and test of previous four generations of Intel's IA-32 CPU. He has managed design methodology development, CAD tool development and standard cell library development targeted for CPU designs for past seven years. His areas of interest include methods and algorithms for test VLSI design, test and platform architecture. Paul has published more than 20 papers in IEEE/Intel conferences and journals. He has also co-authored a book on I_{ddq} Testing published by Kluwer Academic Press. Paul has refereed several IEEE and ACM journals such as *IEEE Transactions on CAD* and *ACM Transactions on Design Automation in Electronic Systems (TODAES)*. He has also offered tutorials on high performance design and test at various International and Intel audiences.

Tutorial T6

Negative Feedback System and Circuit Design

Nagendra Krishnapura, Indian Institute of Technology, Madras, India, *nagendra@iitm.ac.in*
Shanthi Pavan, Indian Institute of Technology, Madras, India, *shanthi.pavan@iitm.ac.in*

Abstract

The negative feedback amplifier structure using an ideal integrator is derived. The time domain and frequency domain descriptions of the integrator are discussed. The response of the negative feedback amplifier in the time and frequency domains is analyzed. From these general conclusions are drawn about the behavior of negative feedback amplifiers.

The ideal integrator is realized using controlled sources and passive elements. This realization clearly shows the cause for finite dc gain in real opamps. The effects of finite dc gain are analyzed. Relationships between amplifier specifications such as speed and accuracy and opamp parameters such as unity gain frequency and dc gain are derived.

Methods of increasing the dc gain to improve accuracy are discussed. These lead to multistage amplifiers. The response of such systems in time and frequency domains are analyzed. It is shown that multistage amplifiers are potentially unstable. Stability conditions for negative feedback systems are discussed.

The gain around the negative feedback loop is computed. The significance of loop gain is illustrated. Stability criteria related to the loop gain such as phase margin and Nyquist's criterion are discussed. Frequently used criteria such as phase margin are clarified.

Multistage amplifiers are essential for realizing high accuracy. Different techniques of realizing high gains while retaining stability-increasing the output resistance, miller compensation, and feedforward compensation are shown.

There are various opamp architectures: folded/telescopic cascode; two stage miller compensated; feedforward compensated; and three stage. The design procedures for the two stage miller compensated opamp, the feedforward compensated opamp, and the three stage opamp are shown. These opamps will be compared in terms of their performance parameters-bandwidth, noise, power dissipation, slew rate, output swing.

The design of a 350MHz bandwidth continuous-time active-RC filter using feedforward compensated opamps is shown. Measurement results from chips designed at IIT Madras illustrate the benefits of the feed-forward opamp architecture for low power applications.

The design details of a three stage opamp with a DC gain exceeding 100dB is shown. The constraints on the design of different stages are evaluated. Simulation results of the opamp illustrate its suitability for a high precision application.

Speaker Biographies

Nagendra Krishnapura is an Assistant Professor of Electrical Engineering at the Indian Institute of Technology, Madras in Chennai. He obtained the B.Tech degree in Electronics and Communication Engg from the Indian Institute of Technology, Madras in 1996 and the masters and doctoral degrees from Columbia University, New York in 1998 and 2000 respectively. Between 2000 and 2005, he worked as a senior design engineer at Celight, Inc. and Multilink (later Vitesse Semiconductor) where he designed integrated circuits for high speed broadband communications. Since June 2005, he has been with the Department of Electrical Engineering of

DOI 10.1109/VLSI.Design.2009.116

the Indian Institute of Technology Madras, where he teaches, conducts research and consults for several companies in the areas of high speed analog circuit design and signal processing. Nagendra has been an adjunct faculty member at Columbia University, New York, where he taught several advanced courses on analog design. He is also involved in improving expertise in the areas of analog and mixed signal design in India through the Prof. K. Radhakrishna Rao foundation. He and Dr. Shanthi Pavan presented a tutorial on continuous time delta sigma data converters at the VLSI conference in 2008. Nagendra Krishnapura is an associate editor of *IEEE Transactions on Circuits and Systems, Part II: Express Briefs.*

Shanthi Pavan is an Assistant Professor of Electrical Engineering at the Indian Institute of Technology, Madras in Chennai. He obtained the B.Tech degree in Electronics and Communication Engg from the Indian Institute of Technology, Madras in 1995 and the masters and doctoral degrees from Columbia University, New York in 1997 and 1999 respectively. From 1997 to 2000, he was with Texas Instruments in Warren, New Jersey, where he worked on high speed analog filters and data converters. From 2000 to June 2002, he worked on microwave ICs for data communication at Bigbear Networks in Sunnyvale, California. Since July 2002, he has been with the Department of Electrical Engineering of the Indian Institute of Technology Madras, where he teaches, conducts research and consults for several companies in the areas of high speed analog circuit design and signal processing. Shanthi Pavan serves on the editorial board of the *IEEE Transactions on Circuits and Systems, Part I: Regular Papers*. Apart from having taught several short courses in industries (like Texas Instruments, ST Microelectronics, National Semiconductor, Genesys Microsystems and many others), Shanthi has offered a tutorial on System Level Aspects of A/D Converter Design in the VLSI Design Conference in Hyderabad in 2006 and, with Nagendra Krishnapura, a tutorial on oversampled delta sigma data converters in 2008. He is also involved in improving expertise in the areas of analog and mixed signal design in India through the Prof. K. Radhakrishna Rao foundation.

Tutorial T7-A

Synthesis and Testing for Low Power

Ajit Pal, Indian Institute of Technology, Kharagpur, India, *apal@cse.iitkgp.ernet.in*
Santanu Chattopadhyay, Indian Institute of Technology, Kharagpur, India, *santanu@ece.iitkgp.ernet.in*

Abstract

In recent years, power dissipation has emerged as the key issue not only for portable computers and mobile communication devices, but also for high-end systems. Reducing power dissipation is of primary importance in achieving longer battery life in portable devices. On the other hand, for high-end systems the cooling and packaging requirements are pushing the chip designers for low power alternatives. As a consequence, apart from the size, cost and performance, now-a-days power is also considered as one of the most important constraints. This has led to vigorous research in the synthesis of low-power and high-performance circuits and systems. Moreover, aggressive device size scaling used to achieve high-performance leads to increased variability due to short-channel and other effects. This, in turn, leads to variations in process parameters such as, $L_{eff,}$ N_{ch}, W, T_{ox}, V_t, etc. Performance parameters such as *power* and *delay* are significantly affected due to the variations in process parameters and environmental/operational (V_{dd}, temperature, input values, etc.) conditions.

Due to variability, the design methodology in the future nanometer VLSI circuits will essentially require a paradigm shift from deterministic to probabilistic and statistical design approach. The tight constraint on power dissipation has also created new challenges for testing low power VLSI circuits, as the traditional test techniques do not account for power dissipation during test application. It is now an accepted truth that test power is often much higher than the power consumed in normal operation, due to voluminous test data, test parallelization and the low correlation between successive test patterns.

The objective of this tutorial is to provide an overview of different aspects of low power circuit synthesis at various levels of design hierarchy. It will introduce techniques to optimize the performance and power in the presence of process variations. Low power testing techniques will also be discussed.

Topic outline:

1. Introduction and sources of power dissipation (30 minutes)
2. Switching power reduction techniques (30 minutes)
3. Leakage power reduction techniques under process parameter variation (30 minutes)
4. Importance of test power reduction (30 minutes)
5. Power minimization strategies for internal testing (30 minutes)
6. Power minimization strategies for external testing (30 minutes)

Speaker Biographies

Ajit Pal is currently a Professor in the Department of Computer Science and Engineering at Indian Institute of Technology Kharagpur. He received his M. Tech. and Ph.D. degrees for the Institute of Radio Physics and Electronics, Calcutta University in 1971 and 1976, respectively. Before joining IIT Kharagpur in the year 1982, he was with Indian Statistical Institute (ISI), Calcutta, Indian Telephone Industries (ITI), Naini and Defence Electronics Research Laboratory (DLRL), Hyderabad in various capacities. He became full Professor in 1988 and

DOI 10.1109/VLSI.Design.2009.117

served as head of Computer Center from 1993 to 1995 and head of the Computer Science and Engineering Department from 1995 to 1998. His research interests include Embedded Systems, Low-power VLSI Circuits, Sensor Networks and Optical Communication. He is the principal investigator of several Sponsored Research Projects including Low Power Circuits sponsored by Intel, USA. He has over 125 publications in reputed journals and conference proceedings and a book entitled *Microprocessors: Principles and Applications* published by Tata McGraw-Hill. He is the Fellow of the IETE, India and Senior Member of the IEEE, USA. The presenter has introduced a PG-level course entitled *Low Power Circuits & Systems*, which has become increasingly popular in IIT Kharagpur in the last couple of years. He has also delivered seminars on this topic in several places and coordinated a 2-week long workshop in the summer of 2007 as part of a project sponsored by the Department of Electronics, Government of India.

Santanu Chattopadhyay is currently an Associate Professor in the Department of Electronics and Electrical Communication Engineering at Indian Institute of Technology Kharagpur. He received his B.E. degree in Computer Science and Technology from Calcutta University in 1990. In 1992 and 1996 he received his M. Tech in Computer and Information Technology and PhD in Computer Science and Engineering degrees respectively, both from Indian Institute of Technology Kharagpur. Before joining IIT Kharagpur, he was with B.E. College, Howrah, West Bengal, and Indian Institute of Technology, Guwahati. His research interests include CAD tools for low power circuit design and test, System-on-Chip testing, Network-on-Chip design and test. He has got a number of projects on these topics from Dept. of Science & Technology, Govt. of India, Dept. of Information Technology, Govt. of India, and Motorola (India) Pvt. Ltd. He has more than 110 publications in refereed international journals and conferences. He is the co-author of the book on *Additive Cellular Automata – Theory and Applications*, published by the IEEE Computer Society Press in 1997. He has also written two text books on *Compiler Design* and *System Software*, published by Prentice-Hall of India, in 2005 and 2007, respectively. He has delivered many lectures on these topics at various forums.

Tutorial T7-B

Power Management for Mobile Multimedia: From Audio to Video and Games

Samarjit Chakraborty, National Univ. of Singapore, Singapore, *samarjit@comp.nus.edu.sg*
Ye Wang, National Univ. of Singapore, Singapore, *wangye@comp.nus.edu.sg*

Abstract

Multimedia applications today constitute a sizeable workload that needs to be supported by a host of mobile devices ranging from cell phones, to PDAs and portable game consoles. Battery life is a major design concern for all of these devices. Whereas both – the complexity of multimedia applications and the hardware architecture of these devices – have progressed at a phenomenal rate over the last one decade, progress in the area of battery technology has been relatively stagnant. As a result, currently a lot of effort is being spent to develop high-level power management and application tuning techniques to minimize energy consumption and thereby prolong battery life. Such techniques include dynamically scaling the underlying processor's voltage and clock frequency in response to a time-varying workload, powering down certain system components when not being frequently used, and backlight scaling in LCDs with controlled image-quality degradation. Some of the application tuning techniques include selectively ignoring certain perceptually-irrelevant computations during audio decoding, and injecting metadata with workload information into video clips which can then be used to accurately estimate the decoding workload at runtime for better power management.

In this tutorial, we plan to give a comprehensive overview of this area and discuss power management schemes for a broad spectrum of multimedia applications. In particular, we will talk about several power management and application tuning techniques specifically directed towards audio decoding, video processing and interactive 3-D game applications. Starting from the basics of power management for portable devices, we will discuss the necessary mathematical techniques, give high-level overviews of relevant algorithms and also present the hardware setup that is necessary to perform research and development in this area.

The main objective of this tutorial will be to cover various techniques for power management for audio, video and graphics-intensive game applications running on battery-operated portable devices. In particular, we would illustrate how power management techniques differ for audio, video and game applications and would present a number of techniques for each of these classes of applications. We would also give an overview of open research problems and the challenges facing this area. Finally, we would describe some of the hardware platforms that we have been using to conduct research in this domain and give demonstrations of selected power management techniques.

Speaker Biographies

Samarjit Chakraborty is an Assistant Professor of Computer Science at the National University of Singapore. He obtained his Ph.D. in Electrical and Computer Engineering from ETH Zurich in 2003. For his Ph.D. thesis, he received the ETH Medal and the European Design and Automation Association's "Outstanding Doctoral Dissertation Award" in 2004. His work has also received Best Paper Award nominations at DAC 2005, CODES+ISSS 2006 and ECRTS 2007, all of which are premier conferences in the real-time/embedded systems area. Samarjit's research interests are primarily in system-level power/performance analysis of embedded systems. He has extensively

DOI 10.1109/VLSI.Design.2009.118

published in major research forums on this topic including DAC, DATE, CODES+ISSS, ASP-DAC, RTSS and RTAS, and regularly serves on the technical program committees of many of these conferences. Over the last few years he has been working on various problems specifically related to power management of multimedia applications and have co-authored several papers and patents in this area. He has given invited talks on various topics related to design, modeling and analysis of embedded systems at various universities and industrial labs, including UC Berkeley, MIT, CMU, Philips, General Motors and Creative Technology Labs. His experience with conducting tutorials include (i) a tutorial at the IEEE International Conference on Multimedia & Expo (ICME) at Amsterdam in July 2005, entitled "Multimedia Processing on Multiprocessor SoC Platforms: What should Multimedia System Developers know about Architectural Design, Performance Analysis and Platform Management?" (jointly with Radu Marculescu from CMU and Paul Stravers from Philips Research), (ii) a half-day solo tutorial at the ACM Multimedia Conference (MM) at Santa Barbara in October 2006 on "Flexible Modelling and Performance Debugging of Real-Time Embedded Multimedia Systems", (iii) a tutorial at the VLSI Design Conference at Bangalore in January 2007 on "Performance Debugging of Complex Embedded Systems" (jointly with Abhik Roychoudhury from NUS), (iv) a tutorial at the ARTIST2 Winter School on Modelling, Testing, and Verification for Embedded Systems (MOTIVES) at Trento, Italy in February 2007 on "Interactive Performance Debugging of Real-Time Systems", (v) tutorial at the VLSI Design Conference at Hyderabad in January 2008 on "Programming and Performance Modelling of Automotive ECU Networks" (with S. Ramesh, General Motors R&D - India Science Lab), and (vi) a tutorial at the Design, Automation and Test in Europe Conference (DATE) at Munich in March 2008 on "Formal Methods in System and MpSoC Performance Analysis and Optimisation" (with Rolf Ernst from TU Braunschweig, Kai Richter from Symtavision, Hans Sarnowski from BMW, and Marco Bekooij from NXP).

Ye Wang received his Dr.-Tech. degree from the Department of Information Technology, Tampere University of Technology, Finland. In 2001, he spent a research term at the University of Cambridge, U.K., working with Prof. Brian Moore on compressed domain audio processing. He is currently an Assistant Professor with the Department of Computer Science, School of Computing, National University of Singapore. Dr. Wang has had a nine-year career with Nokia Research Center in Finland as research engineer and senior research engineer, where he worked on Digital Audio Broadcasting (DAB) receiver prototype development, optimization of perceptual audio coding algorithms, error resilient audio content delivery to mobile phones and compressed domain audio processing for multimedia applications on small devices. His research interests include audio compression and content-based processing, perception-aware and low-power audio processing, and error resilient content delivery to handheld devices via wireless networks. He holds a dozen patents in these areas and has published about 30 international journal and conference papers. He is a member of the technical committee, Coding of Audio Signals of the Audio Engineering Society; and a member of the Multimedia Communications Technical Committee, IEEE Communications Society. Dr Wang has also given a number of tutorials and courses on audio and video processing while at Nokia Research Center Finland and regularly teaches these topics at the National University of Singapore.

Tutorial T8

Robust Circuit Design: Challenges and Solutions

Saurabh K Tiwary, Cadence Research Labs, Berkeley, CA, USA, *stiwary@cadence.com*
Amith Singhee, IBM T J Watson Research Center, Yorktown Hts, NY, USA, *asinghe@us.ibm.com*
Vikas Chandra, ARM R&D, Sunnyvale, CA, USA, *vikas.chandra@arm.com*

Abstract

Scaling with Moore's law is taking us to feature sizes of 32nm and smaller. At these technology nodes designers are faced with an explosion in design complexity at all levels. In this tutorial we discuss three somewhat novel and particularly confounding dimensions of this complexity:

- Electrical complexity: Digital circuit designers have particularly benefited from abstractions of the underlying MOS devices while designing circuits. For them, a transistor is an ideal switch and a wire is a perfect short between two nodes. This simplified abstraction is the driving force behind our capability to design and verify chips with about a billion transistors today. However, with aggressive device scaling, the properties of the devices that are being manufactured today are moving further away from the abstractions that we have been using to verify our designs. In this section of the tutorial, we look at some of the recent trends along these lines and some of the techniques that designers use to extract ideal functionality from non-ideal devices. We use design examples, both from analog/mixed-signal (PLL, ADC design) and digital domain (clock tree, power network, static timing, etc.), as illustrative cases studies.

- Manufacturing complexity: Minimum feature sizes at 45nm are already a quarter of the wavelength of light used for lithography. Consequently, imperfections in manufacturing are unavoidable and large enough to significantly change the intended design, resulting in dreaded yield loss. Any design today has to satisfy stringent manufacturability and yield requirements. At the same time, the complexity of critical variation mechanisms renders any simplified methods, like corner analysis, ineffective. Design methods and tools are being changed at all levels of the design flow to improve yield prediction and increase manufacturing robustness. In this vein, this tutorial will cover a broad spectrum of topics: 1) relevant state-of-the-art manufacturing process steps at 45 nm (193 nm lithography, ion implantation, etc.) and the physical mechanisms resulting in electrical performance variations, 2) recently proposed design techniques for mitigating the electrical variability, and 3) recently proposed design tools for increasing robustness and predicting the yield impact of this variability. We will look at various design phases from circuit architecture down to post-layout, and at several applications from SRAMs to ASICs to analog.

- Reliability complexity: With nearly three decades of continued CMOS scaling, the devices have now been pushed to their physical and reliability limits. Transistors on the latest chips in 45nm technology are so small that some of their parts are just a few atoms apart. Designs manufactured correctly may become unreliable over time because of mechanisms like NBTI, gate oxide breakdown and soft errors. The impact of unreliability manifests as time-dependent variability where the electrical characteristics of the devices vary statistically in a temporal manner, directly translating into design uncertainty in manufactured chips. Scaling to sub-45nm technology nodes changes the nature of reliability effects from abrupt functional problems to progressive degradation of the performance characteristics. The material presented in this section of the tutorial is intended for designers to form a thorough

DOI 10.1109/VLSI.Design.2009.119

understanding of the physics and models of the various reliability mechanisms. The tutorial will also introduce various design techniques to make the design more robust, which can be readily applied to an SoC design.

To successfully optimize any aggressive design in the nanometer regime, a comprehensive understanding of these challenges and available solutions is essential. We will review the physical mechanisms underlying these design complexities, along with relevant models, design tools and techniques for managing them and obtaining design robustness.

Speaker Biographies

Saurabh Tiwary is a Research Scientist at Cadence Research Laboratories, Berkeley, USA. He received his Ph.D. degree in Electrical and Computer Engineering (ECE) from Carnegie Mellon University (CMU). He received his B. Tech. degree from Indian Institute of Technology, Kanpur and his Masters degree in ECE also from CMU. He has spent some of his summers working at IIT Kanpur (India), Department of EECS, RWTH, Aachen (Germany) and Neolinear (USA). He has been nominated for the best paper award at DAC and was awarded the CSSI fellowship during his graduate studies at Carnegie Mellon. He has published papers at refereed conferences like DAC, ICCAD, CICC, DATE, ITC, etc. and has two patents pending. He has served on the program committee of ICCAD and as reviewer for *Transactions on CAD, Transactions on VLSI, TCAS, DAC, FMCAD and ISCAS*. His research interests include design, macromodeling and simulation of analog circuits and mixed-signal systems.

Amith Singhee is a Research Staff Member at the IBM T. J. Watson Research Center. His research interests are in variation-aware design and design automation, design-oriented process variability characterization, analog synthesis and general circuit simulation and optimization. He obtained his Ph.D. and M.S. in Electrical and Computer Engineering from Carnegie Mellon University and his B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Kharagpur. He was with Neolinear, and subsequently Cadence Design Systems, from 2002 to 2004. Dr. Singhee is a recipient of the Arthur G. Milnes dissertation award (2008) for his work on fast DFM algorithms for scaled circuits, two best paper awards (DAC, 2002 and DATE, 2007), one best student paper award (VLSI Design, 2008), and the Silver Medal (IIT, Kharagpur, EE), 2000. His paper on memory yield prediction (DATE, 2007) was chosen as one of the 30 most influential papers in 10 years of DATE. He is a chapter author for the book *Embedded Memories for Nano-Scale VLSI* and holds a U.S. patent in the area of yield-driven circuit optimization.

Vikas Chandra is a Staff Researcher in the Corporate R&D group at ARM. He received his Ph.D. and M.S. degrees in Electrical and Computer Engineering from Carnegie Mellon University (CMU). He received his B.E. degree from the Birla Institute of Technology and Science (BITS), Plain, India. He has worked at Intel's Strategic CAD Labs, IBM Austin Research Lab and the Central R&D group at STMicroelectronics. Dr. Chandra has published papers at prestigious international conferences like ICCAD, DATE, FPGA, ICCD, DFTS etc. He has two approved US patents and three pending patents. Dr. Chandra serves as a vice-chair of the ACM/SIGDA Technical Committee on FPGA & Configurable Computing. He is the co-organizer of the SIGDA Design Automation Summer School (co-located with DAC). He has served or is serving as a Technical Program Committee member for CICC, ICCD, SELSE, DRV workshop and VLSI Design conference. Dr. Chandra's research interests are in high-performance & low-power custom circuit design, memory architecture, DFM and reliability aware design.

Session 1A

Low Power Design for Wireless Communication

Design-Space Exploration of Energy-Delay-Area Efficient Coarse-Grain Reconfigurable Datapath

Sohan Purohit
Department of Electrical and Computer Engineering
University of Massachusetts-Lowell-USA
sohan_purohit@uml.edu

Marco Lanuzza, Stefania Perri, Pasquale Corsonello
Department of Electronics, Computer Science and Systems,
University of Calabria
Rende (CS), Italy
{lanuzza,perri}@deis.unical.it
p.corsonello@unical.it

Martin Margala
Department of Electrical and Computer Engineering
University of Massachusetts-Lowell-USA
martin_margala@uml.edu

***Abstract*—This paper presents the VLSI design of a high data throughput, energy and area efficient data path targeted for DSP and multimedia applications. Three different implementations of the reconfigurable data path using static, dynamic domino and D3L logic styles are presented to serve as low power, high speed, and speed-energy optimized variants of the architecture. When implemented using ST Microelectronics 90nm 1V CMOS technology, the proposed data path leads to a maximum supported clock frequency ranging from 917 MHz to 1.2 GHz with a dynamic power consumption @ 500 MHz ranging from 788 µW to 1.02 mW.**

I. INTRODUCTION

Digital signal processing (DSP) and multimedia applications require large amount of data to be processed in a parallel and very often in a repetitive manner. These applications require high-performance computations alongside the capability of matching the rapid evolution of the algorithms. The simultaneous demand for high computational speed and flexibility makes reconfigurable architectures attractive solutions for DSP and multimedia applications.

Traditional fine grained FPGA-based solutions offer a high level of flexibility but suffer from large area overhead and high power consumption. On the contrary, Coarse Grain Reconfigurable Architectures (CGRAs) [1] seem to be a more feasible solution, as they provide relatively higher performance with lower power consumption and higher area efficiency for the targeted application domains.

In this paper, a high performance, energy and area efficient data path implementation to be part of the recently proposed CGRA [2], targeted for DSP and multimedia applications is presented. As shown in Fig.1, the reconfigurable system consists of a 2D array of Reconfigurable Cells (RCs) organized into a hierarchical two level pipelined structure [2]. The single RC forms the basic unit of the architecture. It is composed of an 8-bit Processing Element (PE), a 256 X 8-bit dual port SRAM and a control unit for overall synchronization. The PE forms the backbone for the RC being its computational unit.

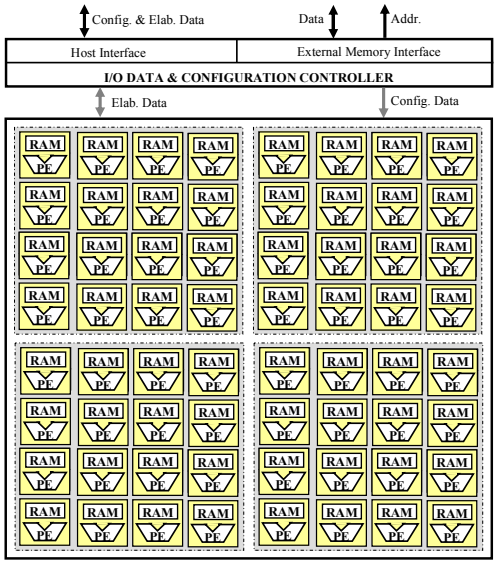


Figure 1: Top level architecture of the reconfigurable array [2].

In this work, three different transistor level implementations of the PE for high speed, low power and area efficient operations are presented and evaluated. More precisely, the PE was designed using static and domino design styles to provide for the low power and high speed applications respectively. The third implementation makes

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.33

use of Data Driven Dynamic Logic (D3L) [3] and acts as a good tradeoff between the high speed and low power approaches. All the circuits were custom implemented in the ST Microelectronics 90nm CMOS process.

The rest of the paper is organized as follows. Section II gives an overview of the PE architecture. Section III discusses in detail the three PE implementations. Post layout results are presented in Section IV. Finally, some concluding remarks follow.

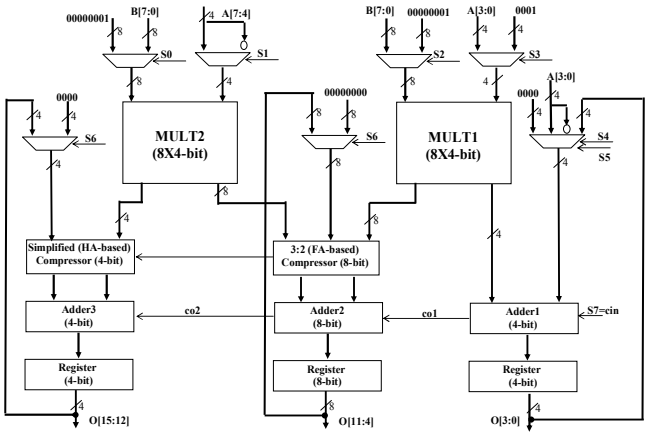


Figure 2: Logic organization of the Processing Element [1].

II. THE PROCESSING ELEMENT

The PE is responsible for all the arithmetic and accumulation operations performed by the RC. The block diagram of the PE is shown in Fig. 2. It consists of two 8X4-bit multipliers, a compressor stage, 16 bit carry linked adders and multiplexer-based auxiliary logic. These multiplexers driven by the control signals S7-S0 route the operands across the data path and ensure efficient data exchange between the arithmetic blocks, thus exploiting hardware reuse. Thanks to such organization, the circuit maintains the maximum functionality required for DSP elaborations with minimum hardware to cut down both silicon area occupancy and power consumption. The PE allows several single clock cycle 8-bit operations to be performed. Note that, the supported arithmetic operations include all the operations, which normally dominate multimedia and digital signal processing algorithms, such as two's complement addition, subtraction, accumulation, multiplication and multiply accumulation. When the 8-bit addition is performed, the PE calculates *B[7:0]×0001* and *A[7:4]×00000001* by the multipliers MULT1 and MULT2, respectively. Adders ADDER1 and ADDER2 compute *A[3:0]+B[3:0]+0* and *B[7:4]×0001+A[7:4]×00000001+co1*, respectively, and the 8-bit result is provided through the 8 least significant output lines *O[7:0]*. The 8-bit subtraction is performed in a similar way, but the operand *A[7:0]* is 2's complemented and the signal *cin* is forced to 1. The 8×8-bit multiplication is executed by using the two 8×4-bit multipliers to compute the two 12-bit partial results *B[7:0]×A[3:0]* and *B[7:0]×A[7:4]* and the three carry-linked adders to opportunely add these partial products. The 16-bit product is finally available through the output lines *O[15:0]*. Simple accumulation operation is performed by accumulating previously stored result with the current operand *A*. Similarly, multiply-accumulation is computed by adding the result of the previous computation with current partial products in the compressor stage.

It should be noted that the critical path of the PE includes only a 2:1 multiplexer, a 8X4-bit multiplier and three small carry linked adders. This ensures optimum performances, thus maximizing the total throughput of the system.

The PE has been implemented using static, dynamic domino and D3L [3] logic design styles to provide three variants of the same architecture for different target applications. These implementations are detailed in the following section.

III. PE IMPLEMENTATION STRATEGIES

It is clear, from the previous section, that the architecture of the PE has been designed to provide high flexibility in performing single clock cycle 8-bit arithmetic operations, along with high speed, low area occupancy and low power consumption. However, in order to emphasize such architectural benefits, an accurate transistor level design is needed. In fact, depending on the different circuit logic families considered for the PE implementation, several area-speed-power tradeoffs can be achieved.

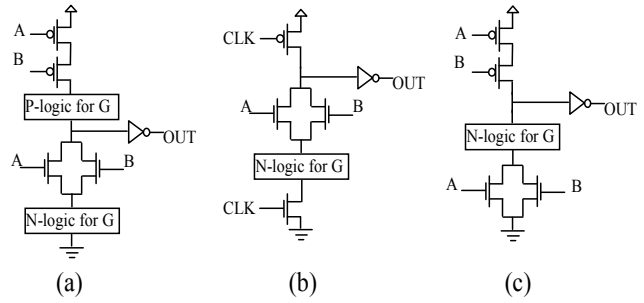


Figure 3: (a) Static, (b) domino and (c) D3L implementations of the function OUT= G· (A+B).

As shown in Fig. 3, in order to implement a specific function the conventional static CMOS logic style combines dual Pull-Down and Pull-Up Networks. For increasing speed, in domino dynamic logic, the Pull-Up Network (PUN) is replaced by a single PMOS precharge transistor which is controlled by the global clock. Compared to static CMOS logic, the input capacitance of every domino dynamic gate is significantly reduced. However, the usual requirement of an additional transistor (i.e. the NMOS evaluation transistor) that has to be cascaded with the Pull-Down Network (PDN), limits the gate speed during the low to high transitions. Moreover, due to the large load on the clock signal (that has to be connected to every dynamic gate) and due to the higher switching activity, circuits implemented using domino dynamic logic usually result in a considerably more power-

hungry solution compared to their conventional static CMOS counterparts [4]. One solution for reducing the excessive load of the clock-tree network is to precharge using local data instead of a global clock, as in D3L logic [3]. The most evident benefits offered by this approach over the dynamic domino implementation are the elimination of the clock distribution system (with the associated reduction in terms of power consumption), and the elimination of the clocked NMOS transistor, which reduces the evaluation path delay. As drawbacks, the input lines of D3L circuits have higher capacitances with respect to their dynamic domino counterparts and the precharge phase is no longer simultaneous for all the gates since a propagation path exists through cascaded stages [5].

By combining the proposed architecture with the previous mentioned circuit design styles, we evaluated three different implementations of the reconfigurable PE. The first, which is based on a standard static CMOS logic, ensures low power arithmetic operations with relatively good delay performance and is highly suitable for portable multimedia processing with strict power constraints. The dynamic domino based approach, provides a faster solution at the cost of more power dissipation. This version finds use in application environments that require extremely high operating speeds, where power optimization can be shifted to other stages in the cell. Finally, the D3L based implementation serves as a golden mean between the conventional static and domino implementations by providing high speed and relatively low power consumption.

A. Static CMOS implementation

The first implementation of the PE involves the design of all the basic arithmetic and logic components using standard CMOS static design style. Static implementation also offers the option of using low power circuit topologies like pass transistor logic [4]. However, the conventional CMOS static approach was chosen owing to its higher operating speed, higher signal integrity and symmetrical layout design.

The designed basic components include fundamental logic gates, Full Adders, 4- and 8- bit Carry-Look-Ahead (CLA)-based adders [4] and two 8 X 4-bit Wallace Tree multipliers. The Wallace Tree topology was selected because it provides efficient carry propagation for each stage of the multiplier, at the cost of relatively lower power and area compared to other topologies. Another advantage of the Wallace Tree approach is the highly regular structure of the multiplier, which facilitates layout design and guarantees better utilization of silicon area [4]. The final adder stage in each multiplier is an 8-bit CLA based adder. This speeds up the final addition stage, thus assuring performance efficiency.

All the basic circuits, belonging to the data path, were implemented using a conventional transistor sizing criterion [4]. More precisely, they were sized for symmetrical high to low and low to high propagation delays, approximately equal to those of the minimum size inverter (i.e. W_n=0.12um, W_p=0.24 um). This approach was chosen in order to control power dissipation.

The input and output registers of the PE are True Single-Phase Clocked (TSPC) edge triggered registers [6].

B. Dynamic Domino implementation

Dynamic Domino circuits have traditionally been known to provide high operating speeds [4]. They operate using a sequence of precharge and evaluation phases on the basis of a clock input. The basic domino approach requires a precharge PMOS device in the PUN and a *foot* NMOS in the PDN. Both of these devices are driven by the clock signal that synchronizes the alternate precharge and evaluate phases.

In the domino-based implementation of the PE, all the critical arithmetic circuits have been implemented using dynamic domino logic style. On the contrary, the multiplexers, which are relatively less critical from a performance point of view, are based on static CMOS implementations. This allows a reduction of the total power without significantly compromising the circuit performance.

It is worth underlining that the domino-based pull-down network of each gate has been designed using the same transistor sizing criterion which was used for the static implementation. Whereas, the pull-up network is now just a minimum sized precharge device. Moreover, as a counter-measure to the charge lost due to the PDN leakage, minimum sized keeper transistors have been used.

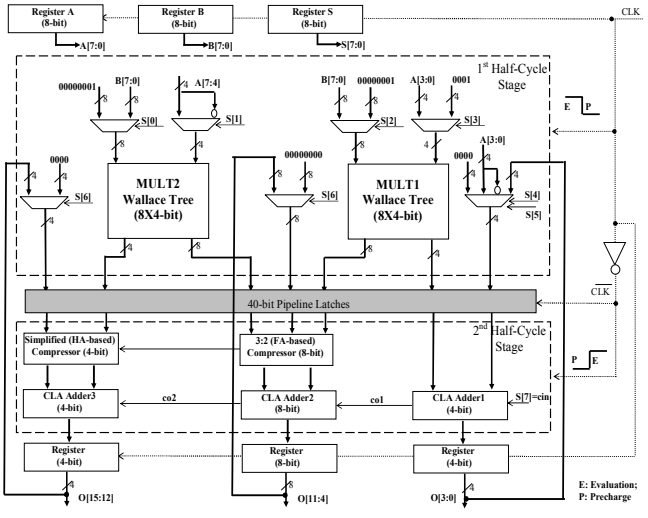


Figure 4: Organization of the pipelined dynamic data path

In order to ensure a correct performance comparison between the static and dynamic data paths, it is necessary that the dynamic implementation utilizes the complete clock period. This was assured by introducing a level of pipelining inverting latches after the multiplier stage. As shown in Fig. 4, this serves to divide the data path into two half-cycle stages, which work using opposite phases of the clock. While the clock is high, the first half-cycle stage evaluates and the second precharges. While the clock is low, the second evaluates and the first precharges. With this approach, the precharge time does not appear in the critical path. The inverting latches hold the result of the first half-cycle while that stage precharges and the next evaluates.

It is worth noting that even though the data path is dynamic, the remainder of the circuitry in the RC is static. Static gates may produce non monotonic output signals; whereas domino gates require inputs that are monotonic during evaluation [4] (only single 0→1 transitions are permitted during the evaluation period). As a consequence, a static-dynamic interface at the input, as well as dynamic-static interface at the output is needed for the dynamic data path to work correctly with the remaining static blocks. This makes it necessary to carefully select the type of registers to be used at the input, output and pipeline stages.

The input registers, used in our dynamic implementation, are based on a special static-dynamic interface latches called Entry Latches (ELATs) [7] used in the Itanium processor. The circuit structure of the generic latch is shown in Fig.5a. This 1-bit latch is similar to the pulsed domino flip-flop and provides a good interface between static and dynamic circuits. The falling inputs must setup before the clock edge while rising inputs can come in slightly later than the clock edge. The output is a monotonically rising signal as is needed by the subsequent dynamic stages inside the data path. Note that this kind of circuit has been also used to implement the pipeline latches.

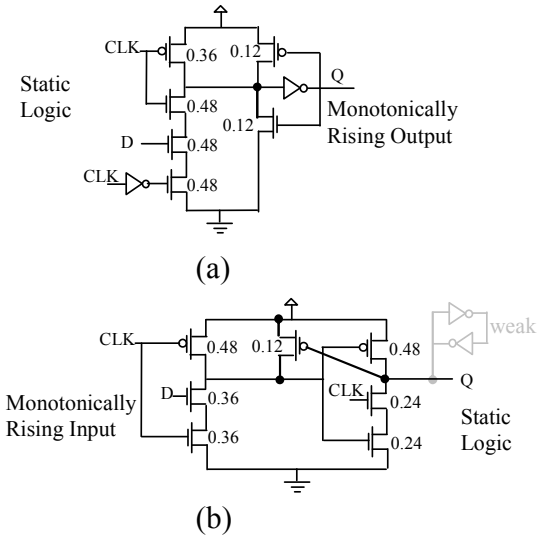


Figure 5: (a) Static to Dynamic Interface [7]
(b) Dynamic to Static Interface [8]

At the output stage, another type of register called pulse-to-level converter was used [8]. The latter converts the output of the domino signals into signals that remain stable until sampled by the static sage. It thus provides the needed dynamic- static interface. Furthermore, as these two types of registers are mutually compatible, accumulation operation which requires link between the output and pipeline latches works perfectly. The circuit diagram for the pulse to level converter is shown in Fig.5b. The dynamic domino-based PE implementation requires careful design of the clock tree. In order to distribute the clock signal to the dynamic domino arithmetic sub-circuits, a two-level clock buffer tree was purposely designed. In order to balance loading to the clock network, the entire data path was divided into four portions such that each portion provides approximately identical loading to the clock network. The logical effort method [4] was used for sizing the inverter chains of the clock buffering.

C. Data Driven Dynamic Logic Implementation

It is well known that dynamic logics are widely used to design high speed data paths due to their intrinsic performance advantages. As a drawback, circuits based on conventional dynamic logic styles suffer from a large amount of power consumption primarily due to the excessive loading on the clock. As demonstrated in [9], the power dissipation owing to the clock distribution network in a dynamic system can range from 20% up to 45% of the overall consumed power.

D3L technique removes the clock distribution system required within conventional dynamic circuits, and utilizes a combination of the input signals (instead of the global clock) to derive the alternate pre-charge and evaluation phases. This provides the advantage of high speed circuit operation like the other dynamic families without the extra cost of power consumption and clock tree design [3, 5, 10].

As with domino implementation, only the arithmetic sub-circuits within the PE are designed using the D3L approach. The multiplexers even in this case are static.

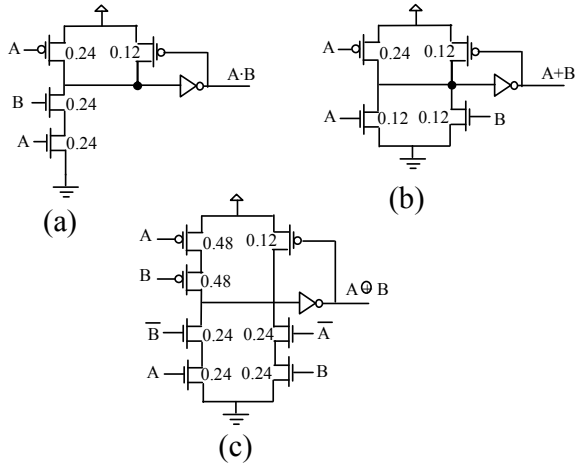


Figure 6: Implementation of (a) AND, (b) OR, (c) XOR gates using data driven dynamic logic

Fig.6 shows the implementation of the D3L basic gates which are inside the arithmetic sub-circuits. A collection of inputs is selected to design the PUN and PDN so as to assure alternate pre-charge and evaluate phases. They must be chosen to satisfy the following conditions: 1) during the precharge phase, the PDN is OFF, the PUPN is certainly turned ON and the output node is charged to *Vdd*; 2) during the evaluation phase, the output node is eventually discharged to 0 by the PDN without any contention with the PUN. Fig.7 shows the design of the implemented D3L full adder. Differently from a nominal D3L-based implementation, in the SUM path, the

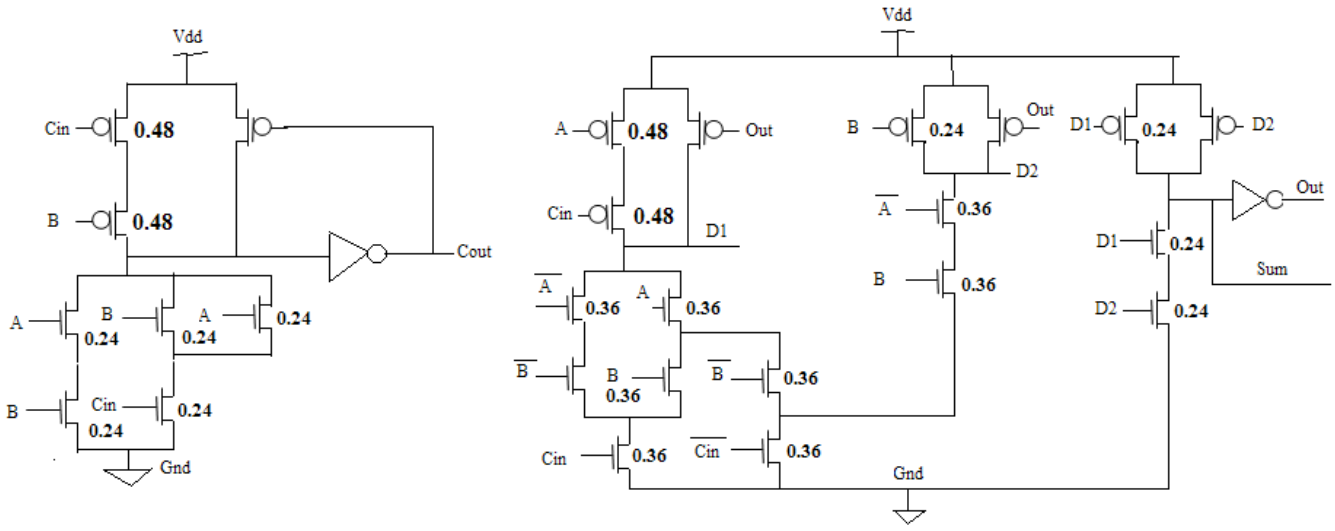


Figure 7: Implementation of Full adder using data driven dynamic logic (D3L)

PUN was split into two parts, thereby reducing capacitance at the output node, and providing for a faster precharge operation. This approach increases the speed at the cost of slightly more power consumption.

The D3L implementation of the PE uses more transistors compared to the corresponding domino based circuit. However, the transistor count is still significantly less than the static implementation.

It is worth pointing out that, similarly to the Domino implementation, the D3L based PE is organized in a pipeline fashion. The input, output and pipeline registers are identical to those used for the domino PE.

IV. RESULTS

Custom design of all three circuits was implemented in ST Microelectronics 90nm 1V CMOS technology, using CADENCE IC Design suite.

After preliminary evaluations using pre-layout simulations, custom layouts were designed. To ensure a fair and correct comparison between the three designs, similar floor planning and routing strategies were adopted for all the circuits. The layout was aimed at achieving maximum possible density, to optimize silicon area efficiency of the data paths. Careful routing strategies were used to ensure minimum interconnect dominance on circuit performance. Fig.14 shows the screen images of all three layouts.

Extensive post layout simulations using the TT 25°C process corner were performed on the data paths to evaluate their performance in terms of speed and power consumption. The Spectre simulator was used to evaluate achieved speed, whereas energy consumption results were obtained using Synopsis Nanosim.

Table 1 shows the obtained post layout results. The data paths were characterized for delay, power and area. As shown from the simulation results, each of the three designs brings with it the features of the used implementation strategy. The static design provides about 23 % and 9.5 % less energy consumption compared to the domino and D3L designs, respectively.

The domino design, exhibits an operating frequency over 1GHZ with 8% and 18% more silicon density compared to the D3L and static counterparts, respectively.

The D3L data path appears ideal for GHz range operations with 10% speed and 15 % energy consumption advantages over the corresponding domino circuit. Note that despite that both dynamic datapaths use more logic gates than the static implementation due to the inserted pipelining registers, they occupy less silicon area. Table 1 shows also a comparison in terms of the Energy-Delay-Area (EDA) product. Analyzing these results clearly proves that among the evaluated implementations the D3L-based PE achievies the best energy-delay-area tradeoff. In fact, it reduces the EDA product of about 20% and 15% with respect to static and dynamic domino implementations, respectively.

Table 1 : post layout results

Data path	**Frequency [GHz]**	**Energy [uW/MHz]**	**Area [mm^2]**	**EDA [pJ*ps* mm^2]**
Static	0.92	**1.58**	0.209	360.27
Domino	1.09	2.04	**0.181**	338.59
D3L	**1.2**	1.74	0.199	**288.43**

CONCLUSION

A high performance power- and area- efficient parallel processing based data path has been proposed. Three variants of the same namely, static, domino and D3L based implementations have been evaluated achieving different area-speed-power tradeoffs. A complete set of data paths is

thus available to be used as processing elements for a novel multimedia oriented reconfigurable architecture [2]. The choice of the implementation strategy to be used can be made depending on the specific need of the targeted applicative domain.

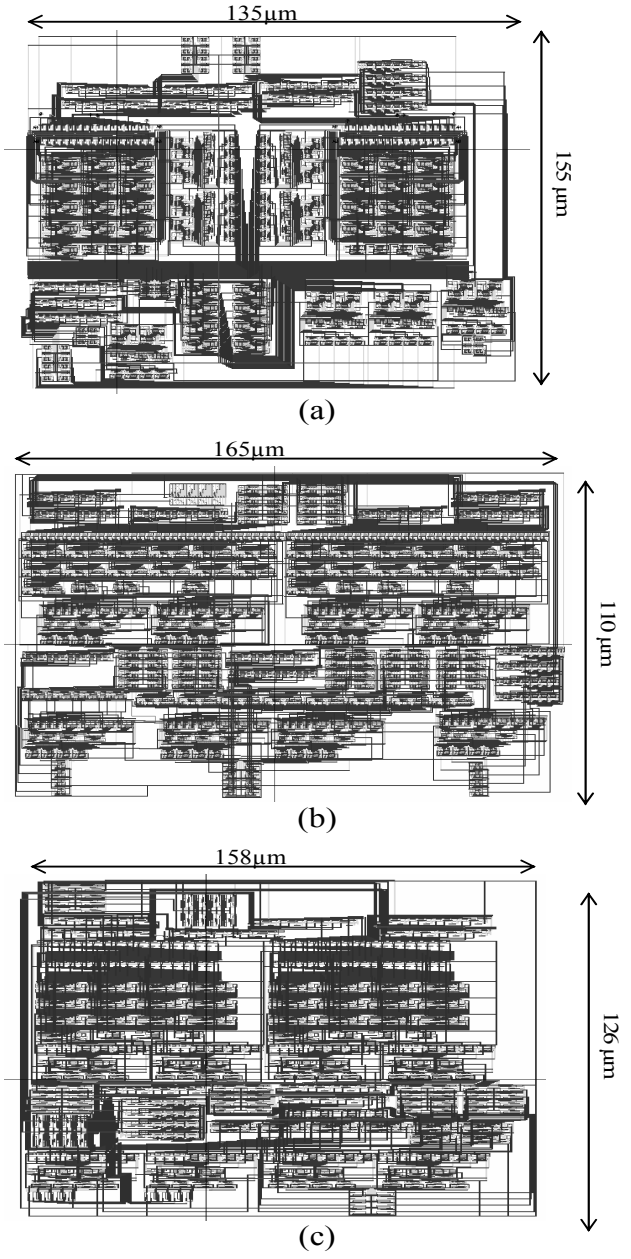


Figure 8: Layouts of the three data path implementations in ST microelectronics 90nm technology: (a) static, (b) domino, (c) D3L.

REFERENCES

[1] Hartenstein, R.: A Decade of Reconfigurable Computing: a Visionary Retrospective. *Proc. of Design, Automation and Test in Europe*, pp. 642-649, 2000.

[2] M. Lanuzza, S. Perri, P. Corsonello, M. Margala "A New Reconfigurable Coarse-Grain Architecture for Multimedia Applications", AHS 2007, Page(s):119-126, 2007

[3] R. Rafati, S. M. Fakhraie, K. C. Smith, "Low-Power Data-Driven Dynamic Logic (D^3L)", Proc. of *IEEE International Symposium on Circuits and Sytems*, ISCAS, Volume 1, Issue , 2000 Page(s):752 - 755 Geneva, Switzerland, 2000.

[4] M. Rabaey, A. Chandrakasan, B. Nikolic "Digital Integrated Circuits– A Design Perspective" Second Edition, Prentice-Hall Editor, 2002.

[5] Rafati, R.; Fakhraie, S.M.; Smith, K.C., " A 16 bit barrel shifter implemented in Data-driven dynamic logic," IEEE transactions on Circuits and Systems, Vol 53, Issue 10, pp-2194-2202, Oct. 2006

[6] C.G. Huang.," Implementation of true single-phase clock D flipflops Electronics Letters, Vol. 30, Issue 17, Page(s):1373 – 1374, 18 Aug 1994

[7] Naffziger, S.D.; Colon-Bonet, G.; Fischer, T.; Riedlinger, R.; Sullivan, T.J.; Grutkowski, T. "The Implementation of the Itanium 2 Microprocessor", IEEE Journal of Solid State Circuits, vol. 37, pp-1448-1460,Volume 37, Issue 11, Nov 2002.

[8] N. Weste, D. Harris "CMOS VLSI Design -A Circuits and Systems Perspective" (3rd Edition), Addison Wesley, 2005.

[9] H. Kawaguchi and T. Sakurai "A reduced clock-swing flip-flop (RCSFF) for 63% power reduction," *IEEE J. Solid-State Circuits*, vol. 33, no. 5, pp. 807–811, May 1998.

[10] Rafati, R.; Charaki, A.Z.; Fakhraie, S.M.; Smith, K.C.," Data-Driven Dynamic Logic versus NP-CMOS Logic, A Comprarison," Proc. of 12th International Conference on Microelectronics 2000, pp-57-60, 2000

Low-Power VLSI Design of LDPC Decoder Using DVFS for AWGN Channels

Weihuang Wang, Gwan Choi
Department of Electrical and Computer Engineering
Texas A&M University, TX 77840
Email: whwang@tamu.edu, gchoi@ece.tamu.edu

Kiran K. Gunnam
Channel Architecture, Storage Peripherals Group,
LSI Corporation, Milpitas, CA 95035
Email: kiran.gunnam@lsi.com

***Abstract*—This paper presents a low-power LDPC decoder design for additive white Gaussian noise (AWGN) channels. The proposed decoding scheme provides constant-time decoding and thus facilitates real-time applications where guaranteed data rate is required. It analyzes each received data frame to estimate the maximum number of necessary iterations for frame convergence. The results are then used to dynamically adjust decoder frequency and switch between multiple-voltage levels; thereby energy use is minimized. It differs from recent publications on speculative LDPC decoding for block-fading channels. Our approach addresses the more difficult problem of decoding requirement prediction for data frames in AWGN channels. It is also directly applicable for fading channels. A decoder architecture utilizing offset min-sum layered decoding algorithm is presented. Up to 30% saving in decoding energy consumption is achieved with negligible coding performance degradation.**

I. Introduction

Low-density parity-check (LDPC) codes are special cases of error correcting codes originally proposed by Gallager [1] in 1960's and rediscovered in late 1990's [2]. LDPC codes have recently gained a significant attention because of their near Shannon-limit performance and high throughput. They have been successfully adopted in next-generation standards, such as *IEEE802.11n*, *IEEE802.16e*, DVB-S2, etc.

LDPC codes are defined by a sparse parity check matrix $H = [Hmn]$ that consists mostly of 0's, as an example shown in Figure 1. The H matrix of (d_c, d_v) regular LDPC code has the following properties. Each column contains a small fixed number d_v of 1's and each row contains a small fixed number $d_c > d_v$ of 1's. The block length of this code is n which is equal to the number of columns in the H matrix. Suppose that the number of data bits before the channel encoding is l, then the number of rows of this H matrix is $m = n - l$. Rate of this code is defined as $l/n = 1 - d_v/d_c$. The code words consist of all one-dimensional row vectors that span the null space of the parity check H matrix. The number for d_v and dc should be no less than 3 and 6 respectively, for good coding performance.

LDPC codes can be equivalently represented in a bi-partite graph, illustrated by Figure 2. The widely used message-passing algorithms exchange information between the variable-nodes and check-nodes in an iterative fashion. In practice, the LDPC decoder is typically set to run for data convergence until a prescribed maximum number of iterations

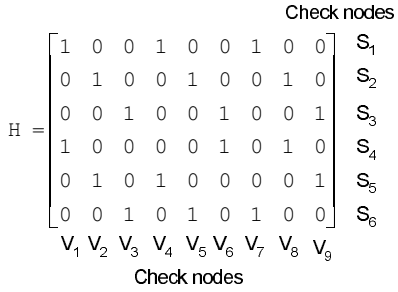


Fig. 1. An example H matrix of $d_c = 3$ and $d_v = 2$.

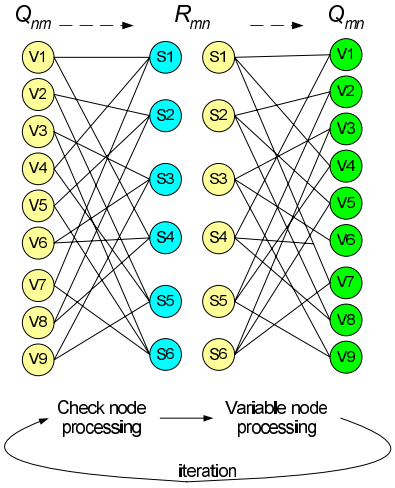


Fig. 2. Bipartite graph representation and two phase message passing decoding algorithm.

(e.g. 20) depending on the code rate [3]–[6]. However, the actual number of decoding iterations varies from frame to frame. In the case that channel data comes in constant-time interval, a conventional decoder has to be configured to accommodate the worst case scenario. As a result, the decoder often remains idle since for most frames, the decoding process ends far earlier than the maximum number of iterations. Thus it is not power efficient. In the decoders proposed in [7]–[10] based on an on-the-fly computation paradigm, optimized dataflow graphs are introduced to reduce the logic and internal memory requirements of the LDPC decoder and at the same time the decoder's parallelization is tailored to average number

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.68

of decoding iterations for the target frame error rate for the operating SNR region. These decoders buffer statistically for different parallelization based on average number of decoding iterations while ensuring the performance similar to that of a fixed iteration decoder configured for maximum number of iterations [7]–[10]. These decoders are almost fully utilized and runs at the maximum frequency. The proposed paper improves the system efficiency in a different approach by adaptively adjusting the frequency and voltage of the decoder to meet the required iterations for each incoming frame.

There have been researches on early termination of frame that can not be decoded even if the maximum iterations are applied [5], [6]. In both of papers, early termination of the iterative process is determined by checking the messages during the decoding. Their attempts are to dynamically switch off the hardware when no additional iteration will amount to improvement in decoding performance. Such architectures yield unpredictable frame-completion time which makes interfacing with the application modules rather difficult. Wang, *et al.* address the problem of dynamic voltage and frequency scaling (DVFS) for block fading channels in [11]. The presented approach predicts the required decoding effort by evaluating number of checks in error for each frame. Such approach will not work for AWGN channels where correlation between initial check error and number of decoding iterations is insufficient. This paper describes a more comprehensive design that accommodate both fading channels and the more difficult case of AWGN channels.

The rest of the paper is organized as following: Section II presents the LDPC decoding algorithm based on offset min-sum and layered decoding. The low-power adaptive decoder architecture is proposed in Section III. Implementation results as well as discussion is given in Section IV. Section V concludes the paper.

II. LDPC LAYERED DECODING BASED ON OFFSET MIN-SUM ALGORITHM

The array LDPC codes [12] are specified by three parameters: a prime number p and two integers k and j such that $j < p$ and $k < p$. It is given by Equation (1):

$$H = \begin{bmatrix} I & I & I & \cdots & I \\ I & \alpha & \alpha^2 & ... & \alpha^{k-1} \\ I & \alpha^2 & \alpha^4 & ... & \alpha^{2(k-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ I & \alpha^{j-1} & \alpha^{(j-1)2} & \cdots & \alpha^{(j-1)(k-1)} \end{bmatrix} \quad (1)$$

where I is the $p \times p$ identity matrix, and α is a $p \times p$ permutation matrix representing a single left or right cyclic shift of I. Power of α in H denotes multiple cyclic shifts, with the number of shifts given by the value of the exponent.

Assume binary phase shift keying (BPSK) modulation (a 1 is mapped to -1 and a 0 is mapped to 1) over an additive white Gaussian noise (AWGN) channel. The received values y_n are Gaussian with mean $x_n = 1$ and variance δ^2. The iterative two-phase message-passing (TPMP) algorithm, also known as belief-propagation (BP) algorithm [13], [14] is computed in two phases. One is a check node processing and the other is variable node processing. In the check node processing, each row of the parity matrix is checked to verify that parity check constraints are satisfied. In the variable node processing the probability will be updated by summing up the other probabilities from the rest of the rows and the a priori probabilities from the channel output. The message-passing algorithm can be simplified to the belief-propagation based algorithm (also called Min-Sum algorithm) [15]. While greatly reducing the decoding complexity in implementation, the Min-Sum degrades the coding performance. The improved BP based algorithm, Normalized-Min-Sum and Offset-Min-Sum eliminates this performance degradation.

Following the same notation in [8], the check node processing can be expressed as:

$$R_{mn}^{(i)} = \delta_{mn}^{(i)} \max\left(\kappa_{mn}^{(i)} - \beta, 0\right), \quad (2)$$

$$\kappa_{mn}^{(i)} = \left|R_{mn}^{(i)}\right| = \min_{n' \in \mathrm{N}(m) \backslash n} \left|Q_{n'm}^{(i-1)}\right|, \quad (3)$$

where $Q_{nm}^{(i)}$ is the message from variable node n to check node m, $R_{mn}^{(i)}$ is the message from check node m to variable node n, and superscript i denotes the i^{th} iteration. $M(n)$ is the set of the neighboring check nodes for variable node n, and $N(m)$ is the set of the neighboring variable nodes for check node m, β is a positive constant and depends on the code parameters [15]. The sign of check-node message $R_{mn}^{(i)}$ is defined as

$$\delta_{mn}^{(i)} = \left(\prod_{n' \in \mathrm{N}(m) \backslash n} \mathrm{sgn}\left(Q_{n'm}^{(i-1)}\right)\right). \quad (4)$$

In the vairalbe node processing,

$$Q_n = L_n^{(0)} + \sum_{m \in M(n) \backslash m} R_{mn}^{(i)} \quad (5)$$

where the log-likelihood ratio of bit n is $L_n^{(0)} = y_n$. For final decoding

$$P_n = L_n^{(0)} + \sum_{m \in M(n)} R_{mn}^{(i)} \quad (6)$$

A hard decision is taken by setting $\hat{x}_n = 0$ if $P_n(x_n) \geq 0$, and $\hat{x}_n = 1$ if $P_n(x_n) \leq 0$. If $\hat{x}H^T = 0$, the decoding process is finished with $\hat{x}_n$ as the decoder output; otherwise, repeat processing of Equation (2)-Equation (6). If the decoding process does not end within predefined maximum number of iterations, it_{max}, stop and output an error message flag and proceed to the decoding of the next data frame.

Mansour, *et al.* introduces the concept of turbo decoding message passing (also called as layered decoding) for their AA-LDPC codes using BCJR [4], which is able to reduce the number of iterations required by up to 50% without performance degradation when compared to the standard message passing algorithm. Contrast to two phase message passing

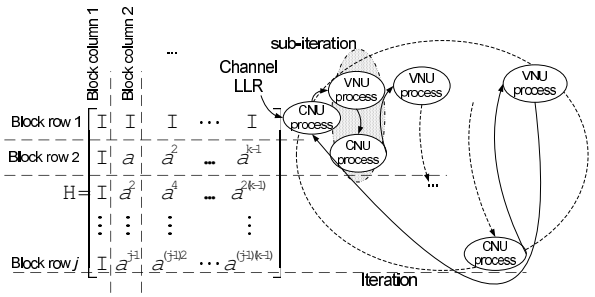


Fig. 3. In layered decoding, the H matrix is viewed as concatenation of d_v sub-codes and each block row (layer) is updated individually.

algorithm, where all check-nodes are updated simultaneously in each iteration, layered decoding view the H matrix as a concatenation of $j = d_v$ sub-codes, as shown in Figure 3. The H matrix is divided into different block-rows and block-columns. After the check-node processing of one layer, the updated messages are immediately used to calculate the variable-node message, whose results are then applied to next layer of sub-code. Each iteration in the layered decoding algorithm is composed of j sub-iterations. The processing of one block-row is called a sub-iteration and each iteration is composed of $j = d_v$ sub-iterations. Mathematically, the layered decoding algorithm can be described as in Algorithm 1:

$\vec{R}_{l,n}^{(0)} = 0,\ \forall l \in [1, d_v], n \in [1, d_c]$
$\vec{P}_n = \vec{L}_n^{(0)},\ \forall n \in [1, d_c]$
foreach $i = 1,\ 2,\ \cdots,\ i_{tmax}$ **do**
  foreach $l = 1,\ 2,\ \cdots,\ d_v$ **do**
    foreach $n = 1,\ 2,\ \cdots,\ d_c$ **do**

$$\left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} = \left[\vec{P}_n\right]^{S(l,n)} - \vec{R}_{l,n}^{(i-1)} \quad (7)$$

$$\vec{R}_{l,n}^{(i)} = f\left(\left[\vec{Q}_{l,n'}^{(i)}\right]^{S(l,n')}\right), \forall n' \in [1, d_c] \quad (8)$$

$$\left[\vec{P}_n\right]^{S(l,n)} = \left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} + \vec{R}_{l,n}^{(i)} \quad (9)$$

Algorithm 1: Layered decoding for array LDPC.

III. Low-power adaptive LDPC decoder architecture

This section presents the adaptive decoder architecture for additive white Gaussian noise (AWGN) channel. Decoding effort is estimated before the decoding process of every frame in block-fading channels [11], [16]. Decoding energy is saved via truncating the distribution of decoding effort and applying dynamic voltage and frequency scaling (DVFS) technique. This approach, however, does not work for LDPC decoding in AWGN channels.

While number of check errors for the coming channel data still partially represents the degree of noise impairment, and indicate effort required for decoding, the correlation between number of check errors and number of decoding iteration though, becomes much less. However, this correlation can be

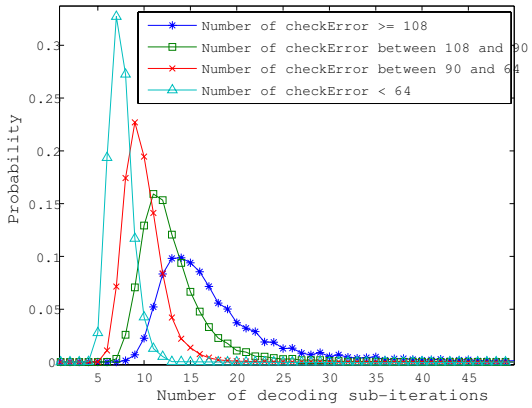


Fig. 4. Distribution of decoding sub-iteration for different number of check errors after three sub-iterations with channel Eb/No at 3.7dB.

$\vec{R}_{l,n}^{(0)} = 0,\ \forall l \in [1, d_v], n \in [1, d_c]$
$\vec{P}_n = \vec{L}_n^{(0)},\ \forall n \in [1, d_c]$
foreach $l = 1,\ 2,\ 3$ **do**
  foreach $n = 1,\ 2,\ \cdots,\ d_c$ **do**

$$\left[\vec{Q}_{l,n}^{(1)}\right]^{S(l,n)} = \left[\vec{P}_n\right]^{S(l,n)}$$

$$\vec{R}_{l,n}^{(1)} = f\left(\left[\vec{Q}_{l,n'}^{(1)}\right]^{S(l,n')}\right), \forall n' \in [1, d_c]$$

$$\left[\vec{P}_n\right]^{S(l,n)} = \left[\vec{Q}_{l,n}^{(1)}\right]^{S(l,n)} + \vec{R}_{l,n}^{(1)}$$

$\hat{x_n}$ = hard decision of P_n
if $\hat{x}H^T \in (cErrThre_1, +\infty)$ **then** $i_{sub,max} = C_1$
else if $\hat{x}H^T \in (cErrThre_2, cErrThre_1]$ **then**
  $i_{sub,max} = C_2$
else if $\hat{x}H^T \in (cErrThre_3, cErrThre_2]$ **then**
  $i_{sub,max} = C_3$
else if $\hat{x}H^T \in (0, cErrThre_3]$ **then**
  $i_{sub,max} = C_4$
foreach $i_{sub} = 4,\ \cdots,\ i_{sub,max}$ **do**
  $i = \lceil i_{sub}/d_v \rceil$
  $l = (i_{sub} - 1) \bmod (d_v) + 1$
  foreach $n = 1,\ 2,\ \cdots,\ d_c$ **do**

$$\left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} = \left[\vec{P}_n\right]^{S(l,n)} - \vec{R}_{l,n}^{(i-1)}$$

$$\vec{R}_{l,n}^{(i)} = f\left(\left[\vec{Q}_{l,n'}^{(i)}\right]^{S(l,n')}\right), \forall n' \in [1, d_c]$$

$$\left[\vec{P}_n\right]^{S(l,n)} = \left[\vec{Q}_{l,n}^{(i)}\right]^{S(l,n)} + \vec{R}_{l,n}^{(i)}$$

Algorithm 2: Adaptive LDPC decoding for AWGN channel.

recovered after several initial decoding iterations. Analysis in this section has been carried out based on a sample array LDPC code, with parameters $d_v = 5$, $d_c = 25$, and $p = 67$. And layered decoding based on offset min-sum (OMS) algorithm is used for decoding. Similar analysis can be extended to other array LDPC codes. It is shown in Figure 4 that after three sub-iterations, the number of check errors remaining is

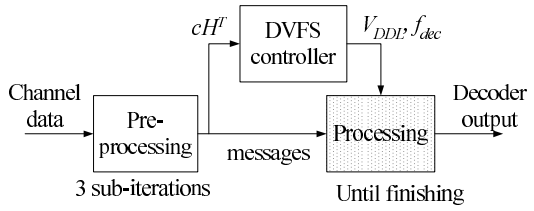


Fig. 5. Block diagram of the proposed adaptive decoding for AWGN channel.

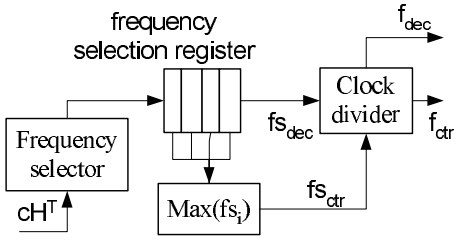


Fig. 6. DVFS controller for adaptive LDPC decoding.

correlated with the total number of decoding iterations. As such, the adaptive decoding for AWGN channel is proposed as in Algorithm 2.

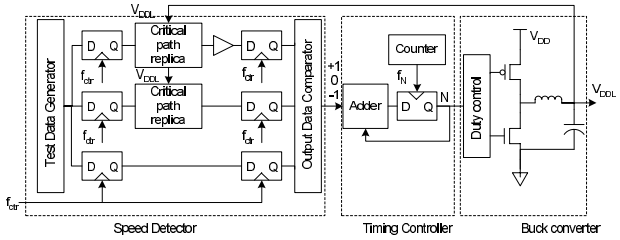


Fig. 7. Variable supply voltage scheme [17].

There are two sets of parameters that need to be determined experimentally for each specific application: the check-error threshold values $cErrThre_{[1:3]}$ and the maximum number of decoding sub-iterations $C_{[1:4]}$. Note that this algorithm can be easily extended to fading channels as well.

An efficient decoder architecture is proposed based on the layered decoding architecture [7]. Block diagram of the decoder is shown in Figure 5. There are three main blocks: the pre-processing unit, the processing unit and the DVFS controller. For each incoming data frame, the first three decoding sub-iterations take place in the pre-processing unit. The check-node messages and P sum messages are stored in frame buffers and the value of cH^T will be sent to the DVFS controller for voltage and frequency adjustment. The processing unit takes over the unfinished decoding process by reading messages from the frame buffers. It operates in the variable voltage V_{DDL} domain. Decoding power is reduced since majority of the decoding process is done with the processing unit.

Diagram of the DVFS controller is presented in Figure 6. Number of check errors is calculated after three sub-iterations of each frame. Since cH^T is needed by the decoder for decoding termination decision and it can be reused, this calculation does not impose any additional hardware resource. Because level of the voltage supply can not be changed instantly, frame buffer is required between the pre-processing unit and the processing unit, the frequency-selection register stores decoding iteration information for the corresponding data frames. The clock divider divides the fast system clock into slower clock signals according to the frequency selection register. Clock divider is preferred over other designs such as phase-loop locker (PLL) in [18], because it provides reasonable frequency resolution for the decoding policy and capability to change immediately. f_{dec} clocks the decoder for current frame, and f_{ctr} is sent to the variable voltage scheme, as shown in 7. f_{ctr} is conservatively generated as the fastest clock such that the voltage supply will be within safe region for operation. The variable voltage scheme is studied in depth by T. Kuroda, *et al.*, and the readers are referred to [17] for the details.

Figure 8 shows the adaptive LDPC decoder architecture in detail. The CNUs in the figure take the d_c variable-node message associated with each check-node serially, and computes the compressed check-node message in the form of $M1$, $-M1$, $M2$ or $-M2$, and index of $M1$ [7]–[9], in which $M1$ is the least magnitude of all d_c variable-node messages and $M2$ is the second least magnitude. These compressed check-node messages are called Final States (FS) and they are stored in FS buffers. Check-node messages to each associated variable-node are sent out serially again and they are selected from $M1$, $-M1$ and $M2$ or $-M2$ by index and sign comparison, where the signs of R messages are stored in sign FIFO. This approach reduces check-node message memory requirement by 50% to 80% depending on the code parameters. As soon as check-node message to the first connected variable-node are ready, the corresponding P sum message can be computed. Therefor, the Q message is ready for check-node message processing of next sub-iteration. As such, each CNU operates on two layers of the H matrix simultaneously: selecting check-node message for one layer and computing FS for the next layer.

The pre-processing unit operates in the nominal V_{DD} voltage domain. In the first iteration, Q^{shift} equals channel LLR or P message subtracting R_{old}, in which R_{old} is check node message from last iteration, i.e. $R_{l,n}^{i-1}$ in Equation (7). Since both FS register and sign FIFO are reset to zero when decoding starts, and remains zero during the first decoding iteration, the subtraction is not necessary for the pre-processing unit. As such, channel data will go directly to the CNU array through the shifter. Only three sub-iterations will take place in the pre-processing unit, less CNU units can be instantiated than the processing unit, to match their throughput. Achieving $p \times p$ cyclic shifts by $M \times M$ shifter, where $M < p$, is introduced in [8]. P messages and check-node messages are passed from pre-processing unit to processing unit through the frame buffer. The frame buffer size K is determined by time response of variable voltage supply V_{ddl} as well as expected decoding time, *i.e.* decoder throughput. As will be presented later, buffer size of 4 frames is typically needed. The overhead is buffer of size 3 frame since a buffer size of 1 frame is intrinsic for the decoder architecture [7]. Considering the memory-saving nature of the decoder architecture, this overhead is very small.

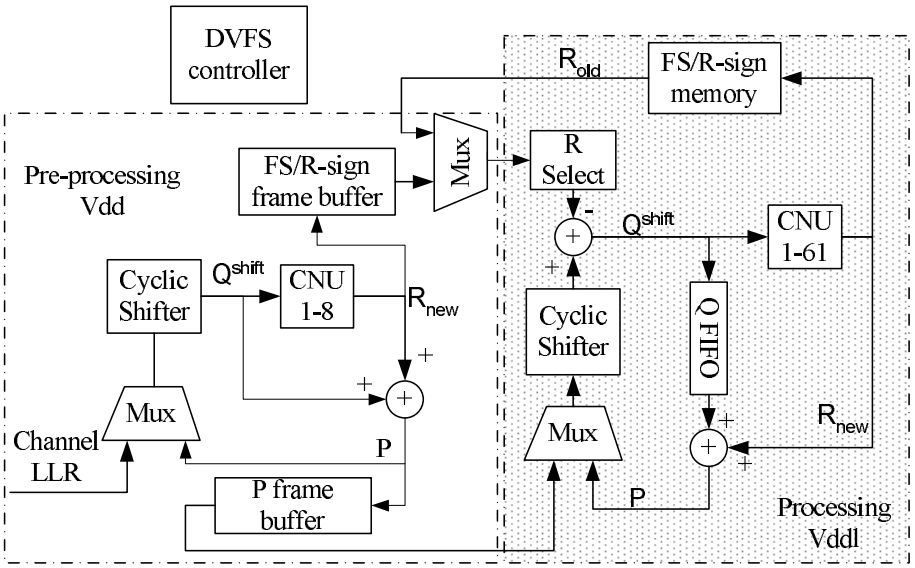


Fig. 8. Proposed decoder architecture for AWGN channel.

Note that other than the frame buffer, the pre-processing unit should not be considered as hardware overhead of the adaptive decoder architecture because it is an inherit part of the decoder and it contributes directly to the final decoder throughput.

The processing unit takes P sum messages and check-node messages from the pre-processing unit and continue decoding. The difference from pre-processing unit lies in that Q messages instead of P message are stored. Computation of Q messages are carried exactly as in Equation (7).

IV. Results and Discussion

The design of voltage-scaling controller has been simulated using $TSMC0.13\mu m$ technology. Critical path of the decoder is extracted from Synopsys design compiler. With $1.5V$ voltage supply, the decoder can be clocked as fast as $175MHz$, as shown in Figure 9. Extra 5% timing margin has been added to the critical path replica in the controller to accommodate variations.

Area of the DVFS controller and variable supply voltage scheme is $0.1 \times 0.13mm^2$, about 1% of the $1.1mm^2$ whole decoder and the power overhead is less than $10mW$, which mainly comes from the buck converter. This power consumption is small comparing to the total power dissipation of the decoder at around $200mW$. The variable voltage scheme is capable of adjusting the V_{DDL} voltage at a speed of $20mV/\mu s$. Similar results has been reported in [17]. For LDPC decoding at Eb/No of $3.7dB$, the maximum number of decoding sub-iterations $C_{[1:4]}$ is choose to be 48, 32, 24 and 19, respectively and the corresponding decoder frequency is chosen to be $165MHz$, $110MHz$, $82.5MHz$ and $66MHz$, which can be easily derived from a $330MHz$ system clock. As such, it will yield a nearly equal maximum decoding time for frames in each category. The voltage supply varies from $1.5V$ to $1V$ within this frequency range. It takes about $25\mu s$ to

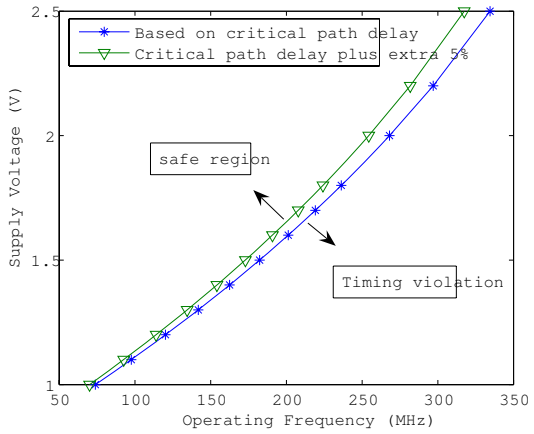


Fig. 9. Voltage supply requirement for different operating frequencies of the processing unit.

scale the voltage up by $0.5V$. In the case of $d_c = 25$, $d_v = 5$ and $p = 67$, the voltage controller is able to respond to as much as $200Mbps$ throughput with a frame-buffer size of 4.

The proposed adaptive decoder architecture saves significant energy consumption. Simulation results is shown in Figure 10, the amount of saving is in comparison with the decoder architecture but without voltage and frequency adjustment. At relatively low SNR level, with Eb/No at $3.4dB$ for example, up to 30% percent saving in energy consumption is achieved. Note that power consumption is estimated assuming 80% dynamic power and 20% static power consumption. Energy consumption is lowered at hight SNR level. At Eb/No of 4.0dB, the saving, though decreases to 21%, is still significant. This decreasing occurs because the reduced average number of decoding iterations. Since the first three sub-iteration takes place in the fixed V_{dd} voltage domain, as SNR increases,

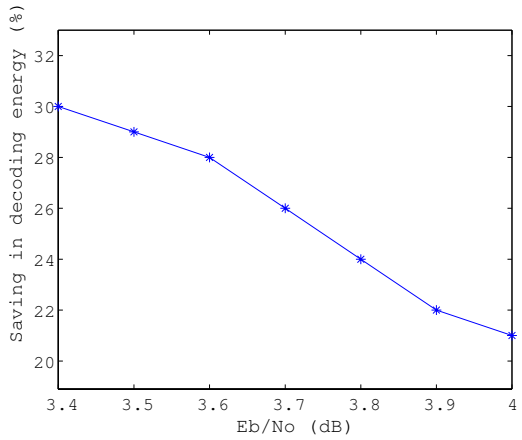


Fig. 10. Saving in decoding energy at different SNR levels.

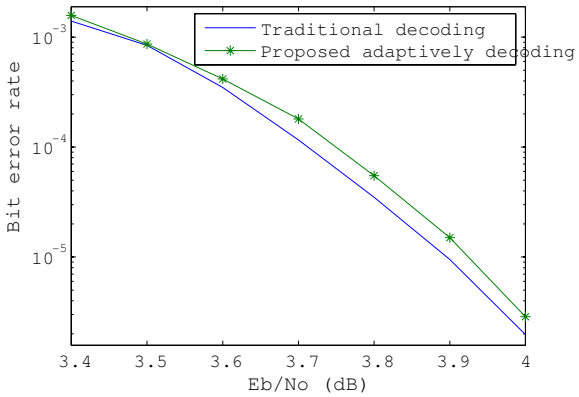


Fig. 11. BER comparison for traditional decoding and proposed adaptive decoding.

decoding energy consumption of the pre-processing block rises relative to the processing unit. Figure 11 shows that the proposed adaptive decoding architecture performs closely with conventional decoding schemes in terms of code performance. As seen, less than $0.05dB$ BER degradation is observed.

V. Conclusion

We have presented a low-power LDPC decoder design for additive white Gaussian noise (AWGN) channels. Decoding effort for each channel frame is estimated in the early stage of the decoding process. The proposed approach solves the difficult problem of decoding prediction for AWGN channels, and it is also applicable for fading channels. Power overhead of the adaptive decoder control unit mainly stems from the variable voltage supply scheme, and it is small compared with power saved. Up to 30% energy saving in decoding process is achieved with less than 0.05dB coding performance degradation in Eb/No.

References

[1] R. Gallager, "Low-density parity-check codes," *IRE Trans. on Inform. Theory*, vol. 8, pp. 21–28, Jan. 1962.

[2] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check code," *Elec. Letters*, vol. 332, pp. 1645 – 1646, Aug. 1996.

[3] M. Mansour and N. Shanbhag, "High-throughput ldpc decoders," *IEEE Trans. on Very Large Scale Integrated (VLSI) System*, vol. 11, no. 6, pp. 976 – 996, Dec. 2003.

[4] M. Mansour and N. Shanbhag, "A 640-mb/s 2048-bit programmable ldpc decoder chip," *IEEE J. of Solid-State Circuits*, vol. 4, no. 3, pp. 684 – 698, Mar. 2006.

[5] F. Kienle and N. When, "Low complexity stopping criterion for ldpc code decoders," *IEEE 61st Vehicular Technology Conference*, vol. 1, pp. 606 – 609, May 2005.

[6] G. Glikiotis and V. Paliouras, "A low-power termination criterion for iterative ldpc code decoder," *IEEE Workshop on Signal Processing Systems Design and Implementation*, pp. 122 – 127, Nov. 2005.

[7] K. Gunnam, G. Choi, W. Wang, E. Kim, and M. Yeary, "Decoding of quasi-cyclic LDPC codes using on-the-fly computation," *Fortieth Asilomar Conference on Signals, Systems and Computers, 2006. ACSSC 06*, pp. 1192 – 1199, Oct. 2006.

[8] K. Gunnam, W. Wang, G. Choi, and M. Yeary, "Multi-rate layered decoder architecture for block LDPC codes of the IEEE 802.11n wireless standard," *IEEE International Symposium on Circuits and Systems*, pp. 1645–1648, May 2007.

[9] K. Gunnam, G. Choi, and M. B. Yeary, "A parallel layered decoder architecture for array LDPC codes," *Proceedings of the 20th International Conference on VLSI Design*, pp. 738–743, Jan. 2007.

[10] K. Gunnam, G. Choi, M. B. Yeary, S. Yang, and Y. Lee, "Next gneration iterative LDPC solutions for magnetic recording storage," *Forty-second Asilomar Conference on Signals, Systems and Computers, 2008*, Oct. 2008.

[11] W. Wang and G. Choi, "Minimum-energy LDPC decoder for real-time mobile application," *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp. 1–6, Apr. 2007.

[12] A. Dholakia and S. Olcer, "Rate-compatible low-density parity-check codes for digital subscriber lines," *2004 IEEE International Conference on Communications*, vol. 1, pp. 415 – 419, Jun. 2004.

[13] T. J. Richarson, M. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. on Information Theory*, vol. 47, pp. 619 – 637, Feb. 2001.

[14] F. Kschischang, B. Frey, and H. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. on Information Theory*, vol. 47, pp. 498 – 519, Feb. 2001.

[15] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density-parity-check codes," *IEEE Trans. on Communications*, vol. COM-50, pp. 406 – 414, Mar. 2002.

[16] W. Wang and G. Choi, "Speculative energy scheduling for LDPC decoding," *Proceedings of the 8th International Symposium on Quality Electronic Design*, pp. 79–84, 2007.

[17] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, and *et al.*, "Variable supply-voltage scheme for low-power high-speed CMOS digital design," *IEEE J. of Solid-State Circuit*, vol. 33, no. 3, pp. 454 – 462, Mar. 1998.

[18] P. Macken, M. Degrauwe, M. van Paemel, and H. Oguey, "A voltage reduction technique for digital systems," *1990 IEEE International Solid-State Circuits Conference, Digest of Technical Papers. 37th ISSCC.*, pp. 238 – 239, Feb. 1990.

2009 22nd International Conference on VLSI Design

Environment and Process Adaptive Low Power Wireless Baseband Signal Processing Using Dual Real-Time Feedback

Muhammad M. Nisar, Abhijit Chatterjee
School of Electrical and computer Engineering
Georgia Institute of Technology, Atlanta, GA, USA
{mnisar, chat@ece.gatech.edu}

Abstract

As technology scales below the 45nm CMOS technology node, RF front ends and baseband processors will need to be aggressively overdesigned to work reliably under worst case channel (environment) conditions as well as worst case manufacturing variations. In this paper, a new dual feedback based design approach is proposed that allows the baseband unit of a wireless OFDM system to adapt dynamically to channel conditions as well as manufacturing process variations. Two nested feedback control loops are used. The first allows the baseband SNR to increase when channel conditions are good and vice versa by modulating system wordlength. The second modulates the system supply voltage in response to the resulting changing wordlength values. Both feedback loops are designed to allow the processor to operate at the minimum power consumption possible without exceeding a specified overall bit error rate across all channel noise and process variability conditions.

1. Introduction

Power consumption and process variations are major concerns in highly scaled and functionally complex devices. In wireless baseband receivers, the noise performance of specific signal processing algorithms for signal demodulation and symbol decoding can be traded off for power under good channel conditions. In operating conditions where the worst case channel is seen infrequently, significant power can be saved by reducing the performance of the baseband signal processing algorithms (trade off performance for power) when channel conditions are not worst-case. The amount of power that can be saved depends however, on the speed of the underlying logic circuitry, i.e. the process parameters corresponding to the manufactured unit. Hence, for minimum power operation, knowledge of both channel conditions as well as logic speed (process parameters) is necessary. We propose a dual feedback mechanism that tailors the operation of each device to changing channel conditions (*dynamic*) and its manufacturing process parameters (device-specific and *static*) for minimum power operation.

This work was supported in part by NSF Grant CCR-0635016 and by GSRC MARCO under award 2003-DT-660.

2. Relationship to Prior Work

Voltage scaling is a well known and very powerful power savings technique because of the quadratic relationship of voltage with power [1]. Wordlength optimization is another important technique for power savings. In prior research, simulation based techniques [2]-[4] have been proposed to find the optimal wordlength for digital signal processing applications. Such algorithms optimize the wordlength according to predetermined system-level performance metrics. Often, the resulting digital circuits are implemented with large wordlength values. A method for tuning the wordlength of digital baseband OFDM signal processing algorithms was proposed in [5]. This allows dynamic adjustment of the wordlengths of digital filtering and FFT operations driven by continuous monitoring of the error vector magnitude (EVM) of the demodulated signal. The method relies on the use of *gated clocks* for wordlength adaptation, which in turn requires significant modifications in hardware implementation.

In prior work [6], a test enabled power savings methodology was presented that trades off power for noise (without compromising overall bit-error rate) in the baseband system under good operating conditions by simultaneously modulating supply voltage (V_{dd}) and wordlength (W). *Signal scaling* [6] is used for system wordlength tuning. The methodology is process tolerant and tunes V_{dd} and W of the DSP blocks according to a minimum *power locus*. The "power locus" is stored in the system as a lookup table, with appropriate V_{dd} and W values for different channel conditions. The channel quality is determined in real time by calculating the EVM of the received symbols. A set of loci are stored in the system representing different intra-die process variations. Although, the scheme results in significant power savings, it requires significant calibration and use of on-chip timing assessment mechanisms. The calibration mechanisms must be appropriately guardbanded to allow margin for calibration errors, resulting in loss of performance and/or power efficiency. In this paper, we present a completely dynamic process-tolerant low power

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.90

adaptation scheme for baseband OFDM processors. The proposed scheme uses two nested control loops for dynamic adjustment of supply voltages and wordlengths of the baseband receiver modules. The outer feedback control loop that monitors system EVM is used to adjust the wordlengths of individual DSP blocks. The inner control loop allows the supply voltage to drop until errors occur in the most significant bits (MSBs) corresponding to the current wordlength value. These errors are then compensated accurately using checksum codes applied to the respective DSP functions (filters, FFT, etc). The inner feedback control loop is designed to control the supply voltage in such a way using a PID controller, that the MSB error rate is always below a prescribed limit. This ensures that the impact of periodic errors on the EVM of the decoded symbols is minimized. The use of two feedback control loops based on two different signal quality metrics (EVM and error rate) ensures that no pre-calibrated lookup tables are needed in the system. Moreover, the use of two separate signal quality metrics to set wordlength and supply voltage values makes this system independent of intra-die process variations [7] effects and allows the control mechanisms to settle at the lowest power operating point for any channel condition automatically.

The rest of the paper is organized as follows. An overview of the dual nested architecture is given in Section 3. Section 4 summarizes the EVM based feedback control presented in prior work. Guided probabilistic architecture (GPC) that enables low power operation of digital filters is also explained in the same section. In Section 5 the two nested loop control for dynamic adjustment of module wordlengths and supply voltages is explained in detail. This is followed by experimental results in Section 6.

3. Proposed Methodology: Rationale and Overview

In wireless communication systems, the wordlength of the baseband signal processing system is determined in such a way that the system bit error rate (BER) is less than the maximum allowed BER value for the wireless communication protocol being used for communication. Note that lower wordlength results in lower signal SNR and thereby a higher bit error probability. The selected wordlength corresponds to the *worst* channel quality that the highest communication data rate can sustain without exceeding a prescribed bit error rate limit (typically between 10e-4 to 10e-5 for many wireless communication protocols). Given the fact that most of the time a mobile device does not operate under worst case channel conditions, the effective wordlength and corresponding module voltage values can be reduced when the channel is not worst-case, saving power while keeping the system BER within quality of service (QoS) requirements.

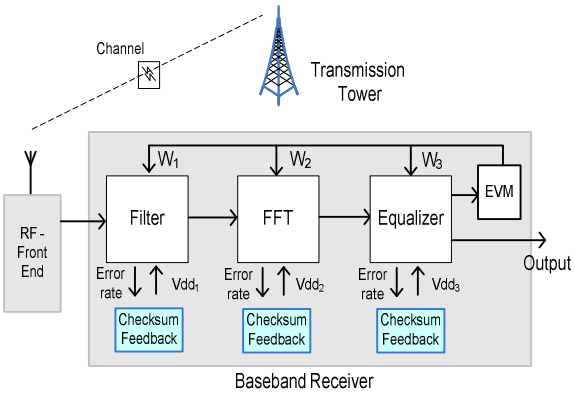


Figure 1: Proposed Architecture

The proposed dual-feedback control architecture is shown in Figure 1 without block error control for simplicity. The EVM value is computed across several frames of data (about 1000 received symbols) in real-time, is updated dynamically and has latency in the order of 10s of milliseconds. The EVM value is used as input to an "outer loop" controller (not shown in Figure 1) that determines the wordlength of the processor. The *EVM* adaptation metric represents the cumulative sum of the quality of transmission, the channel quality degradation and the quality of signal reception. It is shown in previous work that EVM has a strong correlation with BER [5][6].

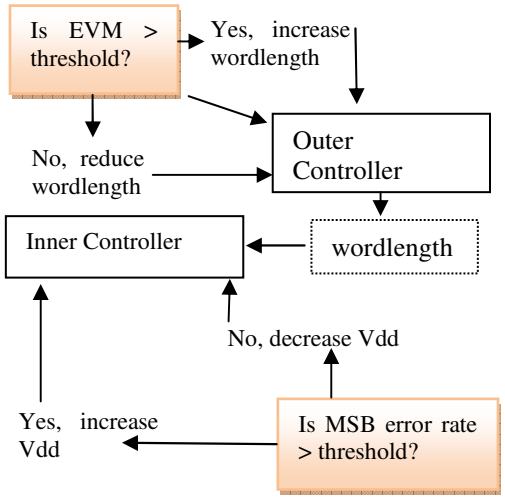


Figure 2: Dual loop control strategy

As the wordlength changes, an "inner control loop" adapts the supply voltage to the current selected wordlength. This is done by decreasing the supply voltage for reduced wordlength (and vice versa) until errors occur in the most significant bits of computation

(MSB errors).These errors are then compensated accurately using checkum codes applied to the underlying DSP algorithms. To allow computation to proceed without loss of throughput, some inaccuracy in the compensation process is allowed. Since a higher error rate has an adverse effect on EVM, the inner loop control mechanism is designed to always maintain a low predetermined MSB error rate. In this manner, the "outer" and "inner" control loops interact with each other to always allow the lowest power operation for any channel condition. The use of two nested feedback loops to adjust V_{dd} and W of the individual blocks at run-time, without any pre-calibrated lookup tables makes this scheme independent of process variations as well. Figure 2 describes the proposed control strategy.

4. Key Concepts

In the following, we first describe how the effective wordlength is dynamically modulated using a concept called signal scaling. Subsequently we review the checksum based error control methodology used in this work. Finally we show how the two can be combined via dual-feedback control to build low power process tolerant wireless baseband units.

4.1 Input Signal/Voltage scaling

In prior work, a signal is degraded by scaling down the input (dropping LSB bits) and correspondingly adjusting the supply voltage (Figure 3). The approach exploits the fact that the critical circuit path lengths of the underlying arithmetic units (multipliers and adders) are reduced by input data scaling. This reduction in circuit critical path length allows the correct operation of the arithmetic units at lower supply voltage without incurring additional bit errors. Since data scaling causes LSB bits drop, it results in graceful system-level performance degradation. This dynamic wordlength/supply voltage adjustment is done to save power while keeping the adaptation metric of the demodulated signal below a specified upper limit.

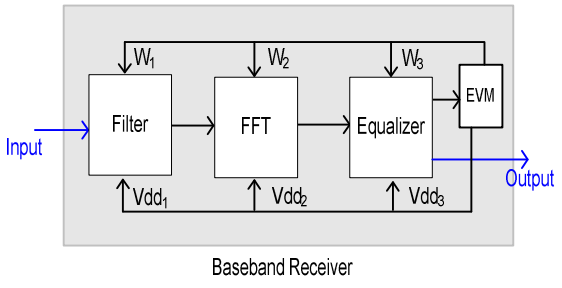


Figure 3: Supply voltage and wordlength scaling

Since, we know the effect of input scaling on the length of the active critical circuit paths and the effect of voltage scaling on the delay characteristics of the critical paths, a lookup table is constructed that determines different supply voltage levels as per the timing requirements of the requisite active critical paths. The advantage of such a lookup table, having wordlength and corresponding voltage entries is that it ensures that no MSB bits get corrupted due to signal scaling and supply voltage modulation.

Process variations cause the performance characteristics (delay, power, etc) of the manufactured devices to vary from their nominal values. If the delay of the circuit changes due to process variations, the entries of the lookup table as described in the previous section are no longer optimal from a power consumption perspective. For example, if the delay of the circuit increases, then a higher voltage level is necessary to meet the timing requirements for a particular wordlength. Therefore to counter the effect of process variations many loci are stored in the system at design time corresponding to different variations on DSP modules. During characterization phase of the device, timing tests are used to pick the best operating locus by determining the process variations on different modules. At run-time, device is operated on the locus under varying channel conditions for low power operation [6].

4.2 Low-power digital filters

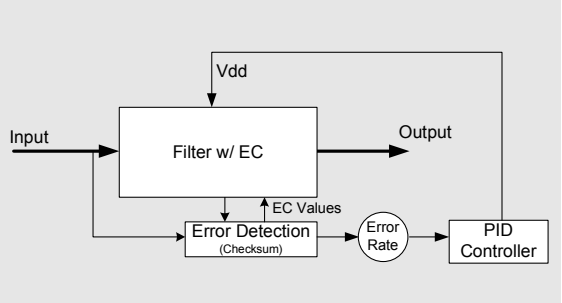


Figure 4: Supply voltage feedback control

Checksum codes are used for low power operation of digital filters by voltage overscaling and operating the filter within allowable error rate. The feedback control system adjusts the supply voltage according to the error rate in the system. Error rate is monitored in the system using a counter which counts the number of errors detected by the error detection block over a certain period of time. If the error rate is low then the computations in the circuit are completing too quickly (critical paths are not being excited) and/or the error correction is working well enough to lower the supply voltage further to save power. On the other hand, if the error rate increases then the filter components are not meeting the timing constraints and error correction is not sufficient to compensate for the number of errors occurring. In that case, feedback control increases the supply voltage to bring the error rate within acceptable range. A PID feedback controller modulates the supply

voltage in proportion to the error rate in the system. To ensure that the feedback control system is stable, the error rate sample period is set to the minimum time needed to change a single voltage step. A filter fitted with continuous error detection and correction along with a feedback controller is shown in Figure 4.

4.2.1 Checksum based error detection

Linear digital circuits can be represented by state variable systems. Let $(u_1,u_2...u_m)$ and $(y_1 \ldots y_w)$ be the primary inputs and primary outputs of the circuit. If $s(t)=[s_1(t), s_2(t) \ldots s_n(t)]^T$ are the system states and $u(t)=[u_1(t), u_2(t) \ldots u_n(t)]^T$ are the system input at time t then the state variable form is:

$$S(t+1) = As(t) + Bu(t)$$
$$Y(t+1) = Cs(t) + Du(t) \qquad (1)$$

Where A,B,C and D represent the arithmetic operations performed on m primary inputs u(t) and current states s(t) to generate w primary outputs y(t), and the next states s(t+1).

A coding vector CV= $[\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_n]$ is used for encoding the matrices A and B such that X=CV.A and Y=CV.B. To detect an error, a check variable c(t+1) is computed as $c(t+1)= X.s(t)^T+Y.u(t)^T$ and subtracted from $CV.s(t+1)^T$. A non-zero value of e(t+1), the result of this subtraction, indicates a transient or a permanent fault in the circuit. In the absence of any error, $c(t+1)=CV.s(t+1)^T$ and e(t+1) [8] is determined as:

$$e(t+1) = CV.s(t+1)^T - c(t+1) \qquad (2)$$

4.2.2 Guided Probabilistic Compensation (GPC) Architecture

GPC architecture [9] is explained in the following section. In GPC shadow latches [10] are used for diagnosis of erroneous states and system level error compensation is performed under the assumption that the shadow latches clearly delineate the erroneous states from the error-free ones.

For error detection in the system states, we use the concept of low precision shadow latches that augment the full precision registers as shown in Figure 5. Since voltage overscaling causes errors in the MSB bits, we only monitor errors in the high order bits using *low precision* shadow latches that operate on a delayed clock compared to the main circuit flip-flops. In case of no error in the circuit paths under voltage overscaling, the values in the MSB bits of the main register and shadow latch are the same. However, if the critical path violates the timing requirements under scaled supply voltage for any input, the comparator flags an error due to mismatch between the main flip-flop and shadow latch values. To ensure that shadow latches always capture the correct data, the operating voltage should be constrained so that under worst case, the logic delay does not exceed the shadow latch's setup time. Also, to minimize the power cost because of added latches and comparator, we limit the number of MSB bits to be monitored. In our experiments, we used shadow latches to monitor just four MSB bits of the data word.

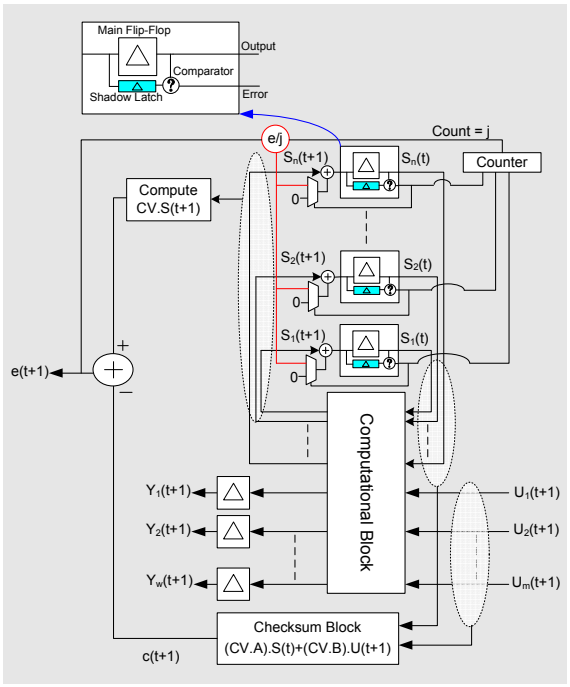


Figure 5: GPC Architecture

Under voltage overscaling, GPC architecture will work as following:

- If the timing requirements of some logic paths are violated in time (t,t+1) then one or more states will have erroneous data at time t+1 and e(t+1) will be non-zero.
- One or more shadow latches will flag an error due to mismatch of the main flip-flop and shadow-latch values.
- The total number of erroneous states is calculated at every time instance t by a counter.
- Divide the error in the system e(t+1) by the number of erroneous states. If the counter value is j, then the compensation value for the erroneous states is e(t+1)/j.
- The error signal of every state is used as the control signal of the multiplexer and selects between a zero or e(t+1)/j correction value (see Figure 5). This correction value is added to the erroneous states at time t+2.

In case, only one state is erroneous, the error value is used to compensate only that erroneous state. If two states are found to be erroneous, then the error value is divided by 2 and both erroneous states are compensated with the same error/2 value. No

compensation is performed for the states that are not flagged erroneous by the shadow latches. The presented guided probabilistic error correction technique only compensates the error at the erroneous states with an overall goal of minimizing the system noise.

5. Nested Loop Architecture

In prior EVM feedback based communication system, system V_{dd} and W is modulated by operating the device on the optimal system locus. However relationship between V_{dd} and W changes under process variations. Therefore for optimal power consumption and also to make sure that no MSB errors occur because of V_{dd} scaling, various loci are stored in the system at design time corresponding to different process variations on various DSP modules. In characterization phase, POTTs is used to pick the correct locus for system operation.

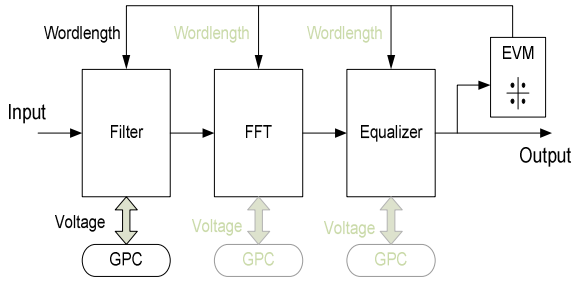


Figure 6: Dual loop architecture

The proposed *dual nested loop architecture* does not require storage of any pre-calibrated lookup tables (loci) in the system. In this architecture, two signal quality metrics are used to independently control wordlength and supply voltage of the DSP modules. EVM feedback control is used as outer loop that defines the wordlength of baseband receiver. Checksum based error rate as defined in Section 4.2 is used in inner control loop to set the supply voltages of the DSP modules. Each DSP module has its own supply voltage feedback control as shown in Figure 6 (In this work, GPC is implemented only on filter). For a given system, maximum allowable EVM value and error rate is defined in the control loops. At run-time, feedback control loops independently strive to operate the system at the defined signal quality limits, thus saving considerable system power under good channel conditions. The key advantage of having two feedback controls is that because of independent control of system V_{dd} and W, this scheme is immune from process variations effects.

In any feedback system, stability is a main concern. To make sure that the proposed dual loop structure is stable, the minimum period for changing the wordlength equals to the time required for supply voltage step change. Also, checksum error rate value is set to a low value. This ensures that supply voltage loop follows the wordlength loop and supply voltage will be lowered only if some bits have been dropped by the outer (wordlength) loop. Moreover, in case of sudden adverse change in channel conditions, wordlength and supply voltage is restored to the highest values to ensure that system BER does not increase more than the maximum allowed limit.

6. Experimental results

The OFDM based transceiver model used for simulation results is shown in Figure 7. At the transmitter side, data encoding, QPSK modulation and IFFT are implemented in the floating point units. To model various channel conditions (good to bad) different values of white noise, interference and multipath fading effects are used. In the receiver demodulator, a 4-tap low pass filter is implemented along with 128-point FFT and MMSE equalizer. Reed-Solomon (255,223) codes were implemented with a correction capability of 16 symbols. All the voltage dependent receiver modules are implemented in fixed point Q4.8 binary format, which enables us to measure system performance changes by signal scaling based voltage adjustment. Negative numbers are in two's complement form. For time delay and power estimations of the circuit, a full-adder was implemented in HSPICE with 70nm BPTM device model [14] and is used as the basic building block of every receiver module. Average power consumption in the receiver modules is based on the number of full adders in those circuits and their average switching activity. The 4-tap IIR filter used in the simulations operates at four times a higher frequency than FFT, thus consuming significant power. In this phase of experiments, GPC architecture is only implemented on low pass filter. The system EVM variations and power savings under different channel conditions without any process variations is shown in Figure 8.

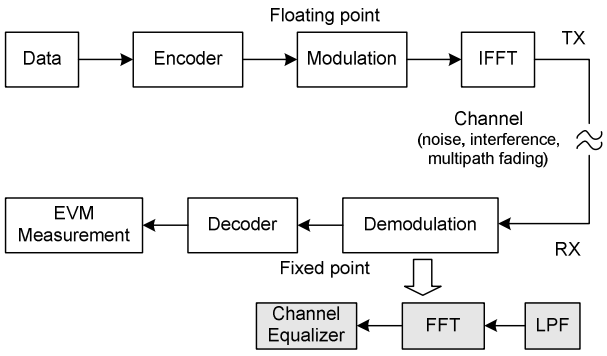


Figure 7: Simulation model

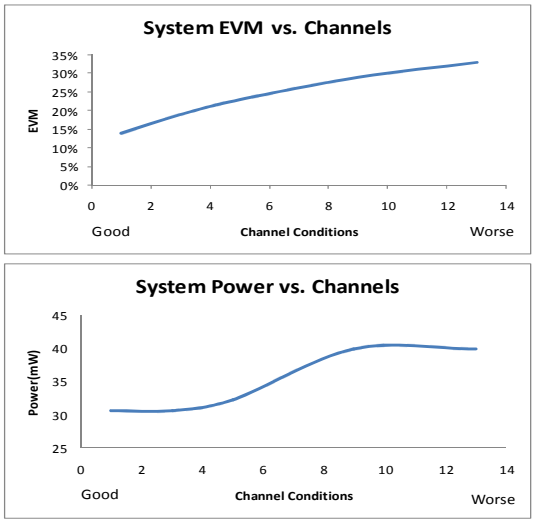


Figure 8: System EVM and Power consumption under different channel conditions.

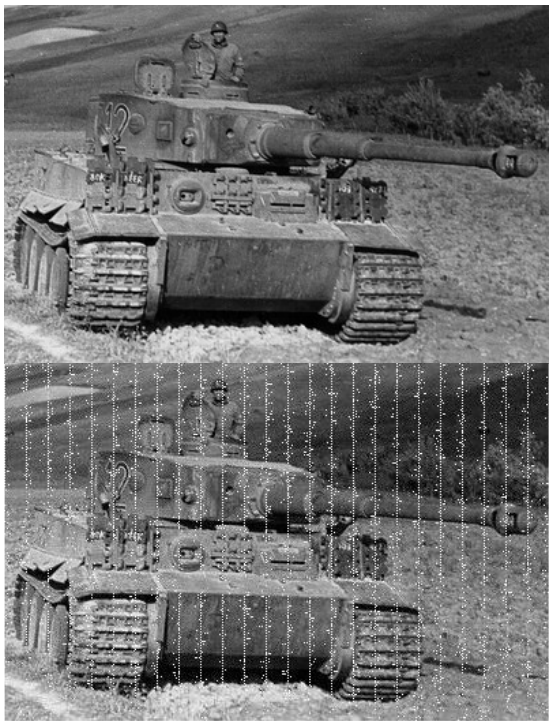


Figure 9: Images received with low and high checksum error rate.

Table 1: Process variations effects

Channel – 1 (Good)		
Process Variation (V_t)	**Wordlength**	**Supply Voltage (volts)**
0 volts	9	0.80
0.04 volts	9	0.90
-0.04 volts	9	0.75

The effect of low and high checksum error rate is shown in Figure 9. The top image is clean as no MSB errors were introduced when checksum error rate (0.001) is low. However this is not the case in second image, when error rate is relatively high (0.01). As shown in Table 1, the nested dual loops result in different values of V_{dd} and W (thus different power consumption) under different process variations. The average power savings in the scheme is approximately 30% under good operating conditions. The default value of threshold voltage V_t is 0.2 volts, maximum system wordlength is 12 bits and maximum system voltage is 1 volts.

7. Conclusion

In this paper, a novel dual feedback based design approach is proposed that allows the baseband unit of a wireless OFDM system to adapt dynamically to channel conditions as well as manufacturing process variations, with end objective of low power operation.

8. References

[1]. T. Pering, T. Burd and R. Broderson, "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms," Proc. Of Int'l Symp. on Low Power Electronics and Design(ISLPED 98), ACM Press, 1998, pp. 76-81.

[2]. H. Choi and W.P. Burleson, "Search-based wordlength optimization in VLSI/DSP synthesis", VLSI Signal Processing Workshop VII, pp.198-207, Oct.1994.

[3]. K. Han and B.L. Evans, "Wordlength optimization with complexity and distortion measure and its applications to broadband wireless demodulator design", Proc. IEEE ICASSP2004, vol.5,pp.37-40, May 2004.

[4]. A.G. Dempster and M. D. Macleod, "Variable statistical wordlength in digital filters", IEE Proc.-Vis. Image Signal Process., Vol. 143, No. 1, Feburary 1996, pp.62-66.

[5]. S. Yoshizawa and Y. Miyanaga "Tuneable wordlength Architecture for a Low Power Wireless OFDM demodulator", IEICE Trans. Fundamentals, Vol.E89-A, Issue10, October 2006, pp. 2866-2873.

[6]. Muhammad Mudassar Nisar, Abhijit Chatterjee, "Test Enabled Process Tuning for Adaptive Baseband OFDM Processor", VTS 2008, pages 9-16.

[7]. K. A. Bowman, J. D. Meindl, "Impact of die-to-die and within die variation parameter fluctuations on the maximum clock frequency distribution for a gigascale integration." IEEE Journal of Solid State Circuits, Vol. 37, Issue 2, Feb. 2002, pp. 183-190.

[8]. V. S. Nair and J. A. Abraham, "Real-number codes for fault-tolerant matrix operations on processor arrays," IEEE Trans. on Computer, vol.39, pp. 426-435, April 1990.

[9]. Muhammad M. Nisar, Abhijit Chatterjee, "Guided probabilistic Checksums for Error Control in Low Power Digital-Filters", IOLTS 2008.

[10]. D. Ernst, S. Das, D. Blaauw, "RAZOR: Circuit-level Correction of Timing Errors for Low-Power Operation", IEEE Micro, Vol. 24, Issue 6, Nov-Dec 2004 pp:10-20.

[11]. Tasic, A., Serdjin, W.A., Long, J.R., "Adaptive multi-standard ciruits and systems for wireless communications", IEEE Circuits and Systems Magazine, Vol 6, Issue 1, pp. 29-37.

[12]. Abidi, A., Pottie, G.J., Kaiser, W.J., "Power-conscious design of wireless circuits and systems", Proceedings of the IEEE, Vol. 88, Issue 10, Oct 2000, pp. 1528-1545.

[13]. J.G. Proakis, Digital Communications, 2nd Edition, McGraw-Hill, 1989.

[14]. BPTM 70nm: Berkley predictive technology model.

Session 1B

SoC Verification

Efficient Techniques for Directed Test Generation using Incremental Satisfiability*

Prabhat Mishra and Mingsong Chen

Department of Computer and Information Science and Engineering
University of Florida, Gainesville FL 32611-6120, USA
{*prabhat, mchen*}*@cise.ufl.edu*

Abstract

Functional validation is a major bottleneck in the current SOC design methodology. While specification-based validation techniques have proposed several promising ideas, the time and resources required for directed test generation can be prohibitively large. This paper presents an efficient test generation methodology using incremental satisfiability. The existing researches have used incremental SAT to improve counterexample (test) generation involving only one property with different bounds. This paper is the first attempt to utilize incremental satisfiability in directed test generation involving multiple properties. The contribution of this paper is a novel methodology to share learning across multiple properties by developing efficient techniques for property clustering, name substitution, and selective forwarding of conflict clauses. Our experimental results using both software and hardware benchmarks demonstrate that our approach can drastically (on average four times) reduce the overall test generation time.

1 Introduction

Functional verification is a major bottleneck in System-on-Chip (SOC) design due to the combined effects of increasing design complexity and decreasing time-to-market. Simulation-based validation is the most widely used form of SOC verification using functional test programs. There are three types of test generation techniques: random, constrained-random, and directed. The directed tests can reduce overall validation effort since shorter tests can obtain the same coverage goal compared to the random tests. However, directed test generation is mostly performed by human intervention. Due to manual development, it is infeasible to generate all directed tests to achieve a comprehensive coverage goal. Automatic test generation is the alternative to address this problem.

Test generation using model checking is one of the most promising approaches due to its capability of automatic test generation. However, it is unsuitable for large designs due to the state space explosion problem. SAT-based bounded model checking (BMC) restricts search space that is reachable from initial states within a fixed number (k) of transitions, called *bound*. After unrolling the model of design k times, the BMC problem is converted into a propositional satisfiability (SAT) problem. SAT solver is used to find a satisfiable assignment of variables that is converted into a counterexample. If the *bound* is known in advance, SAT-based BMC is typically more effective for counterexample (test) generation because search for counterexample is faster and SAT capacity reaches beyond BDD capacity [1]. The effectiveness of BMC can be further improved by observing that the search for counterexamples of increasing lengths is translated into a sequence of SAT checks. Therefore, it is possible to exploit the similarity of these SAT instances by forwarding clauses learned during conflict analysis from one instance to the next.

Incremental SAT-based BMC is very promising to reduce the test generation complexity but all the existing approaches are applicable for a test generation scenario consisting of one design and only one property (with varying bounds). This paper proposes a novel methodology to exploit incremental SAT in the context of test generation involving one design and multiple properties. The basic idea of our approach is to reuse the learning of one test generation instance for other related test generation scenarios. It is important to note that the design remains the same for all test generation scenarios. Although, each test generation instance requires a different property, several properties related to testing specific functionalities are similar or have a significant overlap. A major challenge in implementing this idea is to identify the property similarities and perform efficient clustering to share learning and thereby reduce the test generation time. To enable the knowledge sharing across multiple properties, we have developed a number of efficient techniques for property clustering, name substitution, and selective forwarding of conflict clauses. The contribution of this paper is the development of a number of conceptually simple, but very effective, techniques to generate drastic reduction in directed test generation time.

The rest of the paper is organized as follows. Section 2 presents related work addressing test generation approaches. Section 3 presents our test generation methodology using incremental satisfiability. Section 4 presents the experimental results. Finally, Section 5 concludes the paper.

2 Related Work

Model checking [4] has been successfully used in software and hardware verification as the test generation engine [3, 18].

*This work was partially supported by grants from Intel Corporation and NSF CAREER award 0746261.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.72

Model of design is applied to a model checker along with negated temporal logic properties to exploit falsification capability of model checking. However, traditional model checking does not scale well due to the state explosion problem. Biere et al. [2] introduced bounded model checking (BMC) combined with satisfiability solving. The recent developments in SAT-based BMC techniques have been presented in [15]. BMC is an incomplete method that cannot guarantee a true or false determination when a counterexample does not exist within a given bound. However, once the bound of a counterexample is known, large designs can be falsified very fast since SAT solvers [14] do not require exponential space, and searching counterexample in an arbitrary order consumes much less memory than breadth first search in model checking. The performance of bounded and unbounded algorithms was analyzed on a set of industrial benchmarks in [16]. The capacity increase of BMC techniques has become attractive for industrial use. Intel study [5] showed that BMC performs better over unbounded model checking for real designs. The efficiency of test generation can be further improved by employing incremental SAT solving.

Incremental SAT solvers [6, 12, 17] try to leverage the similarity between the elements of a sequence of SAT instances; most do so by re-utilizing conflict clauses, though when many closely related instances must be solved, caching solutions [11] and incremental translation [13] can also be effective. If a SAT instance is obtained from another by adding some clauses (as in [7]), then all conflict clauses of the first can be forwarded to the second. This is correct because the second instance implies the first, which in turn implies all its (conflict) clauses. Therefore, when clauses are only added through the sequence of instances, there is no need to screen conflict clauses to determine which ones can be forwarded. This, on the other hand, is necessary when arbitrary clauses are both added and subtracted to create a new instance. A common approach to such general case is to have the incremental SAT solver keep track of whether a conflict clause depends on some removed clauses. The approach of [12] is to record, for each conflict clause, the clauses that made up the corresponding implication graph. This approach does not require any prior knowledge of the subsequent SAT instances to be solved incrementally, and does not restrict the changes possible from one instance to the next; however, keeping track of dependencies may be expensive. Strichman [17] was the first to observe that in BMC some clauses are known to survive through all instances in the sequence. A formula passed by BMC to the SAT solver contains clauses that describe the transition relation of the model unrolled a number of times. These clauses are not discarded when the length of the counterexample is increased. Hence, a conflict clause that depends only on them can be forwarded.

To the best of our knowledge, all the existing approaches exploit incremental satisfiability to improve the test (counterexample) generation time involving only one property with different bounds. Our approach is the first attempt at utilizing incremental satisfiability across multiple properties in the context of directed test generation for validation of SOC designs.

3 Test Generation using Incremental SAT

The goal of our approach is to reduce the overall functional validation effort by reducing the test generation time for directed tests. We assume that the designer have developed a set of properties (one property to generate one directed test) based on a specific fault model. For example, a pipelined processor with n functional units needs $n(n-1)/2$ properties to activate all 2-unit interactions. Based on the structure of the design and the fault model, the generated properties can be clustered using functional as well as structural similarity e.g., all properties related to a particular execution path can be placed in a cluster. The basic idea is to learn from solving one property and share learning (through conflict clauses) for solving the similar properties in the cluster. While solving the first property, the SAT solver may have taken many wrong decisions (lead to conflicts) and therefore took long time to find a counterexample. By forwarding conflict clauses, we are ensuring that these wrong decisions are avoided while solving the similar properties. An important question is whether all the wrong decisions are relevant to all the properties in the clusters? Since the properties are similar but not the same, all the decisions are not relevant. Therefore, adding all the conflict clauses while solving the similar properties may increase the solution time. In our approach, we determine the common CNF (conjunctive normal form) clauses by computing the intersection of clauses and use this intersection information to exactly identify the conflict clauses that are relevant to solving the respective properties.

This paper focuses on test generation for safety properties. In this context, we are interested in finding a counterexample for each property. We assume that the bound is pre-determined and given as input to our method. Determination of bound is intractable in general. However, in the context of directed test generation, it is possible to determine the bound based on the structure of the design and the associated property.

Algorithm 1 describes our test generation methodology. It accepts a design and a set of properties as inputs and generates the testcases. Since one property is used to generate a testcase, the number of input properties is exactly same as the number of output testcases. The first step is to partition the set of properties into different clusters based on their similarity in terms of both structure and behavior. The second step is to select the base property who has the potential to generate maximum overall savings for the cluster by sharing learned conflict clauses. The third step computes the CNF clauses for all the properties in each cluster using the design and the respective bound. The fourth step performs name substitution to maximize knowledge sharing. The fifth step computes the intersection of CNF clauses between the base property and all the other properties in the cluster. The sixth step marks the clauses in the base property to indicate whether a particular clause is also in the clause set of another property in the cluster. The marking information includes the identifier of the property (or properties) with which the clause is identical. The next step uses an existing SAT solver to generate the conflict clauses and the counterexample (test) for the base property. Based on the intersection

information with the base property, the set of conflict clauses is filtered to identify the relevant ones for solving the other properties in step 8. The final step uses the relevant conflict clauses to solve the remaining properties using our approach. The algorithm reports all the generated counterexamples (tests).

Algorithm 1: *Test Generation using Incremental Satisfiability*
Inputs: i) Design, D
ii) Properties, P, and their respective satisfiable bounds
Outputs: Testcases
Begin
1. Cluster the properties based on similarity
for each cluster, i, of properties
2. Select the base property P_1^i and generate CNF, CNF_1^i
for j is from 2 to the $size_i$ of cluster i
/* P_j^i is the j^{th} property in the i^{th} cluster */
3. Generate CNF, $CNF_j^i = BMC(D, P_j^i, bound_j^i)$
4. Perform name substitution on CNF_j^i
5. INT_j^i = ComputeIntersection(CNF_1^i, CNF_j^i)
6. Mark the clauses of CNF_1^i using INT_j^i
endfor
/* Generate the counterexample and record conflict clauses */
7. $(ConflictClauses_i, test_1^i)$ = SAT(CNF_1^i)
$Testcases = \{test_1^i\}$
for j is from 2 to the $size_i$ of cluster i
/* Find relevant ones for P_j^i from conflict clauses
8. CC_j^i = Filter ($ConflictClauses_i$, j)
endfor
for j is from 2 to the $size_i$ of cluster i
9. $test_j^i$ = SAT($CNF_j^i \bigcup CC_j^i$)
$Testcases = Testcases \cup test_j^i$
endfor
endfor
return *Testcases*
End

We use a simple example to illustrate how Algorithm 1 works. Let us assume that we are interested in generating tests using n properties for a design. The first step divides the properties into m ($m \leq n$) clusters based on property similarities. Each cluster can have different number of properties. In the worst case, each cluster can have only one property which is not suitable for test generation using incremental satisfiability. However, this scenario is rare in practice since a typical design uses thousands of properties for directed test generation and majority of them share significant parts of the design functionality. For ease of illustration, let us assume that there is a cluster with three similar properties, $\{P_1, P_2, P_3\}$. Let us further assume that the second step selects P_1 as the base property using the method described in Section 3.1. The fourth step computes intersection of CNF clauses of P_1 with P_2, and P_1 with P_3. This information is used to filter conflict clauses (generated while solving P_1) relevant for P_2 and P_3 in step 8. The last step adds the relevant conflict clauses while solving the respective properties to reduce the test generation time. The remainder of this section describes three important steps in our approach: property clustering, computation of intersections, and identification of relevant conflict clauses.

3.1 Clustering of Similar Properties

One obvious, but costly, way to determine property similarity for clustering is to compute the intersection of CNF clauses between properties. We can cluster the properties that have a relatively large number of clauses in the intersection. This is a very time consuming step because it requires $n(n-1)/2$ intersections for n properties. A simple and natural way to cluster properties is to exploit the structural and behavioral information of the properties. As mentioned earlier, in the context of directed test generation, properties are generated based on a set of fault models to obtain a functional coverage goal. Each fault model tries to cover different parts of the design (e.g., all computation nodes, execution paths, various interactions, etc.). Therefore, we can cluster the properties that try to cover a specific part of the design using the same fault model. For example, in an SOC environment, the properties can be clustered based on whether they are related to verifying the processor, coprocessor, FPGA, memory, bus synchronization, or controllers. Each cluster can be further refined based on structural details of each component. For example, the processor related properties can be further divided based on what execution path they cover such as ALU pipeline, load-store pipeline etc.

Once clustering is completed, we need to determine the base property of the cluster. In our approach, the base property is solved first and its conflict clauses are shared between the remaining properties. Although, any property in the cluster can be used as the base property for that cluster, our studies have shown that certain properties serve better as base property and thereby generates maximum overall savings for the cluster. We need to consider two important factors while choosing a base property for a cluster. First, the base property should be able to generate a large number of conflict clauses. In other words, a weak base property may find the satisfiable assignment quickly without making mistakes (generating conflict clauses). In this scenario, the remaining properties have nothing to learn from the base property. Secondly, the SAT checking time for the base property should be relatively small. This will ensure that the overall gain is maximized by reducing the solution time of the properties which takes longer time to solve. In fact, none of these requirements can be determined without actually solving them. Based on our experience, we have observed that the following heuristics works well most of the time.

- Choose the property with the smallest bound that have significant variable and/or sub-expression overlap with the rest of the properties in the cluster.

- If the property bounds in the cluster are same, choose the property with the smallest number of variables that have significant variable and/or sub-expression overlap with the rest of the properties in the cluster.

3.2 Name Substitution for Computation of Intersections

Name substitution is an important preprocessing step in Algorithm 1. Currently there are a few BMC tools that can support the name mapping from the variables of the CNF clauses and the names in the model of the unrolled design. So the variables of the CNF clauses of two different properties may not have any name correspondence. In other words, the same variable in two properties may have different name in their respective CNF clauses. Therefore, without name substitution (mapping), it will miss the structural similarity. As a result, the computed intersection will be small and will adversely affect the sharing of learned conflict clauses. Our experimental studies have shown that the improvement in test generation time without using name substitution is negligibly small due to very small number of clauses being forwarded as a result of small number of clauses in the intersection. Since the properties are similar and the design is exactly the same, the size of the intersection is very large when our name substitution method is employed.

Our framework uses zChaff SAT solver which accepts the input in the DIMACS format. The input DIMACS file for each property provides the name mapping from the CNF variable to the unrolled design. The following example shows that the variable 8 is used in CNF to refer to the 7^{th} bit of variable *var* in the design specification at time step 1. This can also be written as, $8 => var[6]\_1$.

```
c 8 = V1_var[6]
```

Given two DIMACS files $f1$ and $f2$ for two properties P_1 and P_2 respectively, the name substitution is a procedure that changes the names of clause variables of $f2$ using the name mapping defined in $f1$. Figure 1 shows an example for name substitution. Before the name substitution, the intersection $(f1 \cap f2)$ is empty. However, after the substitution, there are two common clauses in the intersection $(f1 \cap f2')$. The complexity of both name substitution and computation of intersection is linear (using hash table) to the size of the DIMACS file of the properties. Therefore, the time required for name substitution and intersection computation is negligible compared to the SAT solving time for the complex properties.

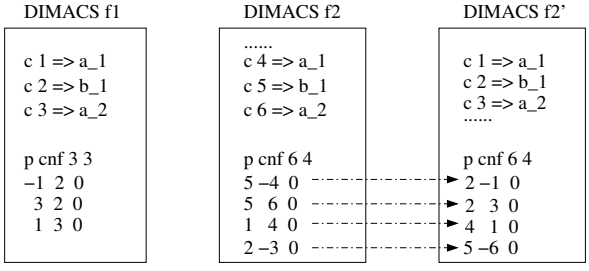


Figure 1. An example of name substitution

It is important to note that the same variable at different time steps can be assigned a different number. Therefore, the name mapping (substitution) method needs to consider the same variable at different time steps in the CNF clauses of the same property as well as in the CNF clauses for the different properties in the same cluster. Moreover, the name mapping routine needs to remap some of the variables in the CNF clauses. For example, when the variable 4 in file $f2$ (in Figure 1) is replaced with the variable 1 (in $f2'$), the name mapping routine needs to remap the original variable 1 in file $f2'$ to a different variable.

3.3 Identification and Reuse of Common Conflict Clauses

Our implementation of relevant conflict clause determination is motivated by the work of [17] which proved that for two set of CNF clauses C_1 and C_2, and their intersection φ, using the conflict clauses generated from the φ when checking C_1 will not affect the satisfiability of the CNF clauses $C_2 \bigcup \varphi$. Therefore, the conflict clauses generated from the intersection when checking the base property can be shared by other properties in the cluster.

Strichman [17] suggested an isolation procedure that can isolate the conflict clauses which are deduced solely from the intersection of two CNF clause set. We have modified the isolation procedure to improve the efficiency of test generation for a cluster of properties. We have modified zChaff [9] and use it as the SAT solver in our framework. The zChaff provides utilities for implementing incremental satisfiability. For each clause, it uses 32 bits to store a group id to identify the group where this clause belongs. Use of group id allows us to generate the conflict clauses for different properties when checking the base property. If the i^{th} bit of the group id is 1, it implies that the clause is shared by the CNF clauses of property P_i. If the clause of the base property is not shared by any property, the field will be 0.

Assume that there are k properties in a cluster with P_1 as the base property. Therefore, there are k sets of clauses with C_1 as the base set (CNF clauses for P_1), and $C_2, C_3, ..., C_k$ are $k-1$ similar sets with C_1. We use the following steps to calculate the conflict clauses for $C_2, C_3, ..., C_k$ when solving C_1.

1. During preprocessing, for each clause in C_1, if this clause exists in $C_i (2 \leq i \leq k)$, then mark the i^{th} bit of C_1's group id 1.

2. When one conflict clause is encountered during the checking of the base property, collect all the group ids of the clauses which leads to the conflict. The group id of the conflict clause is the logical "OR" of these group ids.

3. For each conflict clause, if the i^{th} bit of the group id is 1, then this conflict clause can be shared by C_{i+1}.

Figure 2 illustrates how this computation is done using an example implication graph. The implication graph is a directed acyclic graph where each vertex represents an assignment of the variable and each edge implies that all the in-edges implicate the assignment of the vertex. For example, $x4@4$ means variable $x4$ is assigned value 1 at decision level 4. The graph has a clause $(x1' + x4 + x5)$, we call it the *antecedent* clause of $x4$ i.e.,

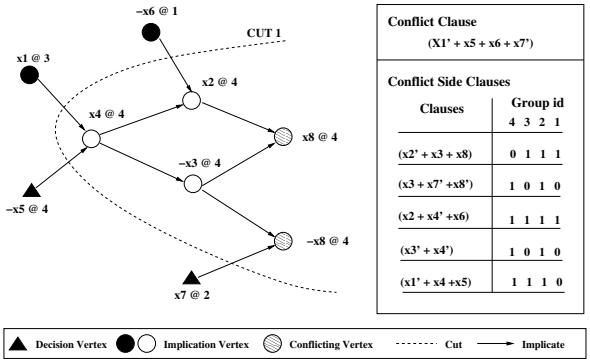


Figure 2. An example of implication graph

the assignments $x1 = 1$ and $x5 = 0$ imply $x4 = 1$. Only the implication vertex (non-decision vertex) has an antecedent clause. The conflict exists when in the implication graph, one variable is assigned both value 0 and 1. So the SAT solver will analyze the conflict and find the reason expressed by the conflict clause. A conflict clause can be found by a bipartition of the implication graph. The side containing the conflicting vertex called *conflict side*, and the other side is called *reason side* which can be used to form the conflict clause. In Figure 2, $CUT1$ is a cut that divide the implication graph into two parts. We list the conflict clause on the reason side and all other clauses on the conflict side. In our approach, each conflict side clause has a group id which is marked during the preprocessing step. So the group id of the conflict clause is the logical "OR" value of all the group ids of the conflict side clauses. For example, in Figure 2, we use four bits to express the group id, and the group id of the conflict clause is 0010. In other words, this conflict clause can only be forwarded to clause set C_2. Therefore, the use of this conflict clause in solving P_2 will reduce the SAT solving (test generation) time.

4 Experiments

We have applied our test generation methodology for validation of various software and hardware designs. In this section, we present two case studies from two different domains: a stock exchange system, and a VLIW implementation of the MIPS architecture. Both experiments were performed on a Linux PC using 2.0GHz Core 2 Duo CPU with 1 GB RAM. In our experiments, we used the NuSMV [8] as our BMC tool to generate the CNF clauses (in the DIMACS format) for the design and properties. We modified zChaff [9] to integrate our methods for name substitution, clause intersection and constraint sharing described in Section 3.

4.1 A Stock Exchange System

The purpose of the on-line stock exchange system (OSES) is to process three scenarios: accept, check and execute the customer's orders (market order and limit order). The system uses the UML activity diagram as its behavior specification and JAVA based implementation. The UML specification has 27 activities, 29 transitions and 18 key paths. We translate the specification into the NuSMV input and generate the corresponding test case based on various functional coverage criteria. We group the properties into five clusters. The first cluster consists of $\{Path_1, Path_2\}$ with $Path_1$ as the base property in the cluster. Similarly, each of the remaining clusters consists of four properties with the first one as the base property.

Table 1. Test Generation for Stock Exchange System

Properties (Tests)	Preproc. Time (s)	zChaff [14] (sec)	Our Method (sec)	Improv. Factor
$Path_1$	Base	0.37	0.37	1.00
$Path_2$	3.79	59.45	4.06	14.66
$Path_3$	Base	2.82	2.82	1.00
$Path_4$	4.16	2.89	0.54	5.35
$Path_5$	4.09	29.67	4.04	7.34
$Path_6$	3.73	42.75	6.28	6.81
$Path_7$	Base	0.44	0.44	1.00
$Path_8$	4.14	7.36	0.30	24.86
$Path_9$	3.88	113.70	33.23	3.42
$Path_{10}$	3.79	40.41	6.53	6.19
$Path_{11}$	Base	4.58	4.58	1.00
$Path_{12}$	4.24	13.04	1.20	10.91
$Path_{13}$	4.44	23.74	14.60	1.63
$Path_{14}$	4.02	102.76	31.42	3.27
$Path_{15}$	Base	0.25	0.25	1.00
$Path_{16}$	4.36	64.04	1.26	50.66
$Path_{17}$	4.38	185.03	5.05	36.62
$Path_{18}$	4.02	176.77	68.78	2.57
Average	**4.08**	**48.33**	**10.32**	**4.68**

Table 1 shows the results involving all the 18 properties of key paths. The first column indicates the properties used for test generation. The second column indicates the preprocessing time that includes the time for both name substitution and computation of clause intersection. This column is not applicable for the base property since the base property is solved using the existing approach. The third and fourth columns indicate the time (in seconds) required to generate the counterexample (test) by zChaff [14] and our approach, respectively. The last column indicates the *improvement factor*[1] in test generation time. The last row in the table presents the averages of all the entries. Our approach can produce almost five times improvement compared to zChaff, a popular CNF SAT solver. Our approach should be used in the test generation scenarios where SAT takes a long time to find a counterexample. As a result, the cost (preprocessing time) will be negligible compared to the savings in test generation time.

4.2 A VLIW MIPS Processor

We applied our methodology on a single-issue MIPS [10] architecture. Figure 3 shows the simplified version of the VLIW MIPS architecture. It has five pipeline stages: fetch, decode, execute, memory (MEM), and writeback. The *execute* stage

[1]The improvement factor is computed as the ratio between the third and fourth column entries.

has four parallel execution paths: integer ALU, 7 stage multiplier (MUL1 - MUL7), four stage floating-point adder (FADD1 - FADD4), and multi-cycle divider (DIV). The oval boxes represent units and dashed boxes represent storages. The solid lines represent instruction-transfer paths and dotted lines represent data-transfer paths.

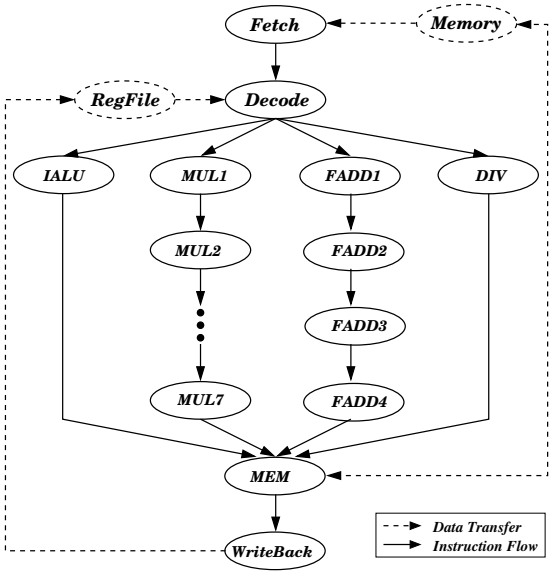


Figure 3. The VLIW MIPS architecture

We translated the specification into the NuSMV input and generated the required directed tests based on various path coverage criteria. Table 2 shows the results. Each row presents the averages of all the properties in that cluster. The first column indicates the four clusters corresponding to the four execution paths in Figure 3. The second column represents the average number of CNF clauses for each property in that cluster. The third column shows the average number of clauses in the intersection. The fourth and fifth columns indicate the time (in seconds) required to generate the counterexample (test) by zChaff and our approach, respectively. Our approach is able to improve the test generation time on average by four times.

Table 2. Test Generation for MIPS Processor

Clusters	CNF Clauses	Intersec. Size	zChaff [9]	Our Method	Improv. Factor
CL_{ALU}	460994	457168	19.35	5.10	3.79
CL_{FADD}	592119	67894	61.61	42.46	1.45
CL_{MUL}	854386	522283	718.85	159.21	4.51
CL_{DIV}	526517	457160	35.07	8.19	4.28
Average	**608504**	**376126**	**208.72**	**53.74**	**3.88**

5 Conclusions

Directed test vectors can reduce overall validation effort of both hardware and software designs since shorter tests can obtain the same coverage goal compared to the random tests. The applicability of the existing approaches for directed test generation is limited due to capacity restrictions of the automated tools. This paper addressed the test generation complexity by exploiting the commonalities between a set of similar properties using incremental satisfiability. The existing incremental SAT approaches are applicable only on a single property and it utilizes the learning between various bounds of the same property. To the best of our knowledge, our work is the first attempt to share learning across multiple properties. To enable knowledge sharing across multiple properties, we have developed a number of conceptually simple, but extremely effective, techniques including property clustering, name substitution, and selective forwarding of learned conflict clauses. Our experimental results using both hardware and software designs demonstrated on average four times reduction in directed test generation time.

References

[1] A. Biere, A. Cimatti, and E. M. Clarke. Bounded model checking. *Advances in Computers*, 58, 2003.

[2] A. Biere, A. Cimatti, E. Clarke and Y. Zhu. Symbolic model checking without BDDs. *TACAS*, 193–207, 1999.

[3] A. Gargantini and C. Heitmeyer. Using model checking to generate tests from requirements specifications. *ACM SIGSOFT Software Engineering Notes*, volume 24, 146–162, 1999.

[4] E. Clarke, O. Grumberg and D. Peled. *Model Checking*. MIT Press, Cambridge, MA, 1999.

[5] F. Copty et al. Benefits of bounded model checking at an industrial setting. *CAV*, 436–453, 2001.

[6] H. Jin and F. Somenzi. An incremental algorithm to check satisfiability for bounded model checking. *BMC*, 51–65, 2004.

[7] J. Hooker. Solving the incremental satisfiability problem. *Journal of Logic Programming*, 15(12):177–186, 1993.

[8] http://nusmv.irst.itc.it/. *NuSMV*.

[9] http://www.princeton.edu/ chaff/zchaff.html. *zChaff*.

[10] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers, 2003.

[11] J. Kim et al. On solving stack-based incremental satisfiability problems. *ICCD*, 379–382, 2000.

[12] J. Whittemore, J. Kim, and K. Sakallah. SATIRE: A new incremental satisfiability engine. *DAC*, 542–545, 2001.

[13] M. Benedetti and S. Bernardini. Incremental compilation-to-sat procedures. *SAT*, 2004.

[14] M. Moskewicz et al. Chaff: Engineering an efficient SAT solver. *DAC*, pages 530–535, 2001.

[15] M. Prasad, A. Biere and A. Gupta. A survey of recent advances in SAT-based formal verification. *STTT*, 7(2):156–173, 2005.

[16] N. Amla et al. An analysis of SAT-based model checking techniques in an industrial environment. CHARME, 254–268, 2005.

[17] O. Strichman. Pruning techniques for the sat-based bounded model checking problem. *CHARME*, 58–70, 2001.

[18] P. Mishra and N. Dutt. Graph-based functional test program generation for pipelined processors. *DATE*, 182–187, 2004.

Inline Assertions – *Embedding Formal Properties in a Test Bench*

Aritra Hazra, Priyankar Ghosh, Pallab Dasgupta and P. P. Chakrabarti
Department of Computer Science & Engineering,
Indian Institute of Technology Kharagpur, INDIA 721302.
{aritrah, priyankar, pallab, ppchak}@cse.iitkgp.ernet.in

***Abstract*— The scope of *immediate assertions* in SystemVerilog is restricted to Boolean properties, where as temporal properties are specified as *concurrent assertions*. Concurrent assertion statements can also be embedded in a procedural block – known as *procedural concurrent assertions* which are used under restricted situations. This paper introduces the notion of *inline assertions* which generalizes the embedding of temporal properties within the procedural code of a test bench. The paper proposes verification methodologies for inline assertions and compares them with the traditional approaches of formal property verification and dynamic assertion based verification. The paper also focuses on coverage related issues when the intent of a concurrent assertion is modeled as an inline assertion.**

I. Introduction

Capacity has always been the major limitation of model checking techniques for formal property verification (FPV) [2], [3]. On the other hand, formal properties or assertions have become very popular for capturing complex functional requirements and monitoring them dynamically during simulation. The dynamic approach, popularly known as bug hunting, relies on the simulation coverage of those behaviors that are relevant to the given properties, unlike model checking techniques that verify the properties over all behaviors.

Most assertions are relevant in certain *contexts*. For example, consider a property which says that *"if the input x is high for three consecutive cycles, then the output y must be asserted in the next cycle"*. The property may be developed as a *concurrent* SystemVerilog Assertion (SVA) [6] as:

```
property P;
   @(posedge clk)
   x ##1 x ##1 x |-> ##1 y ;
endproperty
```

The property is relevant only in the context where the input x has been high for three consecutive cycles. In all other contexts, the property is vacuously true. The context of a concurrent assertion is sometimes specified explicitly as the antecedent of an implication (as in this example), but it may also be implicit. Typically the person who writes an assertion knows the context of the property.

The success of property verification in bug hunting mode relies on (a) the coverage of states satisfying the context of a property, and (b) the exhaustiveness of the search for a refutation from a context satisfying state. When random simulation fails to reach states satisfying the context of a property, the verification engineer has to write directed test benches for driving the simulation towards these states.

In this work we study the option of embedding assertions within the procedural code of a test bench. These assertions, which we call *inline assertions* are expressed as follows:

```
assert property ( <expression> );
```

SystemVerilog [6] supports the embedding of *immediate assertions* within the procedural code of a test bench, but their scope is restricted to Boolean properties only. *Procedural concurrent assertions* can be positioned within the procedural code, but their usage is restricted. On the other hand, our *inline assertions* are generalized embedding of temporal properties. The following example demonstrates the notion of inline assertions.

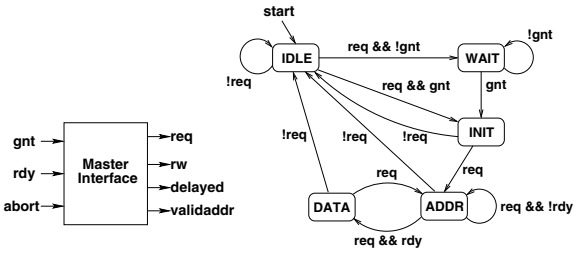


Fig. 1. Context State Machine

Example 1: Consider the specification of a master interface participating in a simple bus protocol with an arbiter and a slave device. Figure 1 shows the master interface and its behavior by the context state machine where each state specifies certain contexts of the protocol. The transitions among the context states of the master is based on the signals like *req* (request to arbiter), *gnt* (grant from arbiter) and *rdy* (slave ready). There are few other signals also, namely, (a) *rw* (indicating the nature of current transfer, write or read), (b) *validaddr* (flag indicating the validity of address floated on the bus), (c) *abort* (transfer terminated by the slave), and (d) *delayed* (delayed transfer).

Suppose we want to verify the following three properties on the master (DUT):

- $\mathcal{P}_0$: If the transfer waits due to non-availability of slave while the abort signal is low, then the delayed signal should be asserted in the next cycle.
- $\mathcal{P}_1$: Once requested, the master holds the request until it gets the grant (owns the bus).
- $\mathcal{P}_2$: If the master is in address cycle of write transfer, it should always hold a valid address in the next cycle.

These properties are relevant in certain contexts of the protocol – $\mathcal{P}_0$, $\mathcal{P}_1$ apply only when the master is in WAIT phase and $\mathcal{P}_2$ apply

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.31

when it is in ADDR phase.

The following test bench for the master shows the use of *inline assertions* corresponding to the above properties.

```
module master_tb_top();
  /*clock & input-output signal definitions*/
  /*DUT (master) instantiation*/
  logic [2:0] state;
  initial state = 3'b000;
  always @(posedge clk) begin
    case (state)
      3'b000: // IDLE state
          state = (req)? 3'b001 : 3'b000;
          gnt = (req)? 1'b0 : gnt;
      3'b001: // WAIT state
          assert property (##1 req[*0:$] ##1 gnt);
          state = 3'b010;
          if (!gnt) begin
            abort = 1'b0;
            assert property (##1 delayed);
            gnt = 1'b1;
          end
      3'b010: // INIT state
          state = (req)? 3'b011 : 3'b000;
      3'b011: // ADDR state
          state = (req)? 3'b100 : 3'b000;
          rdy = 1'b1;
      3'b100: // DATA state
          state = (req)? 3'b011 : 3'b000;
          if(rw) assert property (##1 validaddr);
    endcase
  end
  /*clock generation block*/
endmodule
```
□

It may be noted that the structured test bench for the master models the abstract states of the protocol, and it is easy for the verification engineer to embed the inline assertions at the appropriate places of the test bench code.

The intuitive idea behind inline assertions is to start the matching of the assertion whenever the control flow reaches the statement of the procedural code where it is embedded. The context of the property is implicit in the position where it is embedded in the test bench, and only the *guarantee* is specified in the inline assertion.

One advantage of using inline assertions is that the verification engineer does not have to express the context of a property using a complex antecedent expression. For example, in order to express the properties in Example 1 as concurrent assertions in SVA [6], one has to either encode the state machine shown in the figure as an *auxiliary state machine* [3] or encode the contexts (such as WAIT, INIT and ADDR) within the property, as shown below.

```
property P0;
   @(posedge clk)
   (req && !gnt && !abort)
           |-> ##1 delayed ;
endproperty
property P1;
   @(posedge clk)
   (req && !gnt)
           |-> ##1 req[*0:$] ##1 gnt;
endproperty
property P2;
   @(posedge clk)
   (req && gnt) ##1 (req && rdy && rw)
           |-> ##1 validaddr ;
endproperty
```

A common problem with concurrent assertions having complex antecedents is that random simulation often fails to reach the context of the assertion within reasonable time, and therefore the assertion is *not covered* even though we may have 100% code coverage of the test bench. The coverage of inline assertions is more closely related to code coverage of the test bench, since the assertion is non-vacuously tested every time the control reaches the part of the test bench where it is embedded. In other words the task of reaching the context of an inline assertion is automatically addressed by the developer of the test bench.

With inline assertions, we have the option of switching over to formal search (such as bounded model checking) whenever the context of the assertion is reached. This is a major advantage, since often the truth of an assertion relies on what the DUT may do after reaching the context, *not on how it reached the context*. Complex concurrent assertions often use complex antecedents (having large sequential depth) to specify the context and a short consequent (that is, the guarantee). Such assertions are typically hard to verify formally. With inline assertions, we only verify the consequent, which is more amenable for formal verification.

The main limitation of using inline assertions is that the test bench may not reach all states that satisfy the context of the required property. Let S^* denote the set of context satisfying states of the DUT and let $S(T)$ denote the set of context satisfying states that can be reached using a given test bench T. There are two coverage questions to be answered here:

1) Is $S(T) = S^*$? In other words, is T capable of reaching all states satisfying the context of the property?
2) At a given point of time, has the test bench reached all states of $S(T)$? If so, then we may stop checking the inline assertion from this time.

The first problem is closely related to the traditional functional coverage of the test bench, and can be addressed using the same metrics that are used to assess the completeness of a test bench. This paper presents several approaches for addressing the second problem.

The main contributions of this paper are as follows:

1) We extend the notion of immediate assertions and procedural concurrent assertions to introduce generalized *inline assertions*.
2) We compare several methods to verify the same intent.
3) We present methods for ascertaining verification closure with given inline assertions and test bench.

II. Verification Methods

In this paper we will compare three approaches for verifying assertions. These are as follows:

(a) Assertion Checking over Random Simulation:

In this approach, we develop concurrent assertions and verify them dynamically over random simulation runs. The assertion is said to be covered if the simulation reaches a state satisfying its context. Tools like VCS [8] have support for reporting non-vacuous matches of an assertion.

(b) Formal Assertion Verification:

In this approach, we develop concurrent assertions and use a model checking based formal verification tool to verify the assertions exhaustively. This approach guarantees 100% assertion coverage but suffers from capacity limitations.

(c) Semi-Formal Inline Assertion Verification:

We propose a semi-formal methodology which is motivated by the fact that in most cases, the truth of context sensitive inline assertion depends not on *how the DUT reached the context, but on what the DUT did after reaching the context*. In this approach, we reach states satisfying the context of an assertion through simulation and switch over to bounded formal search methods (bounded model checking(BMC) [1]) to verify the assertion from the context satisfying states. Figure 2 demonstrates the approach proposed.

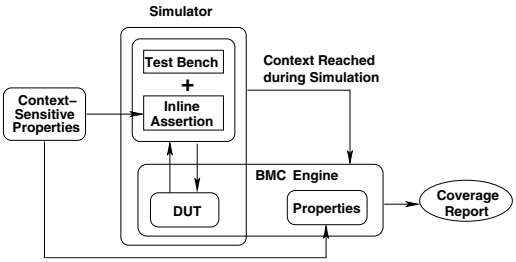


Fig. 2. Semi-Formal Verification Arch.

The goal of these verification methods is to validate a given assertion exhaustively only from the relevant states which are reachable in the total state-space. Repeated simulation of test bench over DUT uses depth-first-search (DFS) to verify an assertion, whereas formal assertion verification methods uses exhaustive breadth-first-search (BFS) to do the same. The verification exhaustiveness (reaching all possible relevant states) by random test bench simulation is restricted by test bench design and the number of repeated executions of it and hence may not be guaranteed. On the other hand, formal methods take care of all possible states together among which very few satisfies the actual context and are relevant for a given assertion. Thus, they stuck in capacity limitations and fails to verify complex properties. Using inline assertions, we provide a perfect blend of DFS and BFS search strategy. Firstly, DFS by test bench simulation helps to reach context-satisfying states for the inline assertion and thereafter BMC over inline assertion uses BFS with reasonable number of states and can verify the property within capacity limits. Figure 3 illustrates the comparison between three different search strategies.

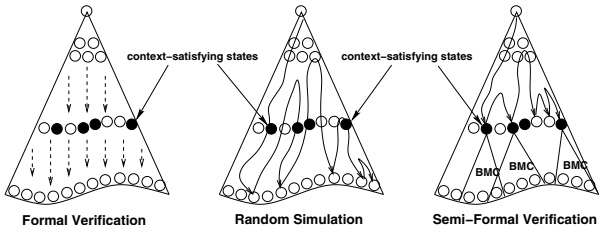


Fig. 3. Comparing Three Verification Search Strategies

If the context-satisfying states are relatively shallow, then formal search would reach those states within its capacity limit and thereby it can verify a property triggered from that context. Whereas, for deep contexts, formal search is ineffective due to capacity limitation, and so it fails to verify properties having deep contexts. Design bugs having deep contexts can be broadly classified into three categories – *(a) bugs that can be detected along all paths from all the context-satisfying states, (b) bugs that are manifested along some path from all the context-satisfying states and (c) bugs that are present only in some path from a few context-satisfying states*. The first category of bugs can be easily verified using simulation methods and therefore the easiest among the three. But, to detect the rest of the two bugs, we need semi-formal approaches. In category-(b), we simulate to reach the context (when inline assertion are hit) and trigger BMC every time to find the error; whereas the last category bugs are hardest to detect. However with the help of experts who can write test benches that reaches the contexts from where the bugs are manifested, the third category bugs can also be detected using the proposed semiformal approach.

The main challenge in this approach is to reach all context-satisfying states. The next section describes several approaches to determine when a test bench has reached all the context-satisfying states that it can possibly reach.

III. Verification Closure

For a given DUT, a test bench with inline assertions reaches *verification closure* when all reachable states (state of test bench with DUT) matching the contexts are traversed by the test bench simulation and then we terminate the simulation. The code of a test bench can also be visualized as a control flow graph (CFG) comprising of the sequence of conditional statements and the set of assignment statements. We define a control point and a state as follows:

Definition 1: **[Control Point ($\mathcal{CP}$):]** A control point is a location within the procedural source code where program control transfers when a conditional expression gets evaluated. □

Definition 2: **[State:]** A state at a particular simulation point (instance) is the values of a subset of state-variables from DUT and test bench. □

Instead of saving the values of all variables of a test bench as well as DUT at every $\mathcal{CP}$ as a *state*, we select a subset of state-variables. The following two kinds of state-variables are selected to represent a *state*.

- With respect to a test bench, these are the control path variables determining the control flow to the locations of inline assertions during simulation.
- With respect to DUT, the selected state-variables include the signals specified within an inline assertions along with other variables of DUT that affects those signals.

This improves efficiency by abstracting the saved state-space into a feasible one and thereby reduce time for loop detection (since less number of value-comparison done) and lessen termination overhead. The reason is that all other variables, except the participating ones in the *state*, do not contribute to the coverage of state-space for a test bench and so their values remain as don't care with respect to the *state*.

A. Approaches for Terminating Test Bench Simulation

The *overall methodology* described below summarizes the basic strategy for determining termination point of test bench simulation with respect to inline assertions. We use VPI-callbacks [5] to save states and to invoke loop-detection algorithm dynamically during simulation at each $\mathcal{CP}$.

Overall Methodology: **[Terminating Test benches:]**
Input : DUT module $\mathcal{J}$ + its test bench $\mathcal{TB}$ with inline assertions.
Output : Total simulation cycles *M* of $\mathcal{TB}$ when terminated and few related metrics indicating the computational effort.

Step 1: identify control points within $\mathcal{TB}$;
Step 2: determine state-variables;
Step 3: simulate test bench $\mathcal{TB}$ with DUT $\mathcal{J}$;
Step 4: Upon reaching $\mathcal{CP}$ for the first-time
save the *state* using VPI-callback;
Step 4: Upon reaching $\mathcal{CP}$ for subsequent times
invoke loop-detection procedure using VPI-callback;
Step 5: ***if*** loop detection SUCCESSFUL
5.1: terminate simulation;
5.2: report M & few related metrics;
Step 6: ***else*** continue to **Step 4**;

The crux of test bench termination methodology lies in the fact that we terminate the simulation of a test bench when all control paths of it traversed with all possible responses from DUT through repeated simulation runs and the last run is bound to be the exact replica of some previous run. Since, test bench and DUT together forms a deterministic closed system, so if a *state* repeats then the sequence between the repetition of that *state* will also repeat.

The main feature of the *overall methodology* lies in detecting the loop by comparing among the saved states through the invocation of loop-detection algorithm in *Step 4*. In the following sub-sections, we describe three different approaches for detecting loops which basically demonstrates three different test bench termination methodologies.

(a) Strategy-1: Approximate Global Loop Detection

In this approach, we do not distinguish among the control points while saving and comparing the states. Since we may detect loops while matching the states at different $\mathcal{CP}$ locations instead of matching them at a particular $\mathcal{CP}$ location similar to where the previous *state* is saved, we get *approximate* matching of states. Therefore, a loop may be detected falsely. However, experimental results reveal that such cases happen rarely.

(b) Strategy-2: Loop Detection Including Individual Control Point

Unlike approach-1 mentioned earlier, here we detect loop with respect to a particular $\mathcal{CP}$. The new *state* is checked with the saved old one only after reaching same $\mathcal{CP}$ location within pre-defined number of simulation/comparison cycles ($\mathcal{K}$). If the current number of cycles exceeds before finding loop or reaching the marked $\mathcal{CP}$, *state* at immediate next $\mathcal{CP}$ location is saved and the maximum cycle limit, $\mathcal{K}$ is doubled. The same procedure is repeated thereafter on reaching next $\mathcal{CP}$. The above mentioned approach compares a *state* saved at one particular $\mathcal{CP}$ with the new *state* saved when visiting that location next time. States at all other control points are ignored for comparison.

(c) Strategy-3: Loop Detection Including All Control Points

This approach extends the previous one by saving the states at every $\mathcal{CP}$. It is done by introducing maximum simulation/comparison check limit ($\mathcal{K}_i$) with each control point-*i* separately instead of a global limit. On reaching a particular $\mathcal{CP}$ for the second time, we try to detect loops with respect to the *state* saved on that $\mathcal{CP}$ before, provided the number of comparison does not exceed maximum simulation-cycles/comparison-checks. Upon exceeding the limit, we double the maximum limit with respect to the individual $\mathcal{CP}$.

This approach has two variants. In case of the first variant, we double $\mathcal{K}_i$ when the current simulation-cycle count of *i*-th $\mathcal{CP}$ exceeds the maximum simulation-cycles indicated by $\mathcal{K}_i$. Whereas for the second variant, we double $\mathcal{K}_i$ when current comparison-checks (instead of simulation-cycles) of *i*-th $\mathcal{CP}$ exceeds its $\mathcal{K}_i$.

B. Computational Effort Measurement Metrics

We introduce the following metrics to measure the computational effort required for each of these three methodologies proposed earlier.

- *Simulation-Cycle Count* – total number of clock cycles elapsed to detect state-loop.
- *Comparison-Cycles Count* – total number of cycles where two states get compared for state-loop detection.
- *Saved-States Count* – total number of distinct states needed to be saved before terminating simulation.
- *Compared-States Count* – total number of distinct saved-state pairs to be compared to detect state-loop.
- *State-Value-Changes Count* – number of times value change occurs in any state-variable of a *state*.

With the help of these metrics, we can compare the relative efficiency and mutual benefits of our proposed methods. In the next subsection, we compare the proposed methodologies.

C. Relative Study of Three Proposed Approaches

The proposed approaches are compared using the number of simulation/clock cycles, comparison-cycles count and total saved states required in each of these cases. Total clock cycles required for Strategy-1 will be minimum among all three strategies since it does some approximation during global loop detection. Between Strategy-2 and Strategy-3, Strategy-2 will have more clock-cycle count than Strategy-3, but lesser number of comparison-cycles count and saved states. The reason is, in Strategy-2 we save states at a particular $\mathcal{CP}$ and *state* comparison is done when the same $\mathcal{CP}$ is reached again. Whereas in Strategy-3, we save states corresponding to every $\mathcal{CP}$. When simulation reaches any of the control points, *state* comparison is done. Hence, the number of saved states is lesser and more number of states get compared in Strategy-3. Moreover, due to more comparisons, Strategy-3 detects loop early in terms of simulation-cycles than in Strategy-2.

IV. Implementation and Results

We use Verilog HDL [4] to describe the design and the test bench with inline assertion is written in System Verilog [6]. We have used VCS [8] from Synopsys to simulate the test bench over the design. Moreover, to formally verify the inline assertions, we have used VIS tool [10] as a formal verifier.

A. Implementation Details

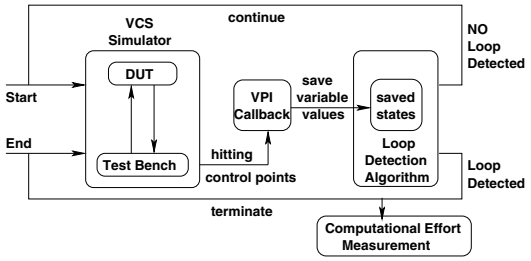


Fig. 4. VCS-based Implementation Architecture for Verification Closure

We instrument the test bench code by inserting VPI-callbacks [5] at each $\mathcal{CP}$. The *state* variables are computed a priori. During simulation, at each $\mathcal{CP}$, these VPI-callbacks are used to dynamically save the states and simultaneously execute the loop-detection algorithm to detect repeating states. Finally, we either terminate the simulation encountering a state-loop or else, continue it. At the simulation termination point, we also report several metrics determining computational efficiency of our three approaches. Figure 4 shows the overall architecture of our *verification closure* strategy.

We continue simulating the test bench over the DUT till we reach the targeted inline assertion. Upon reaching the target context, the *state* (values of the state-variables) is saved to form a reset-state to be used in the next phase. In the subsequent phase, this saved *state* is utilized as a initial state by VIS to reset the DUT, from where we trigger the BMC routine of VIS without rebuilding the network (as in the usual case). Figure 5 shows the overall *semi-formal verification* architecture of our approach.

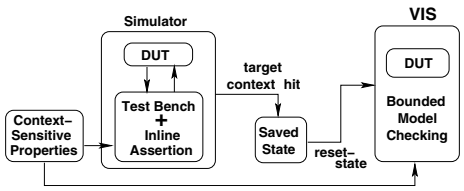


Fig. 5. Implementing VIS-based Semi-Formal Verification Architecture

B. Experimental Results

We have used the PCI Bridge and Ethernet examples from OpenCores verification benchmarks [9] and also PCI circuits from Texas97 benchmarks [7] as our case study. We have modified the test benches of the mentioned benchmarks by inserting inline assertions. All experiments are performed on a 2.8 GHz. Processor of INTEL(R) XEON(TM) machine with 4 GB RAM.

Table I shows the result of our proposed *verification closure* methodologies over the benchmarks. Column 1 indicates the type of benchmark circuit/module and column 2-6 presents design statistics. Column 7 shows three different strategies. Columns 8-12 denote the relevant metrics that are defined to measure the computational efficiency in for each proposed strategy to terminate simulation. These metrics are simulation-cycles count, comparison-cycles count, saved-states count, compared-states count and state-value-changes count as indicated through columns 8-12 respectively. From Table I, we see that the values of the metrics for measuring computational efficiency matches as per our discussion in section III-C.

In Table II, we show the capacity blowups and inability of pure formal verification in VIS to validate a property triggered at deep context-satisfying states; which is easily handled (with less memory requirement) by our semi-formal approach. Column 1 denotes two of our written properties from the target and the master modules of PCI circuit of Texas-97 benchmarks [7]. Columns 2 to 6 present design statistics extracted by VIS. Column 7 shows the set of properties for the respective module. Columns 8 and 9 show the memory requirements (in MB) and time needed (in sec.) respectively to formally verify an inline property from the initial states of the design. Finally, columns 10 and 11 denote the same by our semi-formal methodology. The time indicated in column 11 is the total (summed) time required to formally verify an inline property (using BMC) from all its satisfying contexts reachable by the test bench and the memory usage is the average memory required during these phases. Column 8

Circuit Module	No. of Inputs	No. of Outputs	No. of Seq. Elem.	No. of Comb. Elem.	No. of Nets	Strategy	Count of Relevant Metrics (Measuring Computational Effort)				
							Simulation Cycles	Comparisons	Saved States	Compared States	State-Value Changes
PCI Bridge (OpenCores)	162	207	3802	26424	30388	Strategy-1	56848	50888	31	2	100245
						Strategy-2	5856098	4407860	22	35	10121400
						Strategy-3a	2641620	9219686	61	58	4553203
						Strategy-3b	2641620	7578812	61	58	4553203
Ethernet Circuit (OpenCores)	96	115	10545	47296	57937	Strategy-1	358977	66	118	10	242
						Strategy-2	359119	176	7	18	258
						Strategy-3a	358977	184	10	9	242
						Strategy-3b	358977	182	10	9	242
PCI Target (Texas97)	12	8	156	3071	7038	Strategy-1	3217975	3217910	114	864570	2799413
						Strategy-2	4691067	4690972	20	53938	2601697
						Strategy-3a	3257383	8171724	14383	14381	350562
						Strategy-3b	3257383	7557364	14383	14381	350562
PCI Master (Texas97)	12	11	253	3685	8862	Strategy-1	3045615	3044542	90	46185	830346
						Strategy-2	4522849	4522532	23	34270	1233107
						Strategy-3a	3170465	6923146	1152	987	321401
						Strategy-3b	3170465	7210127	1152	987	321401

TABLE I

VERIFICATION CLOSURE AND COMPUTATIONAL EFFICIENCY MEASUREMENT OVER FEW BENCHMARKS

Ckt. Modules	No. of Inputs	No. of Outputs	No. of Seq. Elem.	No. of Comb. Elem.	No. of Nets	Prop	VIS Pure Formal Method		Our Semi-formal Approach	
							Memory (MB)	Time (Sec)	Memory (MB)	Time (Sec)
PCI Target	12	8	156	3071	7038	P1	342	timeout	0.094	853.76
						P2	306	timeout	0.091	694.28
PCI Master	12	11	253	3685	8862	P1	296	timeout	0.092	751.24
						P2	284	timeout	0.089	712.84

TABLE II

COMPARISONS OF MEMORY USAGE & TIME OF PCI MODULES FROM TEXAS97 BENCHMARKS

shows the greater memory requirement of VIS when the timeout is set to two minutes. We have further experimented with higher timeout values (in order of hours). Even with higher timeout values, VIS fails to detect counter-example. We have observed that for higher timeout values, the memory requirement increases drastically.

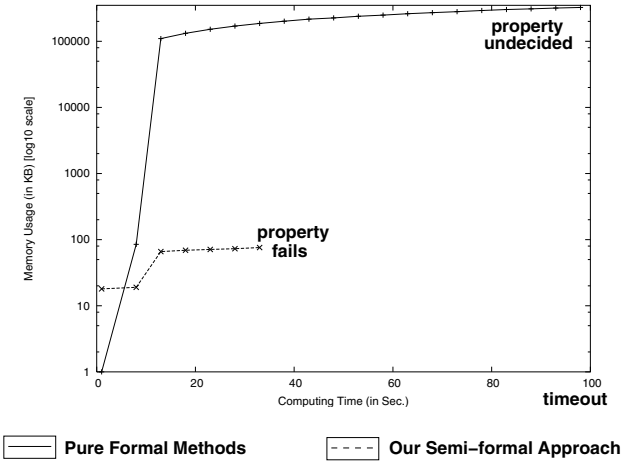


Fig. 6. Dynamic Memory Usage Graph

We have also plotted a dynamic memory usage graph as shown in Figure 6 for a property from PCI target module. It reveals the efficiency of our approach over pure formal methods, which essentially blows up (in capacity) to verify a property. In the graph shown above, we have found how memory requirement blows up when a property is run for 100 seconds by a pure formal engine and yet the property remains undecided. On the other hand, our semi-formal approach first simulate up to a satisfying context in 8.22 sec. and then invoking the BMC engine of VIS tool, we can prove the failure in 24.65 sec. (total in 32.87 sec.) for the same inline property.

REFERENCES

[1] Biere A., Cimatti A., Clarke E. M., Strichman O., Zhu Y.; Bounded Model Checking, In Journal of Advances in Computers, vol. 58, pp. 118149, 2003.

[2] Clarke, E. M., Grumberg, O., and Peled, D. A.; Model Checking. MIT Press, 2000.

[3] Dasgupta P.; A Roadmap for Formal Property Verification. Springer, 2006.

[4] IEEE 1364-2001 Standard Verilog Hardware Description Language; http://ieeexplore.ieee.org/iel5/7578/20656/00954909.pdf?arnumber=954909

[5] Sutherland S.; The Verilog PLI Handbook, Second Edition, Kluwer Academic Publishers, 2002.

[6] System Verilog from Accellera; http://www.systemverilog.org/

[7] Texas97 Benchmarks; http://www-cad.eecs.berkeley.edu/Respep/Research/vis/texas-97/

[8] VCS Simulator from Synopsys; http://www.synopsys.com/vcs/

[9] Verification Benchmarks from OpenCores; http://www.opencores.org/

[10] VIS homepage; http://vlsi.colorado.edu/vis

2009 22nd International Conference on VLSI Design

Dedicated Rewriting: Automatic Verification of Low Power Transformations in RTL

Vinod Viswanath
Computer Engineering Research Center
University of Texas at Austin
vinod@cerc.utexas.edu

Shobha Vasudevan
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
shobhav@uiuc.edu

Jacob A. Abraham
Computer Engineering Research Center
University of Texas at Austin
jaa@cerc.utexas.edu

Abstract

We present dedicated rewriting, a novel technique to automatically prove the correctness of low power transformations in hardware systems described at the Register Transfer Level (RTL). We guarantee the correctness of any low power transformation by providing a functional equivalence proof of the hardware design before and after the transformation. We characterize low power transformations as rules, within our system. Dedicated rewriting is a highly automated deductive verification technique specially honed for proving correctness of low power transformations. We provide a notion of equivalence and establish the equivalence proof within our dedicated rewriting system. We demonstrate our technique on a non-trivial case study. We show equivalence of a Verilog RTL implementation of a Viterbi decoder, a component of the DRM SoC, before and after the application of multiple low power transformations.

1 Introduction

In today's electronic systems, power is one of the most critical design parameters, especially in portable computing and personal communication applications such as cell phones, PDAs, and the emerging Mobile Internet Devices (MID) market. Even in the case of stationary systems, power-aware designs can offer competitive advantages regarding considerations of size and cost of power supply and cooling systems. In the case of today's extremely large and complex designs, implementing a reliable power network and minimizing power dissipation have become major challenges for design teams. Employing low power transformations in existing designs has emerged as an extremely important technique in achieving the power goals of a design.

Utilizing the program structure or the dataflow information available at the architectural and RTL levels can lead to many interesting and complicated low power transformations [4], [6], [16]. Yet most power transformations in today's designs are at the gate-level [2], [13]. The primary reason for this is the difficulty of the verification problem. If low power transformations are at the gate level, then equivalence checking methods [12] can be used to automatically prove the correctness of the transformations. Although there are a few sequential equivalence checking algorithms [10], [14], they are not widespread in the hardware industry. In a typical industry scenario, an RTL or architectural low power transformation implies a full cost of dynamic validation, which can extend to many months, and require a lot of resources.

We present *dedicated rewriting*, a rewriting methodology to automatically prove the correctness of low power transformations at the RT-level. We propose a highly automated deductive verification technique which is fine-tuned for low power transformations. We prove the equivalence of two Verilog RTL designs, one derived from the other after the application of a low power transformation. Our notion of equivalence is defined with respect to an observable. An observable is a variable in the Verilog RTL that, by specification, is expected to have the same function in both RTLs. The inputs to our system are the two RTLs and a set of observables.

The computation model in our technique involves Term Rewriting Systems (TRSs) [11]. A TRS is defined as a tuple of terms and rules. Our technique starts by automatically deriving a TRS from a given Verilog RTL. The variables in the Verilog RTL form the terms of the TRS. The rules of the TRS represent the hardware design and are both time-preserving and atomic. The time-preserving nature of the rules guarantees that both explicit and implicit timing dependencies in Verilog RTL are captured in the rules. Atomic transactions ensure that a given rule is executed in its entirety without interruption or conflict with rest of the system. We derive TRSs from both Verilog RTLs, before and after applying the low power transformation. We have defined a notion of equivalence of two TRSs with respect to an observable (a term in both TRSs). We now proceed to prove the equivalence of the two TRSs with respect to the observable. In the process, we create a dedicated database of low power transformation rules. If a proof cannot be established, then we either add a new rule to the database, or we have found a bug. This process is repeated for all observables.

We present the result of our technique on different low power transformations applied to a Verilog RTL implementation of Viterbi decoder [17] module that is a part of the Digital Radio Mondiale (DRM) SoC [1]. The main contributions of this work are the following.

- We present a methodology to automatically verify correctness of applying a low power transformation on existing hardware, thereby drastically reducing design cycle time.
- We present dedicated rewriting, an automatic and dedicated prover for low power transformations in RTL. There is minimum overhead of providing environment and additional lemmas as opposed to a general purpose rewriting engine.
- We present a novel notion of capturing the functional and timing description of RTL in the form of atomic transactions called rules.
- We have created a dedicated database of low power transformation rules.
- We leverage the expressive power and relative simplicity of high-level designs by reasoning entirely at that level.
- We demonstrate our technique by proving the correctness of multiple low power transformations on a real life SoC RTL.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.85

```
input clk;
input pgate_signal;

always @(posedge clk) begin


 case (sw)
   0:  butterfly_1 = fifo[11:8] + gsm[23:20];
   1:  butterfly_1 = fifo[7:4] + gsm[19:16];
   default:  butterfly_1 = fifo[3:0] + gsm[15:12];
 endcase
end
```

(a) Pre-transformation Verilog RTL.

```
input clk;
input pgate_signal;
assign gated_clk = clk & ~pgate_signal;
always @(posedge gated_clk) begin
 if (c3_state) butterfly_1 = fifo[3:0] + gsm[15:12];
 else
   case (sw)
     0:  butterfly_1 = fifo[11:8] + gsm[23:20];
     1:  butterfly_1 = fifo[7:4] + gsm[19:16];
     default:  butterfly_1 = fifo[3:0] + gsm[15:12];
   endcase
end
```

(b) Post-transformation Verilog RTL.

```
Rule:  butterfly_1(t) ⟶ fifo[11:8](t-1) + gsm[23:20](t-1)
       if (sw(t-1) == 0) and (posedge_clk(t-1))
Rule:  butterfly_1(t) ⟶ fifo[7:4](t-1) + gsm[19:16](t-1)
       if (sw(t-1) == 1) and (posedge_clk(t-1))
Rule:  butterfly_1(t) ⟶ fifo[3:0](t-1) + gsm[15:12](t-1)
       if ((sw(t-1)!=0) and (sw(t-1)!=1)) and (posedge_clk(t-1))
```

(c) Pre-transformation TRS Rules.

```
Rule:  gated_clk(t) ⟶ (clk(t) & ~pgate_signal(t))
       if (T)
Rule:  butterfly_1(t) ⟶ fifo[3:0](t-1) + gsm[15:12](t-1)
       if (c3_state(t-1)==T)
Rule:  butterfly_1(t) ⟶ fifo[11:8](t-1) + gsm[23:20](t-1)
       if (sw(t-1)==0) and (c3_state(t-1)!=T) and (posedge_gated_clk(t-1))
Rule:  butterfly_1(t) ⟶ fifo[7:4](t-1) + gsm[19:16](t-1)
       if (sw(t-1)==1) and (c3_state(t-1)!=T) and (posedge_gated_clk(t-1))
Rule:  butterfly_1(t) ⟶ fifo[3:0](t-1) + gsm[15:12](t-1)
       if ((sw(t-1)!=0) and (sw(t-1)!=1)) and (c3_state(t-1)!=T)
         and (posedge_gated_clk(t-1))
```

(d) Post-transformation TRS Rules.

Figure 1. Clock gating for lowering switching activity power dissipation.

Initial application of Term Rewriting Systems in verification was in program verification [3], [7]. Subsequently they have been used to design correct circuits [9], and verifying them [5], [15], [18] in the context hardware design. In sharp contradistinction to prior work in this area, this paper presents a new technique in a completely different hardware context of low power transformations at the RT-level.

We explain our notion of rules in full detail in Section 2. Section 3 describes our dedicated rewriting methodology. In Section 4 we present a case study of using our technique on multiple low power transformations on the Viterbi decoder. We discuss the merits of our technique and conclude in Section 5.

2 Rules

A Term Rewriting System is defined as a tuple $\langle S, R \rangle$, where S is a set of terms, and R is a set of rewriting rules. The state of a system is represented as a TRS term, while the state transitions are represented as TRS rules. The general structure of rewriting rules is:

Rule: $s_1 \longrightarrow s_2$ if $p(s_1)$

where s_1 and s_2 are terms and p is a predicate. We can use a rule to rewrite a term if the rule's left-hand-side pattern matches the term or one of its sub-terms, and the corresponding predicate is true. The new term is generated in accordance with the rule's right-hand side. If several rules apply, then any one of them can be applied. If no rule applies, the term cannot be rewritten any further.

Applying a rule is also called executing or firing. When a rule is applied, the state on the left-hand-side is read at the beginning of the clock cycle and updated at the end of the clock cycle. This single cycle notion automatically enforces the atomicity constraint of each rule. All enabled rules fire in parallel (or in no particular order). If two rules modify the same state element, then we have a race condition. We expect the Verilog RTL to be race-free and combinational-loop-free at the input. This is an easy constraint to impose on the input Verilog, since it can checked by standard Verilog linting tools.

We have two kinds of rules within our system, structural rules and logical rules. Structural rules are timing preserving atomic transactions representing a state update in hardware. These are derived directly from the Verilog RTL. Logical rules represent identities about the RTL operators and carry information about the low power transformations. We explain these further in the next two subsections.

2.1 Structural Rules

Consider the example in Figure 1. Verilog RTL prior to a clock gating transformation is shown in Figure 1(a) and the post transformation RTL is shown in Figure 1(b). When we derive the TRS from the Verilog RTL, we arrive at the structural rules as shown in Figure 1(c) and Figure 1(d).

Each hierarchical signal in Verilog is represented by a new constant function symbol (*signal function*), thereby creating a "flattened" TRS. Rewrite rules rewrite each signal function into an expression consisting of RTL operators and other signal functions. The cycle-accuracy of the RTL semantics is maintained in the TRS by each signal function being a function of time t. The notion of time is relative. A combinational logic assignment in the Verilog is a rewrite rule with all terms being a function of the same time t (the first rule in Figure 1(d)). Whereas, a sequential logic assignment in the Verilog is a rewrite rule with the assigned term being a function of time t and all other terms relatively 1 cycle before the assigned term, as a function of time $(t-1)$ (all other rules in Figure 1(c) and Figure 1(d)).

2.2 Logical Rules

The logical rules codify identities about RTL operators and various low power transformations. This is an independent database that is part of our dedicated rewriting system. These rules can be generic or low power transformation specific. Some examples of the generic rules in this database are shown in Figure 2.

```
Rule:  ( x & x ) ⟶ ( x ) if (T)
Rule:  ( x & ~x ) ⟶ ( F ) if (T)
Rule:  ( (x << 1) - x ) ⟶ ( x ) if (T)
Rule:  ( x + y ) ⟶ ( y + x ) if (T)
Rule:  ( x & (y|z) ) ⟶ ( (x&y) | (x&z) ) if (T)
Rule:  ( (x << 2) - x ) ⟶ ( x * 3 ) if (T)
```

Figure 2. Logical rules in the dedicated rule database.

Generic rules define RTL operator identities and are helpful in simplification of terms during the equivalence proof. We also do not restrict the level of abstraction of the input RTL. This structure in our system allows us to, for example, use a new (or non-synthesizable) RTL operator in the input Verilogs, and define the identities of that operator within this rule database to allow for simplification during the proof process. Typically, the predicate function of these generic logical rules is always true (`T`).

The low power transformation specific rules define the identities associated with the transformation. These will be different for each transformation. However, these are independent of the RTL on which the transformation will be applied, and as such need to be incorporated as part of this database exactly once, for each transformation.

Consider the clock gating example in Figure 1. It shows the Verilog RTL and the derived TRS rules before and after the application of the transformation to achieve lower switching activity power dissipation. In this example, the transformation creates a new clock gated by the signal `pgate_signal`. As shown in Figure 1(d) two new rules corresponding to the creation of the gated clock are added to the list of rules. In this example, apart from these structural rules, two new, transformation specific rules are added to the logical rule database:

- An external assumption when to enable power clock gating decides the value of the signal `pgate_signal`. The enabling assumption needs to be codified into the transformation specific rules database as follows:

  ```
  Rule: ( pgate_signal ) ⟶ T if (T)
  ```

  If we were to run our proof with power clock gating disabled, then we would add the corresponding rule, setting the signal to `F`.

- The algorithm of power gating has the hardware assume a special state (`c3_state`) when the transformation is enabled. This information is captured in a transformation specific rule as follows:

  ```
  Rule: ( c3_state ) ⟶ T if (pgate_signal==T)
  ```

  Assumptions of this nature which are very specific to the low power transformation need to be specially encoded as rules in order to assist the proof system.

Rules are powerful representations of always blocks. The active rules, where the guards (predicates) are true, can be applied in parallel, but each rule operates as an atomic transaction, *i.e.*, each rule observes and ensures a consistent state relative to all other rules in the system.

3 Dedicated Rewriting

We propose a refinement based rewriting methodology to automatically generate proofs for low power transformations in RTL. Figure 3 gives a flow chart representation of our proof methodology.

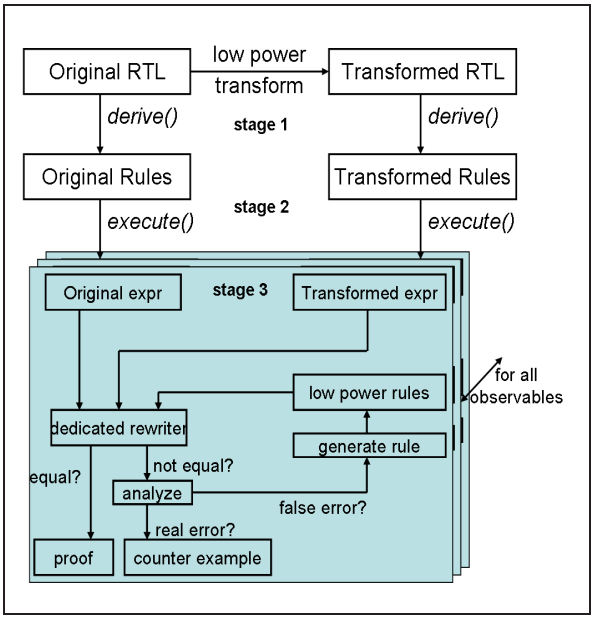


Figure 3. Dedicated rewriting proof system flow chart.

The input to the system are two RTLs, an original RTL and a transformed RTL (after the application of the low power transformation), and a set of observables. The proof methodology works in three primary stages. First, we derive the structural rules of the TRS from the Verilog RTL for both models. Next, we execute

the rules to derive expressions for all observables in each model. Finally, for each observable, we go through an iterative, mostly automated proof process. These are labeled as **stage 1**, **stage 2**, and **stage 3** in Figure 3. Figure 4 gives the algorithm for the dedicated rewriting procedure. We describe the procedure stage by stage by elaborating on the functions involved in each stage in the rest of this section.

3.1 *derive ()*: Verilog RTL to TRS rules

This function translates Verilog RTL to TRS rules. As we described in Section 2, rules are atomic transactions carrying timing information. The *derive ()* function will generate the structural rules of the TRS. We have fully automated this translation process. Examples of this automated translation in different contexts are shown in Figure 1. Combinational logic statements in the RTL get translated to rules with all terms at the same relative time t, whereas, sequential logic statements get translated to different relative times. We do this translation on both models.

The fully automatic nature of this translation allows us to use our methodology on any existing Verilog RTL design. This is particularly useful in the context of verification of low power transformations since most of these transformations tend to be RTL changes late in the design stage in order meet the power requirements of the landing zone.

3.2 *execute ()*: TRS rules to expressions

This function computes the expression of a particular observable by rewriting the structural TRS rules. The process of rewriting is based on firing the rules that are ready to fire. If more than one rule can fire at any given time (clock cycle), then all the rules fire in no particular order. Since our rules are strictly atomic transactions this maintains the correctness of the Verilog semantics. At the end of this process, we have essentially computed the symbolic expression of the observable. We do this in tandem on both models for each observable in the input set of observables. The next subsection explains how the equivalence of the two expressions in the two models, for every observable, is proved.

Depending on the design we are working with, these expressions can be arbitrarily large, thereby making the equivalence proof arbitrarily hard. In order to mitigate that problem, we use some heuristics to reduce the complexity of the expressions. One heuristic we have successfully used in the context of arithmetic circuits is a bit-wise partitioning of assignments to a particular RTL signal. If we have matching bit partitions aligned in both models, then the expression corresponding to that bit partition is treated as the basis for the equivalence proof between the models.

This procedure is also completely automated and includes the above decomposition heuristic. We have structured our methodology such that it can accommodate any decomposition strategy. Our tool is designed such that we can easily and seamlessly incorporate any library of decomposition heuristics in our routine that calculates expressions by rewriting structural rules.

3.3 *prove ()*: Equivalence of expressions

In this function we check the equivalence of two expressions by rewriting based on simplification using the logical rules from the dedicated rule database. As described in Subsection 2.2 these logical rules codify various identities about RTL operators, as well as rules specific to the low power transformation.

These rules can be generic or design specific or transformation specific. While trying to prove the equivalence of two expressions, we select the set of rules ready to fire from the rule database and apply them in some arbitrary order. If the resulting simplifications fail to establish the equivalence, then the rules are applied in a different order. These *proof iterations* continue until equivalence is established or no more rules can be applied in any order. This step is executed in the function *dedicated_rewrite ()*. These iterations are guaranteed to terminate since we are working with finite bit-width hardware signals, and in the worst case, all possible value-combinations of all variables is explored over a finite (*albeit* large) number of iterations.

When two designs are declared unequal by this technique, the eponymous function *analyze ()* analyzes the result and discovers one of two possibilities. In the first case, the two designs are truly not equal, in which case we have caught a "bug" in the transformation, and our technique provides a detailed proof trace and also picks a counterexample starting from the inputs at specific times. In the second case, we may not have sufficient rules in the dedicated rule database. In this case we allow for user intervention to create and add the required rules to the database (this is handled in the function *generate_and_add ()*), and the entire process can be repeated again. The shaded part of Figure 3 (**stage 3**) describes this process. This entire process is repeated for every pair of expressions that need to be proved equivalent.

Algorithm Dedicated Rewriting.

main (V_o**: Original RTL model,** V_p**: Transformed RTL model,** O**: Set of observables,** $DRdb$**: dedicated rule database)**

```
begin
   proved = T
   T_o = derive (V_o)
   T_p = derive (V_p)
   for every observable o ∈ O
   begin
      {⟨expr_1, expr_2⟩} = execute (T_o, T_p, o)
   end
   for every pair of expressions ⟨expr_1, expr_2⟩
   begin
      proved = proved && (prove(expr_1, expr2, DRdb))
   end
   return (proved)
end
```

prove ($expr_1$**: Original expression,** $expr_2$**: Transformed expression,** $DRdb$**: dedicated rule database)**

```
begin
   do
   begin
      out = dedicated_rewrite (expr_1, expr_2, DRdb)
      if (out == T) return (T)
      else
      begin
         true_error = analyze ()
         if (true_error == T)
            return (counter_example)
         else
            generate_and_add (DRdb)
      end
   end
   while (out==F && true_error==F)
end
```

Figure 4. Dedicated rewriting algorithm

Figure 4 gives the full algorithm for the dedicated rewriting

process we described in this section. Once we establish the equivalence of two TRSs with respect to a given set of observables, in effect, we have proved he correctness of the low power transformation which created the transformed RTL from the original RTL. In this process, we have established the logical transformation specific rewrite rules as part of our dedicated rule database. Therefore, all further proofs of application of the same transformation on any other RTL design should be possible with minimal user intervention or changes to the dedicated database. This is what separates dedicated rewriting from a general purpose rewriting engine or a theorem prover, and makes the technique a highly automated approach.

4 Case Study: Viterbi Decoder

We perform our experiments on a Viterbi decoder, that is a part of the Digital Radio Mondiale (DRM), implemented in Verilog RTL. The initial Viterbi decoder RTL is a basic model that implements the Viterbi decoding algorithm, but has no optimizations for power, area, or performance. On this model, we perform certain low power transformations aimed at reducing the switching activity power dissipation. These transformations do not cross register boundaries. We then use our framework to prove that these transformations do not affect the functionality of the original design.

Next, we use a more complicated Viterbi decoder design, also implemented in Verilog RTL, but one optimized for lower power dissipation. In contrast to the earlier transformations, this transformation crosses register boundaries. We estimate the power dissipation savings and also prove the equivalence of the two sequential designs.

Finally, we use a third Viterbi decoder design, also implemented in Verilog RTL, but optimized for a combination of power and timing. This is more realistic in today's industry where every functional unit block in the design has specific area, power, and timing budgets, and all optimizations have to meet an optimal landing zone satisfying all the three requirements. Again, we prove the equivalence of the two sequential designs, the optimized design and the original Viterbi design by dedicated rewriting. These automatically generated correctness proofs are the contribution of our system to this problem, and not the quality of power savings generated by these specific transformations.

4.1 Combinational low power transformations

There are two major stages to the functionality of the Viterbi decoder. One is collecting the inputs depending on the Puncture Pattern and storing them in a buffer (FF Buffer). The other stage is the Trellis computation. The next state values of the Trellis matrix are computed by a function (Butterfly network) of current state values of the Trellis matrix and the inputs stored in the FF Buffer.

We perform combinational low power optimization in the logic computing the values of the Trellis matrix. The low power transformations performed include common subexpression elimination, movement of operations, constant propagation and commutativity and associativity based optimizations. All these optimizations were strictly between register boundaries of the design. Hence, the symbolic terms at every comparison point were at the same relative times in both models. We used the macro-modeling technique employed in [8] to estimate power at the RT-level of abstraction. This method involved characterizing the power consumed by the basic combinational blocks (macros) used in the design. We did the characterization for the all macros in the `GetMatrixSet` block of the Viterbi decoder design. All our optimizations were also in the same block.

Optimizations used	Estimated power (mW)
Nil	143
Common sub-expression reduction-1	112
Common sub-expression reduction-2	105
Constant propagation	104
Commutative rearrangement	127
Associative rearrangement	121

Figure 5. Results of power estimation due to combinational logic low power transformations in the Viterbi decoder.

Design Configuration	Estimated power (mW)
Original Viterbi, clock_delay=10ns	416.37
Transformed Viterbi, clock_delay=10ns	354.33
Original Viterbi, clock_delay=20ns	208.41
Transformed Viterbi, clock_delay=20ns	177.17

Figure 6. Results of power estimation after sequential low power transformations in the Viterbi decoder.

The table in Figure 5 shows the power estimation results for these optimizations. The first column in the table denotes the transformations used to optimize the design. The entry "Nil" corresponds to the base design before optimization. The power estimated (not including glitch power) after applying each transformation sequentially is listed in the second column. This estimated power is only for the `GetMatrixSet` block.

4.2 Sequential low-power optimizations

The sequential optimizations for low-power included clock-gating, some functional gating, and reorganizing pipeline registers. The table in Figure 6 shows the power estimation before and after the low-power transformation. These numbers were obtained by using the Artisan TSMC 0.18um library. The Verilog designs were synthesized using Synopsys Design Compiler and the power estimated using Synopsys Power Compiler. The activity factors were kept as default, which is 1 per cycle for input signals and 2 per cycle for the clock signal. We show the power estimation numbers for a clock delay of 10ns and 20ns.

4.3 Optimizations for power and timing

These optimizations were similar to the sequential lower power optimizations except that some aggressive power saving was sacrificed to lower the critical path delay time of the design. The power estimation methodology was the same as in the case of the sequential optimizations and the estimated dynamic power dissipation is very close to the previous case. The table in Figure 7 shows the results.

4.4 Correctness of low power transformations on the Viterbi decoder

We start with the basic Viterbi decoder design, and three transformed designs, and construct the equivalence proof of each of

Design Configuration	Estimated power (mW)
Original Viterbi, clock_delay=10ns	416.37
Transformed Viterbi, clock_delay=10ns	358.06
Original Viterbi, clock_delay=20ns	208.41
Transformed Viterbi, clock_delay=20ns	179.04

Figure 7. Results of power estimation after sequential optimization for low power and timing in the Viterbi decoder.

the three transformed designs against the basic design. In this subsection, we outline the proof of equivalence of the sequential low power transformation using our technique. We have four observables in this design, `8 FIFO entries`, `64 Trellis Matrix entries`, `2 entries in the MatDecTable`, and a `decoded output`. We build the proof of equivalence for each observable. In accordance with our algorithm in Figure 4 we follow the three stages. We first translate both designs into TRSs. Next, We identify and obtain the expressions that need to be proved equivalent for each observable. Finally, we use the dedicated rule database, and prove the equivalence of the expressions by rewriting. We had to add several rules to the dedicated rule database while we proved the correctness of this transformation. The key insight here is that the transformations once proved on any RTL, yield the same set of logical rules for any new RTL. This is in a way similar to thinking of proving the correctness of the transformation irrespective of the RTL context (although new RTL contexts themselves might throw up new logical rules, independent of the transformation). Similarly, we derive correctness proofs for the transformations in each design.

5 Discussion and Conclusions

We have presented dedicated rewriting, a novel technique for proving correctness of low power transformations in RTL. Our technique uses Term Rewriting Systems and decomposes the problem into smaller and tractable proofs. The deductive nature of the rewriting system ensures that we do not encounter any size or capacity issues that are common in an algorithmic sequential formal verification framework. However, the complexity of the verification problem in our system is converted into the problem of incompleteness of the dedicated rule database. While this might lead to a highly interactive system in the general case, restricting the rule database to low power transformations, helps us leverage a higher degree of automation. Once the database captures a transformation (in the form of rules), then those rules work for any RTL and have high reuse value. The primary advantage of this technique is that we reason with uninterpreted operators and bit abstractions at the RT-level, which is decidedly a more abstract (and therefore smaller) representation than reasoning at the gate-level. In this paper we have described the technique in the context of a real life SoC, the Viterbi decoder. We have also proved complicated low power transformation on pipelined microprocessors using the same dedicated rewriting system. This work gives a specialized rewriter in the context of low power transformations. Given the cost of re-validating hardware systems in a traditional design cycle, an automated technique of this nature is extremely desirable and can add immense value to the hardware design cycle.

References

[1] Digital Radio Mondiale *http://www.drm.org/*.

[2] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou. Precomputation-based sequential logic optimization for low power. *IEEE Trans. Very Large Scale Integr. Syst.*, 2(4):426–436, 1994.

[3] S. Antoy and J. Gannon. Using term rewriting to verify software. *IEEE Trans. Softw. Eng.*, 20(4):259–274, 1994.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th annual international symposium on Computer architecture*, pages 83–94. ACM Press, 2000.

[5] M. S. Chandrasekhar, J. P. Privitera, and K. W. Conradt. Application of term rewriting techniques to hardware design verification. In *24th ACM/IEEE conference proceedings on Design automation conference*, pages 277–282, 1987.

[6] D. Chaver, L. Piñuel, M. Prieto, F. Tirado, and M. C. Huang. Branch prediction on demand: an energy-efficient solution. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 390–395. ACM Press, 2003.

[7] T. Genet and F. Klay. Rewriting for cryptographic protocol verification. In *Conference on Automated Deduction*, pages 271–290, 2000.

[8] S. Gupta and F. N. Najm. Power macro-models for dsp blocks with application to high-level synthesis. In *ISLPED '99: Proceedings of the 1999 international symposium on Low power electronics and design*, pages 103–105, New York, NY, USA, 1999. ACM.

[9] J. Hoe and Arvind. Hardware synthesis from term rewriting systems. In *X IFIP International Conference on VLSI (VLSI 99)*, Lisbon, Portugal, November 1999.

[10] S.-Y. Huang, K.-T. Cheng, and K.-C. Chen. Verifying sequential equivalence using atpg techniques. *ACM Trans. Des. Autom. Electron. Syst.*, pages 244–275, 2001.

[11] J. Klop. Term Rewriting Systems. In *In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors: Handbook of Logik in Computer Science, Oxford University Press*, volume 2, pages 1–116, 1992.

[12] Y. Matsunaga. An Efficient Equivalence Checker for Combinational Circuits. In *Proceedings of Design Automation Conference*, pages 629–634, 1996.

[13] V. Tiwari, S. Malik, and P. Ashar. Guarded evaluation: pushing power management to logic synthesis/design. In *Proceedings of the 1995 international symposium on Low power design*, pages 221–226. ACM Press, 1995.

[14] S. Vasudevan, V. Viswanath, J. A. Abraham, and J. Tu. Automatic decomposition for sequential equivalence checking of system level and rtl descriptions. In *Proceedings of IEEE International Conference on Formal Methods and Models for Co-Design (MEMOCODE 2006)*, pages 71–80, 2006.

[15] S. Vasudevan, V. Viswanath, R. W. Sumners, and J. A. Abraham. Automatic verification of arithmetic circuits in rtl using stepwise refinement of term rewriting systems. *IEEE Trans. Comput.*, 56(10):1401–1414, 2007.

[16] V. Viswanath, J. A. Abraham, and Warren A. Hunt, Jr. Automatic insertion of low power annotations in rtl for pipelined microprocessors. In *DATE '06: Proceedings of the conference on Design, automation and test in Europe*, pages 496–501, 2006.

[17] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. In *IEEE Transactions on Information Theory*, pages 260–269, 1967.

[18] Z. Zhou and W. Burleson. Equivalence checking of datapaths based on canonical arithmetic expressions. In *Proceedings of the 32nd ACM/IEEE conference on Design automation conference*, pages 546–551. ACM Press, 1995.

Session 1C

Fault Diagnosis

2009 22nd International Conference on VLSI Design

A Novel Approach for Improving the Quality of Open Fault Diagnosis

Koji Yamazaki
Toshiyuki Tsutsumi
Meiji University

Hiroshi Takahashi
Yoshinobu Higami
Takashi Aikyo
Yuzo Takamatsu
Ehime University

Hiroyuki Yotsuyanagi
Masaki Hashizume
The University of Tokushima

Abstract

With the shrinking process technologies and the use of copper process, open defects on interconnect wires, contacts and vias often cause failure. Development of an efficient fault diagnosis method for open faults is desired. However, the diagnosis method for open faults has not been established yet. In this paper, we propose a novel approach for improving the diagnostic quality of open faults by introducing a threshold function in which the logical value of the line with open defect depends on the weighted logical values of its adjacent lines. By using the threshold function, we can deduce not only a faulty line but also an open defect site at the faulty line. Experimental results show that the proposed method can identify an exact faulty line in most cases with a very small computation cost. The proposed method can also identify the open defect site within 25%-length of the faulty line.

1. Introduction

In order to reduce the cost and to improve the yield, it is important to develop an efficient fault diagnosis method. With the shrinking process technologies and the use of copper process, open defects on interconnect wires, contacts and vias often cause failure. Such defects are modeled as open faults. However, the diagnosis method for open faults has not been established yet. Therefore, development of an efficient fault diagnosis method for open faults is desired strongly.

Some open fault models have been proposed [1-5]. In the fault model of [1], the line with open fault has an intermediate voltage, and the faulty line behaves as stuck-at 0 or 1, depending on the threshold voltage of the gate with the faulty line in its input. Some fault diagnosis methods for this model have been proposed [7-10]. In the fault model described in [2-5], the voltage of a line with open defect is determined by the voltage of its adjacent lines. In this open fault model, the value of the faulty line is represented as a logic function of its adjacent lines.

In [3], a fault excitation function in which the logical value of the line with open defect depends on the logical values of its adjacent lines is proposed to diagnose the open faults. In the previous method, we defined the fault excitation function as a majority function. However, not all fault excitation functions are a majority function. From observation obtained from TEG (Test Element Group) chips using a 90nm process in [11], we determined that an effect of an adjacent line depends on the length of the adjacent line and the distance to a faulty line. Each adjacent line has each weight for the faulty line. Therefore, we propose a new approach that introduces a fault excitation function as a threshold function of the adjacent lines.

Some open diagnosis methods considering a fault excitation function have been proposed [4-6]. However, these methods do not use the feature that the fault excitation function for open faults is a threshold function.

In this paper, we propose a method for improving the quality of open fault diagnosis. We introduce the fault excitation function as a threshold function of the adjacent lines. On a real defect in a chip, an open defect site exists at somewhere on the faulty line with the narrow defect region. To deduce the open defect site at the faulty line is more useful for the failure analysis. In the method proposed in this paper, 1) candidate faults are deduced by deducing a fault excitation function from test results, and 2) open defect sites at the faulty line is deduced by using relative weights of the adjacent lines for the faulty line, where the weight represents how much the adjacent line affects the faulty line.

Experimental results show that the proposed method can identify an exact faulty line in most cases. The proposed method can also identify the open defect site

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.53

within 25%-length of the faulty line. The computation time for the circuit with about 2 million gates is less than 2 minutes.

This paper is organized as follows. In section 2, we describe the open fault model. In section 3, we explain the outline of the proposed method. In section 4, the effectiveness of the method is demonstrated by computer experiments and section 5 concludes the paper.

2. Open fault model with considering adjacent lines

In this paper, we consider the open fault model described in [2-5]. Let V_f be voltage of a faulty line f with the open defect. Also let a_i (i = 1, 2, ... , n) be adjacent lines of f, and V_i be voltage of a_i. In the fault model, V_f is defined as follows;

$$V_f = \sum W_i \cdot V_i , \qquad (1)$$

where the weight W_i represents how much the adjacent line a_i affects f. Let V_{th} be threshold voltage of the gate with f in its input. The logical value v_f of f is given by the following equation.

$$v_f = \begin{cases} 0 & V_f < V_{th} \\ 1 & V_f \geq V_{th} \end{cases}$$

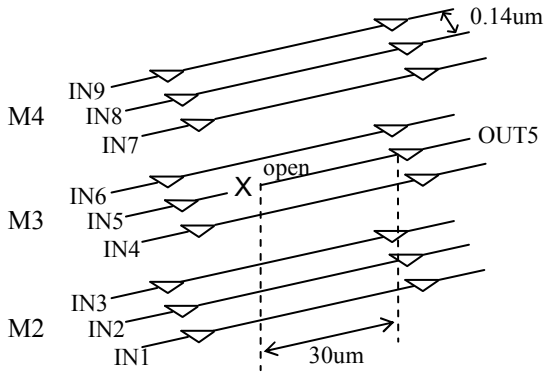


Fig.1 A structure of the faulty line and its adjacent lines

Table 1 The observed values at the faulty line 5

IN1	IN2	IN3	IN4	IN6	IN7	IN8	IN9	OUT5
0	0	0	1	0	0	0	0	0
1	1	1	0	0	1	1	1	0
0	0	0	1	1	0	0	0	1
1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	1	1

This equation means that v_f is determined by the logical values of its adjacent lines.

In order to ascertain the effect from adjacent lines, we have fabricated TEG chips using a 90nm STARC/ASPLA 6 layer-Metal CMOS technology [11]. Figure 1 shows the structure of a test element in the TEG chips. In this figure, the line 5 is the faulty line with open defect, and others are its adjacent lines. The length of each line is 30um. Table 1 shows the observed logical values of the line 5 in the TEG chip. As shown in the fourth row of Table 1, logic 1s on two adjacent lines 4 and 6, which are on the same layer of the faulty line 5, force the line 5 to a logic 1. Even if the line 4 or 6 has a logic 0, 1s on all of the other adjacent lines force the line 5 to a logic 1, as shown in the fifth and sixth rows of Table 1.

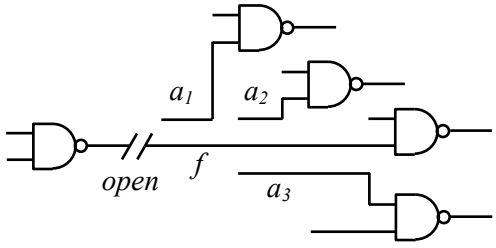
A logical expression which represents conditions of exciting a fault is called a fault excitation function (*FEF*). For the open fault model, since the logical value v_f of the faulty line depends on the logical values v_i of its adjacent lines a_i, the FEF is a function of the adjacent lines.

$$v_f = FEF(v_1, v_2, \ldots , v_n)$$

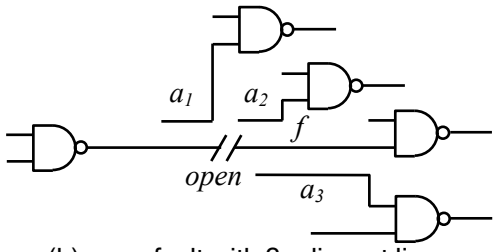
We have proposed a method to diagnose open faults[3]. In this method, we defined a fault excitation function as a majority function.

$$v_f = FMJ(v_1, v_2, \ldots , v_n)$$

However, the weight W_i in the equation (1) depends on the length of the adjacent line a_i and the distance to the faulty line. Therefore, not all the fault excitation functions are equally determined by the majority function.



(a) open fault with 3 adjacent lines



(b) open fault with 2 adjacent lines

Fig.2 An example of open fault

In order to improve the diagnostic quality of open faults, we introduce the fault excitation function as a threshold function (*FTH*) of the adjacent lines. In generally,

$v_f = FTH(v_1, v_2, \ldots, v_n)$.

Since FTH changes according to the open defect site at the faulty line, it is very difficult to predict it before diagnosis process. For example, FTH of Fig.2(a) is

$v_f = FTH\ (v_1, v_2, v_3)$

while that of (b) is

$v_f = FTH\ (v_2, v_3)$.

In the proposed method, FTH is not given before diagnosis. The proposed method deduces the FEF for each line from test results, and deduces candidate faults by checking the FEFs are a FTH or not.

3. Open fault location procedure

The proposed method consists of 4 steps.
(STEP-1) Deduce candidates by using a stuck-at fault simulator.
(STEP-2) Deduce candidates by checking monotonicity of FEFs.
(STEP-3) Deduce candidates by checking complete monotonicity of FEFs.
(STEP-4) Deduce open defect sites.

Definition 1 *A test t_i is called a fail test if errors have been observed at primary outputs in t_i, and otherwise a pass test.*

3.1. Outline of STEP-1

In all of the fail tests, an error generated on the faulty line must be propagated to primary outputs. Therefore, if there is at least one fail test that does not detect stuck-at fault on a line f_c, f_c is decided to be fault-free. Figure 3 shows the procedure of STEP-1.

```
STEP-1()
{
  C = the set of all lines;
  for (every fail test t) {
    stuck_at_fault_simulation(t);
    for (every line fc ∈ C ) {
      if (t does not detect stuck-at fault on fc) {
        delete fc from C;
      }
    }
  }
}
```

Fig.3 Procedure of STEP-1

3.2. Outline of STEP-2

We first explain three premises used in this paper which form the basis of the algorithm in STEP-2.

Premises 1: The value of faulty line f is decided uniquely by the combination of the signal values on its adjacent lines.
Premises 2: An error must occur on f at all fail tests.
Premises 3: If a pass test t detects a stuck-at fault on f, no error occurs on f at t.

Using an example, let us explain the outline of STEP-2. Let v_i be the signal value of an adjacent line a_i, and v_c be the signal value of a candidate fault f_c. Also let G be a deduced FEF for f_c. Table 2 shows fault-free values of f_c and its adjacent lines a_1, a_2 and a_3. The tests t_1 and t_2 are fail tests. The tests t_3 and t_4 are pass tests of which detect the stuck-at fault on f_c. If open fault occurs on f_c, 0-error must be generated on f_c at the fail test t_1. In test t_1, v_1=0, v_2=1 and v_3=1. Since the FEF G must be an FTH and an FTH is a monotonic function, v_1=0 is a sufficient condition for v_c=0.

Although the test t_3 detects stuck-at fault on f_c, t_3 is a pass test. Therefore, v_1=v_2=v_3=0 is a sufficient condition for v_c=0. Therefore,

$$\overline{G} = \overline{v_1} + \overline{v_1}\overline{v_2}\overline{v_3} = \overline{v_1}\ .$$

Similarly, we obtain

$$G = v_1 v_2 + v_3$$

for tests t_2 and t_4. It is obvious that $G \cdot \overline{G}$ must be 0. However,

$$G \cdot \overline{G} = \overline{v_1} v_3$$

in this example. This means that $G(0, x, 1)$=0 at the fail test t_1, while $G(0, x, 1)$=1 at the pass test t_4. From this inconsistency, f_c is decided to be fault-free.

Figure 4 shows the procedure of STEP-2.

Table 2 An example of STEP-2

		v_c	v_1	v_2	v_3
fail tests	t_1	1	0	1	1
	t_2	0	1	1	0
pass tests	t_3	0	0	0	0
	t_4	1	0	0	1

```
STEP-2()
{
  // C : the set of candidate faults
  // a_i : adjacent lines
  // v_i : variables for a_i
  // n : the number of adjacent lines
  // v(x, t) : fault-free value of line x in test t
  G = Ḡ = 0;
  for (every candidate fault fc ∈ C ) {
    for (every fail test t) {
      if (v(fc, t) ==1) {
        Ḡ = Ḡ ∨ (⋀_{i=1}^{n} (v(a_i, t) ∨ v̄_i));
      } else {
        G = G ∨ (⋀_{i=1}^{n} (v̄(a_i, t) ∨ v_i));
      }
    }
    for (every pass test t) {
      if (t detects a stuck-at fault on fc) {
        if (v(fc, t) == 0) {
          Ḡ = Ḡ ∨ (⋀_{i=1}^{n} (v(a_i, t) ∨ v̄_i));
        } else {
          G = G ∨ (⋀_{i=1}^{n} (v̄(a_i, t) ∨ v_i));
        }
      }
    }
    if ( G · Ḡ ≠ 0 ) {
      delete fc from C;
    }
  }
}
```

Fig.4 Procedure of STEP-2

3.3. Outline of STEP-3

In STEP-2, we check the monotonicity of FEF to deduce the candidates. In order to reduce the number of candidates, in STEP-3, we add to check whether FEF is a threshold function or not. It is known that an n-variable ($n \le 8$) complete monotonic function is a threshold function. We, therefore, check the complete monotonicity of FEF ($n \le 8$) as follows [12].

Let G be a *FEF*, and X_i be a input vector.

Definition 2 *If a pair of input vectors (X_1, X_2), (X_3, X_4) satisfies $X_1+X_2=X_3+X_4$, the pair is called a diagonal vector pair.*

Definition 3 *If a pair of input vector (X_1, X_2), (X_3, X_4) satisfies*

$$X_1 - X_3 = X_4 - X_2 = (v_1,...,v_n)$$
$$k = \sum |v_i|,$$

the pair is called a distance-k diagonal vector pair.

Definition 4 *If no diagonal vector pair whose distance is lower than k satisfies*

$$\{G(X_1) - G(X_3)\} \cdot \{G(X_4) - G(X_2)\} < 0,$$

G is called a k-monotonic function.

If an n-variable function G is a $\lfloor n/2 \rfloor$-monotonic function, G is a complete monotonic function.

3.4. Outline of STEP-4

It is difficult to decide the correct value of weight W_i in the equation (1), because W_i depends on the threshold voltage of the gate with the faulty line in its input. However the relative weights of adjacent lines are decided easily from the length of adjacent lines and the distance to the faulty line [5]. We define the weight W_i as follows;

$$W_i = C \cdot w_i$$

while C is a constant, and w_i is the relative weight. In STEP-4, the open defect site at the faulty line is deduced by using this relative weight w_i.

Let w_i be the relative weight of the adjacent line a_i in the fault-free circuit, and w_i' be that in the faulty circuit. Assume that a candidate fault has adjacent lines a_j, a_k, a_m and a_n. The relationship of relative weights in the fault-free circuit is given as follows.

$$w_j + w_k \ge w_m + w_n \qquad (2)$$

Let G=(a_j, a_k, a_m, a_n) be the FEF deduced in STEP-3. If

$G(1, 1, 0, 0) = 0$
$G(0, 0, 1, 1) = 1$,

then we deduce that the relationship of relative weights in the faulty circuit as follows.

$$w_j' + w_k' < w_m' + w_n' \qquad (3)$$

The expression (3) is different from (2). Therefore, the open defect must occur in the range that the expression (3) is satisfied.

Using an example, let us explain the outline of STEP-4. Figure 5 shows a geometrical pattern of a faulty line and its adjacent lines, and relative weights of adjacent lines. Let $G(v_1, v_2, v_3, v_4)$ be the FEF deduced at STEP-3. If

$G(1, 1, 0, 0) = 0$
$G(0, 0, 1, 1) = 1$,

then we deduce that the relationship of relative weights in the faulty circuit as follows.

$w_1'+w_2' < w_3'+w_4'$

In this example, the relationship of relative weights in the fault-free circuit is as follows.

$w_1 + w_2 > w_3 + w_4$

In order to decide the open defect site easily, we consider only the discrete distance on the candidate faulty line from its input gate. In Fig.5, the length of the candidate faulty line is 8. Since the weight of an adjacent line is proportional to its length, the weight at each point is decided as shown in Table 3. From Table 3, $w_1'+w_2'$ is smaller than $w_3'+w_4'$ from point 2 to 4. Therefore, the open defect site must be within this range. If no point satisfies the inequality, f_c is decided to be fault-free.

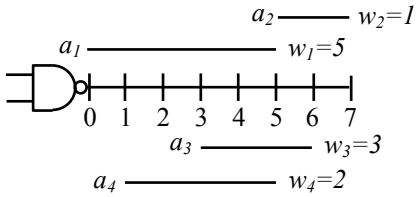


Fig.5 Faulty line and its adjacent lines

Table 3 Weights in the faulty circuit

	0	1	2	3	4	5	6	7
w_1'	5	4	3	2	1	0	0	0
w_2'	1	1	1	1	1	1	0.5	0
w_3'	3	3	3	3	2	1	0	0
w_4'	2	2	1.5	1	0.5	0	0	0
$w_1'+ w_2'$	6	5	4	3	2	1	0.5	0
$w_3'+ w_4'$	5	5	**4.5**	**4**	**2.5**	1	0	0

4. Experimental results

To evaluate the performance of the diagnosis method presented in Sec.3, we used ISCAS'89 benchmark circuits, ITC'99 benchmark circuits and subcircuits of STARC'03 benchmark circuit [13]. The profiles of STARC03 benchmark circuits are shown in Table 4, where the number of primary inputs, the number of primary outputs and the number of gates are shown. In the experiment, we used 3000 random tests. Adjacent lines and fault excitation functions are generated randomly. The number of adjacent lines is 8 for all lines. In STEP 4, the line length is 20 for all lines. The diagnosis program was run on a computer with Intel Core2 (2.4GHz) and 2GB memory. In this experiment, we use only pass/fail information. Table 5 shows the mean values for randomly sampled 100 open faults. In Table 5, "# of DSs" shows the number of candidate open defect sites deduced in STEP-4. In order to reduce the CPU time, stuck-at fault simulation in STEP-1 was applied to first 32 fail tests.

Simulation results show that the faulty line can be exactly identified in most cases. CPU time for s3-3 (2.3M gates) is less than 2 minutes. The candidate open defect site is about 25%-length of the faulty line.

5. Conclusion

We have proposed a method for improving the quality of open fault diagnosis by introducing the fault excitation function as a threshold function of the adjacent lines. Experimental results show that the proposed method can identify an exactly faulty line in most cases. CPU time for 2.3M gates circuit is less than 2 minutes. Therefore the proposed diagnosis method is applicable to large circuits.

In future, we have a plan to submit a paper for modeling dynamic open faults. We also have a plan to propose a method for generating tests for dynamic open faults with considering adjacent lines.

Acknowledgment

This work was supported in part by Semiconductor Technology Academic Research Center (STARC) under the Research Project.

The VLSI chip in this study has been fabricated through the chip fabrication program of VLSI Design and Education Center (VDEC), the University of Tokyo, with the collaboration by STARC, Fujitsu Limited, Matsushita Electric Industrial Company Limited., NEC Electronics Corporation, Renesas Technology Corporation, and Toshiba Corporation.

References

[1] S. M. Reddy, I. Pomeranz, H. Tang, S. Kajihara and K. Kinoshita, "On Testing Interconnect Open Defects in Combinational Logic Circuits with Stems of Large Fanout," Proc. ITC, pp.83-87, 2002.

[2] Y. Sato, I. Yamazaki, H. Yamanaka, T. Ikeda and M. Takakura, "A Persistent Diagnostic Technique of Unstable Defects," Proc. ITC, pp.241-249, 2002.

[3] H. Takahashi, Y.Higami, S. Kadoyama, T. Aikyo, Y., Takamatsu, K. Yamazaki, T. Tsutsumi, H. Yotsuyanagi and M. Hashizume, "Clues for Modeling and Diagnosing Open Faults with Considering Adjacent Lines, " Proc. ATS, pp.39-44, 2007.

[4] C. Liu, W. Zou, S. M. Reddy, W-T. Cheng, M. Sharma and H. Tang, "Interconnect Open Defect Diagnosis with Minimal Physical Information," Proc. ITC, 2007.

[5] R. R-Montanes, D. Arumi, J. Figueras, S. Einchenberger, C. Hora, B. Kruseman, M. Lousberg and A. K. Majhi, "Diagnosis of Full Open Defects in Interconnecting Lines," Proc. VTS, pp.158-166, 2007.

[6] R. Desineni, O. Poku and R. D. Blanton, "A Logic Diagnosis Methodology for Improved Localization and Extraction of Accurate Defect Behavior," Proc. ITC, 2006.

[7] S. Venkataraman and S. B. Drummonds, "A Technique for Logic Fault Diagnosis of Interconnect Open Defects," Proc. VTS, pp.313-318, 2000.

[8] S. Huang, "Diagnosis of Byzantine Open-Segment Faults," Proc. ATS, pp.248-253, 2002.

[9] W. Zou, W. Cheng S. M. Reddy, "Interconnect Open Defect Diagnosis with Physical Information," Proc. ATS, pp.203-209, 2006.

[10] Y. Sato, H. Takahashi, Y. Higami and Y. Takamatsu, "Faillure Analysis of Open Faults by Using Detecting/Un-detecting Information on Tests," Proc. ATS, pp.222-227, 2004.

[11] H. Yotsuyanagi, M. Hashizume, H. Takahashi, T. Tsutsmi, K. Yamazaki, T. Aikyo, Y. Higami and Y. Takamatsu, "Fault Effect of Open Faults Caused by Adjacent Signal Lines in a 90nm IC," Proc. VLSI Design, 2009.

[12] Y. Takamatsu, K. Murakami and T. Aibara, "Isobaricity for a Set of Multivalued Logic Functions and Its Application to the Synthesis of Multithreshold Threshold Elements," IECE Systems. Computers. Controls., vol.5, No.5, pp.28-35, 1974.

[13] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama, and S.Kajihara, "Invisible Delay Quality-SDQM Model Lights Up What Could Not Be Seen," Proc. ITC, pp.1202-1210, 2005.

Table 4 Specifications of STARC'03 benchmark circuits

circuits	# of PIs	# of POs	# of gates
s3-1	1,486	1,307	44,516
s3-2	18,373	17,679	107,695
s3-3	52,111	45,641	2,272,320

Table 5 Experimental results

Circuit	# of gates	# of candidate faults			CPU time (s)	# of DSs
		min	max	avg		
cs38417	22k	1	1	1.00	0.2	5.3
cs38584	19k	1	2	1.02	0.1	5.9
cb17	31k	1	4	1.04	0.5	7.3
cb18	111k	1	3	1.02	2.8	6.9
cb19	225k	1	1	1.00	3.6	6.9
s3-1	45k	1	1	1.00	1.0	4.5
s3-2	108k	1	1	1.00	1.6	5.9
s3-3	2.3M	1	1	1.00	108.7	5.2

Fault Effect of Open Faults Considering Adjacent Signal Lines in a 90 nm IC

Hiroyuki Yotsuyanagi[1], Masaki Hashizume[1],
Toshiyuki Tsutsumi[2], Koji Yamazaki[3], Takashi Aikyo[4],
Yoshinobu Higami[4], Hiroshi Takahashi[4], Yuzo Takamatsu[4]

[1]Institute of Technology and Science, The Univ. of Tokushima
[2]School of Science and Technology, Meiji University
[3]School of Information and Communication, Meiji University
[4]Graduate School of Science and Engineering, Ehime University

Abstract

Open faults are difficult to test since the voltage at the floating line is unpredicted and depends on the voltage at the adjacent lines. The modeling for open faults with considering adjacent lines has been proposed in [10]. In this work, the 90 nm IC is designed and fabricated to evaluate how the voltage at adjacent lines affect the defective line. The open fault macros with a transmission gate and with an intentional break are included in the IC. The nine lines are placed in parallel in three layers to observe the effect of the coupling capacitance when an open occurs. The benchmark circuits with the open fault macro are also included in the IC. The simulation and experimental results show that the relationship between the floating line and the adjacent lines. The experimental results are also compared with the open fault model that calculate the weighted sum of voltages at the adjacent lines.

1. Introduction

Open faults and bridging faults are the faults most likely occur in deep sub-micron(DSM) ICs. Some of the faults can not behave like conventional stuck-at fault model and are hard to be detected [1,5–8]. Especially open faults cause unstable voltage at the faulty wires. It has been pointed out that the voltage at the floating wires occurred by an open fault is influenced by its adjacent lines [8, 13].

In [2, 7, 9, 11], test generation methods for open faults have been proposed. Another test approach in [3] detects open defects by applying time-varying electric field from outside of ICs to swing the voltage of floating wires that causes the logic error at outputs.

In [4, 10–12], fault model of open faults which takes the effects of the adjacent lines into account has been proposed. In the model, it is required to estimate how the voltage at adjacent lines affects the defective line.

The fault effects of open defects in real chips have been reported in some paper [1, 5, 6, 13]. These results are obtained for 1.5 μm to 0.35 μm technology. In this paper, the 90 nm IC is designed and fabricated which includes open faults. The open faults are intentionally inserted into the IC to observe the effect of adjacent lines around a faulty line. The adjacent lines are placed not only in the same layer but also in the different layers in our IC to evaluate the effects between the layers.

This paper is organized as follows. In Section 2, the open fault model assumed in this work is shown. To estimate the effect of the adjacent lines, the open fault macros with a transmission gate and an intentionally break are designed. Its design and simulation results are also shown in Section 2. In Section 3, the description about the test chip fabricated in 90 nm technology is given. Experimental results are shown in Section 4 and section 5 concludes the paper.

2. Open fault macros

2.1. Open fault model with adjacent lines

In [10, 11], the voltage at a floating line is modeled as the weighted sum of the voltages at the adjacent lines. The faulty voltage on the line v is defined as the following equation.

$$V_{v'} = \Sigma(\alpha_i \times V_{ai}) + \beta \times V_v \quad (1)$$

where $V_{v'}$ is the voltage at the faulty line, V_{ai} is the voltage at adjacent line ai, and α_i and β are the coefficients. For a complete disconnected open fault, these coefficients are mainly related to the coupling capacitance and the gate capacitance connected to the faulty line. The purpose of our experimental IC is to estimate the relationship between a faulty line and its adjacent lines.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.60

2.2. Design of open fault macros

In this work, two kinds of open faults are intentionally inserted into the design. One is an open fault realized by a transmission gate as shown in Fig. 1. Input TG is utilized to control the transmission gate to emulate an open fault. The other is a complete open fault realized by a break in a signal line as shown in Fig. 2. The break is designed by inserting a minimum space (0.14 μm) that is allowed by the design rule. To estimate the effect of the coupling capacitances between the floating line and the adjacent line, an nMOSFET is placed between the floating line and the ground in order to remove the charge unlike the test chip in [1]. It also prevents the excessive current from flowing in open fault macros which are not selected to be observed. In the normal mode, output $OUT4$ of the open fault macro is set to H level when $RESIN$ = L level. The expected output including this macro is the same as the circuit with a stuck-at 0 fault at the faulty line.

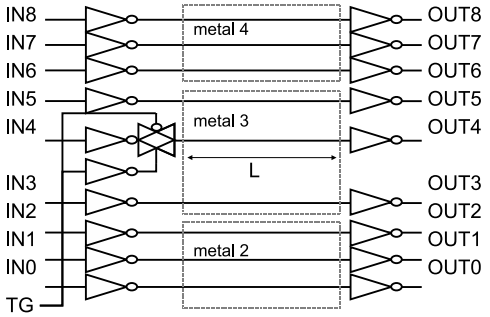


Figure 1. Open fault macro with a transmission gate

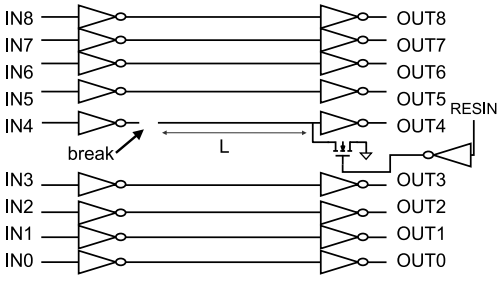
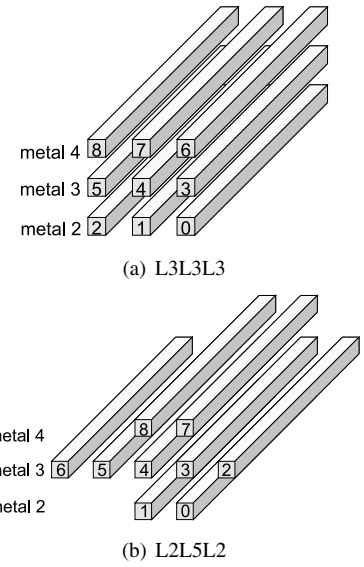


Figure 2. Open fault macro with an intentional break

The line with an open fault is designed with eight adjacent lines in three layers , Metal 2, Metal 3 and Metal 4. Two structures of lines called L3L3L3 structure and L2L5L2 structure are constructed as shown in Fig. 3(a) and Fig. 3(b), respectively. An open fault is placed at line 4 for each open fault macro. In L3L3L3 structure, three lines are placed in metal 2, metal 3, and metal 4. This structure is utilized to observe the difference of the coupling effects among the layers. In L2L5L2 structure, two lines are placed in metal 2 and metal 4, and five lines are placed in metal 3. This structure is utilized to observe the effect of multiple adjacent lines in the same layer. The lines placed in parallel with minimum distance (0.14 μm) described in the design rule.



(a) L3L3L3

(b) L2L5L2

Figure 3. Placement of adjacent lines

2.3. Simulation results

The layout of the open fault macros described above is designed in 90 nm CMOS technology. Some layouts are designed with different length L of the faulty line. The HSPICE simulation is applied for the netlists extracted from the layouts. The maximum voltage at the floating line, V_{open}, is calculated by applying the input vector which bring the signal at all eight adjacent lines from L to H. Fig. 4 shows the relationship between the maximum voltage at the faulty line V_{open} versus the length of the faulty line L obtained for the open fault macros. The results obtained for the open fault macro with a transmission gate are labeled as TG. The results obtained for the open fault macro with a break are labeled as BR. Since the supply voltage is 1.0[V] and the threshold voltage for logic H is about 550[mV] obtained from an inverter gate, logical error will be occurred for the faulty line with length more than 30 μm.

Fig. 5 and Fig. 6 show the HSPICE simulation results

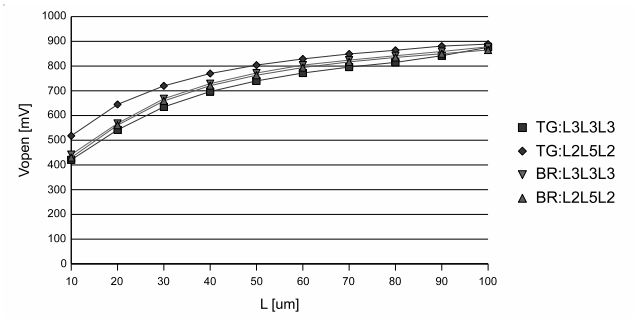


Figure 4. Simulation results of the length of a faulty line L vs V_{open} characterization

how much the adjacent lines affect the voltage of the faulty line. The simulation is applied for the open fault macro with a transmission gate with L3L3L3 structure of adjacent lines. The length L of the faulty line is set to 100 μm.

The effect of each adjacent line is shown in Fig. 5. In this simulation, the input pattern is provided such that the signal transition from L to H occurs only one adjacent line at a time. Note that an inverter gate is inserted between the input and the parallel lines. An open fault is inserted in line 4 in Fig. 3(a). The signal transition occurs at the adjacent lines (8, 0, 6, 2, 7, 1, 5, 3) in this order. The waveforms shown below in the figure are the voltage waveforms of the output $OUT4$ and the faulty line (dashed line). It can be seen that the adjacent lines at the same layer can bring the largest voltage change at the faulty line among the eight adjacent lines. Since the single transition from L to H can not bring the voltage at the faulty line over the threshold voltage of a logic gate, the output voltage was not affected by the fault.

The simulation result to observe the effect of combination of the adjacent lines is shown in Fig. 6. In this simulation, the signal transition from L to H consecutively occurs at the adjacent line (8, 0, 6, 2, 7, 1, 5, 3) in this order. The voltage at the faulty line rises as the adjacent lines at which the voltage changed from L to H increase as shown in the waveform at the faulty line shown in the dashed line. The output voltage shows an error when all lines at the different layer (lines 8,0,6,2,7,1) and one line at the same layer (line 5) becomes H level.

3. An experimental test chip with open faults

In this work, an IC including the circuits with open faults is designed and fabricated in 90 nm STARC/ASPLA 6 layer-Metal CMOS technology. Our experimental test chip includes the open fault macros described in Section 2 and the macros of ISCAS85 benchmark circuits c17, c880, and

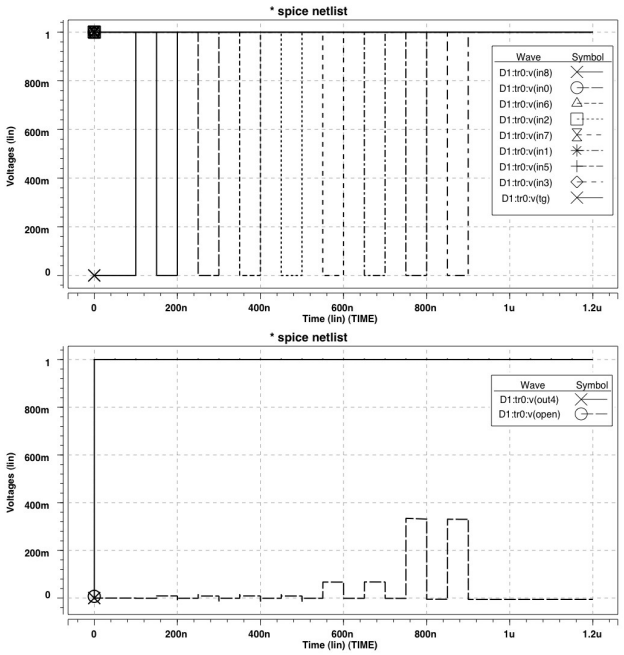


Figure 5. Fault effects due to a single adjacent line

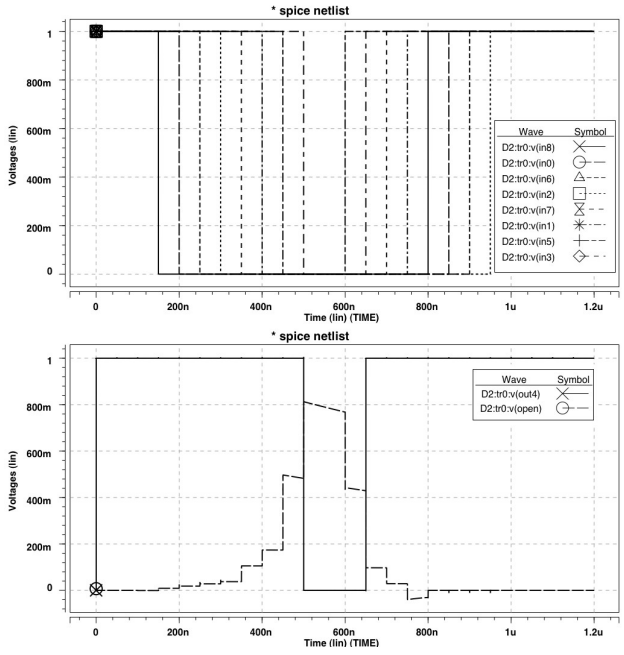


Figure 6. Fault effects due to multiple adjacent lines

c3540, with the open fault macro. The length of the floating line L in the open fault macros is set to 100 μm in the IC.

Each macro has nine inputs and nine outputs and a control input TG or $RESIN$ that selects the faulty behavior of a macro. For benchmark c17, some extra inputs and outputs and inverter chains are added. Five lines in the benchmark circuit and four lines in the inverter chains are selected and replaced with the open fault macro. For benchmark c880 and c3540 that have primary inputs/outputs more than nine, the two input patterns can be provided through scan chains. Fig. 7 shows the circuit diagram of the macros with benchmark circuits c880 and c3540. Nine lines in the benchmark circuits are selected and replaced with the open fault macro. These lines are selected such that many combination of signal values are obtained by 1000 random input patterns. To observe effects of an open fault, the lines are selected such that they are not on the output paths from an open fault. The selected lines are replaced with an open fault macro in verilog netlist. The circuit layout is then obtained by a place and route tool. Fig. 8 shows the layout of the c3540 with an open fault macro. The IC includes twenty-four macros comprised of the open fault macros and the benchmark circuits with the open fault macro.

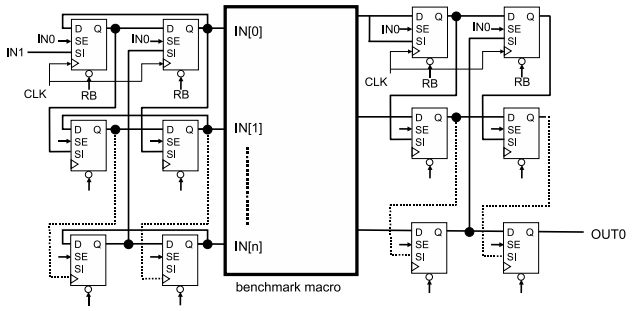


Figure 7. The input/output circuits for benchmark macros

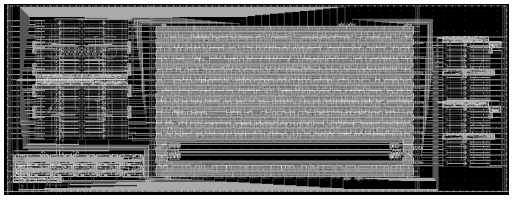


Figure 8. The layout of the benchmark circuit with an open fault macro

4. Experimental results

To observe the influence of the adjacent lines on the signal line with an open fault, several input patterns are applied to the fabricated ICs. Some of the patterns and the results obtained for the IC are shown in Table 1. The pattern p1 to p8 are applied that provide the signal values at the parallel lines as in Column ($a_8a_7a_6\ a_5va_3\ a_2a_1a_0$). v denotes the input of the faulty line and a_j denotes the adjacent lines as in Fig. 3. For the macros including benchmark circuits c880 and c3540, the two patterns that provide the assignments in the column are selected and applied through the scan chain shown in Fig. 7. Column $OUT4$ shows whether an error is detected at the output of the fault line or not. The voltage supplied to the faulty line is set to L, therefore the fault-free output value at $OUT4$ is H. This is the case that V_v is set to L in the open fault model in Eq. 1. In this experiments, first the control input TG or $RESIN$ is set to L to reduce the trapped charge for 2 clock periods (0.4 μ sec). and then these inputs are set to H to make the open fault macro behave like complete open. After that, input patterns are applied.

Pattern p1 is the basic pattern that set all adjacent lines to L level. The pattern is considered not to activate a fault effect. In pattern p2, when all of eight adjacent lines changed its signal from L to H, the output changed to L level and the open fault can be detected. In pattern p3, the output value was changed when two adjacent lines at the same layer (metal 3) changed its signal value. Only two nearest adjacent lines a_5 and a_3 can bring the voltage at floating line up to the threshold voltage of the inverter. This is also deduced from the simulation result in Fig. 5. Pattern p4 changes the adjacent lines except the nearest lines a_5, a_3. In this case, the fault effect at the output signal did not be observed. Pattern p5 changes only one nearest adjacent line a_5. The faulty output values were not observed for the pattern. The same output patterns were also be observed for the other adjacent lines. Any one of the adjacent lines can not bring the faulty line from L to H in the experiments. Pattern p6 changes the adjacent lines except one of the nearest line a_3. In this case, the error was found at the output. This is also deduced from Fig. 6. It is confirmed by the results obtained for pattern p5 and p6 that the voltage at the floating line is affected from the other layers. The results shown in Table 1 were obtained for both the open fault macros and the benchmark circuit with the open macros. Pattern p7 and p8 shows the case that the adjacent line a_0 affected whether the faulty output was observed or not. The line a_0 is placed in the lower right side from the floating line. Although the effect from a_0 is considered to be very small, the logic value at the line affects the faulty behavior as shown in the case for pattern p7 and p8.

In our experiments, the clock is provided at a speed of 5

MHz. Pattern p1 to p8 are repeatedly applied with some different period for the open fault macro and the fault macro of benchmark circuit c17. Fig. 9 is the output signal obtained for pattern p1. In this experiments the fault effect did not remain for 1 μsec. The voltage at output $OUT4$ becomes H before the adjacent lines a_j changed from H to L. It seems that the trapped charge was reduced with time by leakage. The similar output pattern was also observed for the other patterns and for the other open fault macros.

Table 1. The examples of the input patterns and the results of fault detection

pattern	adjacent lines ($a_8a_7a_6\ a_5va_3\ a_2a_1a_0$)	OUT4 (v')
p1	(LLL LLL LLL)	H
p2	(HHH HLH HHH)	L
p3	(LLL HLH LLL)	L
p4	(HHH LLL HHH)	H
p5	(LLL HLL LLL)	H
p6	(HHH HLL HHH)	L
p7	(HHL LLH HHH)	L
p8	(HHL LLH HHL)	H

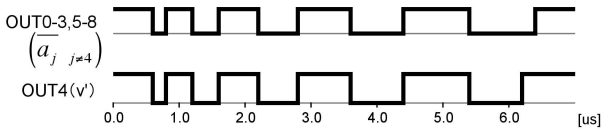


Figure 9. Output signal obtained for pattern p1 and p2

The input pattern used in the simulation of Fig. 6 was applied to the IC. The pattern changes the voltage at the adjacent lines in descending order of the distance from the faulty line. Fig. 10 shows the output pattern obtained for the same input pattern as in the simulation of Fig. 6. The output $OUT4$ changes when the adjacent lines except the nearest lines a_5, a_3 and one of the nearest adjacent line are H level.

The coupling capacitance between the lines in the same layer is about five to ten times greater than the capacitance between the lines in the different layer according to the extracted value from the layout. Table 2 shows the coupling capacitance extracted from the open fault macro with $L = 100\mu m$. The simulation results also shows the difference among adjacent lines as shown in Fig. 4,5 and 6. The experimental IC also shows there exists the difference between the nearest adjacent lines and the other adjacent lines.

Table 2. The coupling capacitance extracted from the layout of Fig. 2 (L=100μm)

adjacent lines	coupling capacitance [fF]
$a_5 - v'$	8.82
$a_3 - v'$	8.81
$a_7 - v'$	1.80
$a_1 - v'$	1.71
$a_2 - v'$	0.25
$a_8 - v'$	0.25
$a_6 - v'$	0.25
$a_0 - v'$	0.22
$v - v'$	0.01

To estimate the relation between the model in [10, 11], we calculate $V_{v'}$ in Eq.(1) with $\alpha_i = C_{v'-a_i}/C_{total}$. C_{total} is the total amount of the capacitances shown in Table. 2 (=22.12). All combination of nine inputs were applied to the fabricated chips. All pattern are applied three times for five chips. The relation between $V_{v'}$ calculated for the inputs and the experiments are follows:

- For $V_{v'} < 0.497$ (235 vectors), the floating lines behave as L level for all input patterns and for all chips.
- For $0.497 < V_{v'} < 0.505$ (45 vectors), the floating lines behave as L level for the inputs that provide $v = L$. However, the logic value is different among the chips for the inputs that provide $v = H$.
- For $0.505 < V_{v'} < 0.570$ (60 vectors), the observed logic values differed from the chips.
- For $0.570 < V_{v'}$ (172 vectors), the floating lines behave as H level for for all input patterns and for all chips.

For the intermediate value of $V_{v'}$, the logic value is not determined by the input vector. The process variation affects the results. Therefore, some modification of α may differentiate the logic value of the floating line, there may remain the case that the logic value at the floating line is not predicted. In this example, for 407 vectors out of 512 vectors the observed logic values are matched with the prediction given by the calculation.

The IC also includes test elements of open fault macros with various parameters such that the length of the floating lines and the distance between the floating lines. To investigate more detailed condition for the faulty output due to the coupling effects, another 90 nm chip is currently fabricated and the results obtained for both chip will be reported in the future work.

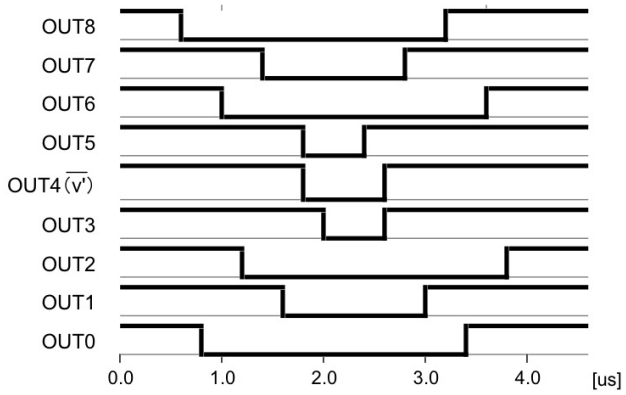


Figure 10. Output signal obtained for pattern Fig. 6

5. Conclusion

The IC with test elements for investigating the fault effect of open faults and its adjacent lines are designed and fabricated in 90 nm CMOS technology. It can be found that two or more adjacent lines will affect the output logic. The effect of adjacent lines depends on the length and the distance between the faulty line and the adjacent lines. There exists the case that the fault effect depends on the adjacent lines at different layers from the faulty line. The comparison between the weighted sum of voltages at the adjacent lines and the observed logic values shows the logic value at the floating lines can be predicted for almost 80% of input vectors. The more detailed relationship between the voltage at the faulty line and the parameters of adjacent lines such as the length and the distance will be analyzed in the future work.

Acknowledgement

This work was supported in part by Semiconductor Technology Academic Research Center (STARC) under the Research Project. The VLSI chip in this study has been fabricated through the chip fabrication program of VLSI Design and Education Center(VDEC), the University of Tokyo, with the collaboration by STARC, Fujitsu Limited, Matsushita Electric Industrial Company Limited., NEC Electronics Corporation, Renesas Technology Corporation, and Toshiba Corporation. The authors would also like to thank the reviewers of this paper for their valuable comments.

References

[1] D. Arumi, R. Rodriguez-Montanes, and J. Figueras. Experimental characterization of CMOS interconnect open defects. *IEEE Trans. on Computer-Aided Design*, 27(1):123–136, Jan 2008.

[2] R. Gomez, A. Giron, and V. Champac. Test of interconnection opens considering coupling signals. *Defect and Fault-Tolerance in VLSI Systems, IEEE International Symposium on*, pages 247–258, 2005.

[3] M. Hashizume, M. Ichimiya, H. Yotsuyanagi, and T. Tamesada. CMOS open defect detection by supply current measurement under time-variable electric field supply. *IEICE transactions on information and systems*, E85-D(10):1542–1550, Oct. 2002.

[4] M. Hashizume, Y. Yamada, H. Yotsuyanagi, T. Tsutsumi, K. Yamazaki, Y. Higami, H. Takahashi, and Y. Takamatsu. Fault analysis of interconnect opens in 90nm CMOS ICs with device simulator. *Proc. International Technical Conference on Circuits/Systems, Computers and Communications*, pages 249–252, 2008.

[5] J. C.-M. Li, C.-W. Tseng, and E. McCluskey. Testing for resistive opens and stuck opens. *Proc. International Test Conference*, pages 1049–1058, 2001.

[6] E. J. McCluskey and C.-W. Tseng. Stuck-fault tests vs. actual defects. *Proc. International Test Conference*, pages 336–342, 2000.

[7] S. M. Reddy, I. Pomeranz, H. Tang, S. Kajihara, and K. Kinoshita. On testing of interconnect open defects in combinational logic circuits with stems of large fanout. *Proc. International Test Conference*, page 83, 2002.

[8] Y. Sato, I. Yamazaki, H. Yamanaka, T. Ikeda, and M. Takakura. A persistent diagnostic technique for unstable defects. *Proc. International Test Conference*, pages 242–249, 2002.

[9] S. Spinner, I. Polian, P. Engelke, B. Becker, M. Keim, and W.-T. Cheng. Automatic test pattern generation for interconnect open defects. *Proc. VLSI Test Symp.*, pages 181–186, 2008.

[10] H. Takahashi, Y. Higami, S. Kadoyama, T. Aikyo, Y. Takamatsu, K. Yamazaki, T. Tsutsumi, H. Yotsuyanagi, and M. Hashizume. Clues for modeling and diagnosing open faults with considering adjacent lines. *Proc. Asian Test Symp.*, pages 39–44, 2007.

[11] H. Takahashi, Y. Higami, T. Kikkawa, T. Aikyo, Y. Takamatsu, K. Yamazaki, T. Tsutsumi, H. Yotsuyanagi, and M. Hashizume. Test generation and diagnostic test generation for open faults with considering adjacent lines. *Proc. International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pages 243–251, Sept. 2007.

[12] K. Yamazaki, T. Tsutsumi, H. Takahashi, Y. Higami, T. Aikyo, H. Yotsuyanagi, M. Hashizume, and Y. Takamatsu. A novel approach for improving the quality of open fault diagnosis. *VLSI Design 2009*, 2009. (to appear).

[13] A. Zenteno, V. H. Champac, and J. Figueras. Detectability conditions of full opens in the interconnections. *Journal of Electronic Testing: Theory and Applications*, 17:85–95, 2001.

2009 22nd International Conference on VLSI Design

Efficient Grouping of Fail Chips for Volume Yield Diagnostics

Lavanya Jagan, Ratan Deep Singh, V. Kamakoti
Reconfigurable & Intelligent Systems Engineering Group,
Department of Computer Science and Engineering,
Indian Institute of Technology-Madras, India.
lavanya@cse.iitm.ernet.in, rataniitm@gmail.com,
kama@cs.iitm.ernet.in

Ananta K. Majhi
Corporate Innovation & Technology,
NXP Semiconductors, Eindhoven
The Netherlands.
ananta.majhi@nxp.com

***Abstract*— Volume Yield Diagnostics (VYD) is crucial to diagnose critical systematic yield issues from the reports obtained by testing thousands of chips. This paper presents an efficient clustering technique for VYD that has been shown to work successfully both in the simulation environment as well as on real industrial failure data.**

Keywords: *Volume Yield Diagnostics, Clustering, Fail Signatures, Systematic Defects, Yield*

I. INTRODUCTION

The success of a semiconductor industry crucially depends upon the number of correct chips it produces (yield) within a given time window. Yield analysis has thus become a major issue for the progress of any electronics industry [1], [2]. For the current nanometer technologies, yield remains very low during the initial phases of manufacturing. This is mainly because the manufacturing and lithographic techniques still remain at 193nm whereas the technology nodes have been driven below 60nm [3]. These process-design interactions lead to design rule violations that further results into a class of faults that are termed in the literature as *Systematic defects* [4], [5], [6], [7]. These defects appear over many failing dies of the same device leading to a drastic decrease in yield. Such defects are dangerous as they target very strongly on system's topology and possess very specific defect characteristics. Traditional techniques for yield analysis presented in [8], [9], [10], use memory bit mapping, inline inspection etc. and does target common systematic defects in current nanometer technologies that includes source drain defects, isolation defects, and gate stack defects. The main drawbacks of these approaches are not only that they need highly sensitive advanced inspection tools but also take prohibitively large time for diagnosis making them non-scalable with increasing volumes of failing chips.

Volume Yield Diagnostics (VYD) presented in [7], [11], [12] specifically addresses the problem of scalability. The objective of VYD is to conduct logic diagnostics over a very large set (thousands) of statistically significant failed chips of the same device to identify most critical yield issues. The critical step in VYD is to select the most statistically significant set of failed dies followed by a trends analysis on the diagnostic results of them to precisely identify the cause of failure. If the cause is still unidentified after this stage, a Precision Yield Diagnostics (PYD) is done for a few statistically significant chips to find out the root cause of the failures. Performing VYD as fast as possible without losing much diagnostic accuracy is the challenge faced by the EDA industries today. Methods to reduce the runtime of VYD have been proposed in [13], [14]. The method presented in [13] uses only the first or first few failing test patterns for each failing device for diagnostic simulation. A considerable amount of runtime is saved by following the proposed approach but at the cost of accuracy. An extensive precision diagnostics would be needed in such a case to get reasonably good results even after the trend analysis. The technique proposed in [14] overcomes this by clustering similarly failing chips on the basis of their fail signatures [15]. Fail Signatures of each failed die is essentially a compact representation of the raw failure data of a chip obtained from an Automatic Test Equipment (ATE). The failure data can be further supported with various other types of information like design data, layout data, die coordinates (in case of spatial wafer signatures) primarily to speedup the diagnosis step [6].

The paper proposes an effective way of describing the fail information of the chips (fail signatures) by giving more importance to the data which are most likely to contribute in detecting systematic defects. It also presents an algorithm that uses the proposed fail signatures and has been found to greatly reduce the run-time of clustering without compromising the performance. Comparison of the proposed scheme with the recent one [14] reported in the literature shows the efficiency of the proposed approach.

The rest of this paper is organized as follows. In section II, we discuss the motivation for our work. In section III, we describe our proposed technique. We explain the *weight-based grouping* algorithm and the simulation framework in section IV and V respectively. In section VI, we show experimental results, and finally in section VII, we conclude the paper.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.59

II. MOTIVATION

The main motivation behind the proposed work is the following drawbacks that were observed in the fail signature scheme proposed in the most recent VYD [14] reported in the literature. The test report for a chip, as reported in [14], comprises of K scan flip flops and N test patterns. It is represented by a $N \times K$ matrix *TEST* , such that *TEST* $[i, j]$ is a bit which is set to 1 if the i^{th} pattern failed on the j^{th} scan-flipflop, else it is 0. The method proposed in [14] derives two types of fault signatures from the TEST matrix, namely, scan-flop based and pattern based ones. The scan-flop based signature is a K-bit vector, SCAN, such that *SCAN*$[i]$ is 1, if at least one of the N patterns failed on the i^{th} scan flipflop, else it is 0. Similarly, the pattern-based signature is a N-bit vector, *PAT*, such that *PAT*$[i]$ is 1, if at least one of the K scan flip flops failed on the i^{th} pattern, else it is 0. The technique proposed in [14] groups the chips based on how close are the fail signatures. The *commonality measure* between two k-bit fail signatures as defined in [14] is the ratio of the number of bit positions that have a 1 in both the signatures to the number of bit positions in which at least one of the signatures has a 1. In other words, it is the ratio of the common failures to the total number of failures. The major drawbacks of the grouping technique proposed in [14] are as follows:

Ineffective Fail Signatures: The fail signatures do not attach any additional importance to probable systematic fault. Consider a simple example of 8-bit *pattern-based* signatures of four chips C1, C2, C3 and C4.

$$C1 = (1, 1, 0, 0, 1, 1, 0, 0)$$
$$C2 = (1, 1, 1, 1, 0, 0, 0, 0)$$
$$C3 = (1, 0, 0, 0, 0, 0, 0, 0)$$
$$C4 = (0, 0, 1, 1, 1, 1, 0, 0)$$

It is easy to see that the most probable systematic defects is the one indicated by the first bit of the signature, as they are common to three out of the four chips. However, the commonality measure as computed by [14] is 1/3 between every pair of the three chips C1, C2 and C4. The commonality measure between C3 and the chips C1 and C2 is 1/4, and, between C3 and C4 it is 0. By this, we see that C1, C2 and C4 have more commonness between them than C1, C2 and C3, which is not desirable from the point of grouping to identify systematic faults. Hence, the chip C4 which does not have the probable systematic fault (the first bit of its fail signature is 0) is grouped with the others which have the systematic fault. Ironically, the chip C3 which does have the probable systematic fault is not grouped with C1 and C2.

All the patterns in the fail signature vector are considered equally likely to detect the systematic faults occurring in them. Realistically, a test pattern that detects almost 90% of the chips to be faulty is more significant than the one that detects only 5% of them and thus the former better describes the systematic faults present over large volumes of chips. Similar argument holds for the scan-flops based fail signatures as not all the scan-flops have the same importance in representing the systematic faults. Hence, such a type of representation of fail signatures is however, not a very efficient way of describing the raw failure data from the ATE. In other words, quantification of the raw fail data is significantly lost by using such kind of failure representation.

Tedious Commonality Analysis: The vector size of the fail signatures varies from few hundreds to few thousand bits based on the number of test patterns applied during the test or the scan-flops at which the fault is observed. Due to this, the commonality measure calculation for every pair of chip involves sum of product calculation of huge fail signature vectors making it time-consuming. Also, since the commonality analysis proposed in [14] is inherently quadratic in the number of chips, the very purpose of reducing the run-time of VYD is lost with the presence of large *volumes of chips* to be diagnosed.

The major contribution of this paper is a linear time clustering algorithm reducing the run-time significantly. It also addresses the problem of quantification of the raw fail data much more effectively than the ones reported in literature, specifically from the point of view of detecting systematic faults, which, in turn, is the main goal of VYD.

III. PROPOSED TECHNIQUE

In this paper, we propose an efficient way of clustering the similarly failing chips by introducing *quantification* in the fail signatures for each failed chip. The quantification is provided by retaining the information regarding the number of chips that fail for a particular test pattern or at a scan-flop or a standard cell or node. This modifies the fail signatures from a fail signature vector as described in [14] to a fractional number.

We now define an *element* as a feature of the fail chip that detects the fault in it. The *element* can thus be a *test pattern* applied to the chip during testing or a *scan-flop* at which the fault is observed or a *standard cell* present in the back trace cone of the failing scan-flops or any *node* present in the back trace cone of the failing scan-flop.

	element 1	element 2	...	element M-1	element M
chip 1	1	0	...	0	1
chip 2	0	0	...	1	0
chip 3	1	0	...	1	0
chip 4	0	1	...	1	0
...	...	...	...	...	...
...	...	...	...	...	...
...	...	...	...	...	...
chip N-2	1	1	...	1	0
chip N-1	1	0	...	1	0
chip N	1	1	...	1	0

TABLE I FAIL CHIP DATA

Consequently, there can be four different types of fail signatures possible for a faulty chip, namely, *pattern based* fail signature, *scan-flop based* fail signature, *standard-cell*

based fail signature and *node based* fail signature.

The fail chip data is symbolically represented in the TABLE I by a $N \times M$ *FAIL* matrix, where N is the number of chips that failed and M is the number of instances of the same element type that detects the failures in the chips. The columns correspond to one of the four element types. They can be test patterns, scan flops, standard cells or nodes. The rows correspond to the chips that have failed during testing. If an element detects a chip to be faulty, the cell of the table corresponding to that element and the chip takes a value 1, otherwise it is 0. For example, the cell $e(i,j)$ is 1 when the j^{th} element detects i^{th} chip to be faulty during testing. From this we note that, the row corresponding to a particular chip in TABLE I is the same as the fail signature vector defined in the data mining technique proposed in [14]. The vector is of the form

$$FV_i = \{1, 0, ..., 0, 1\},\ i \in [1, N] \tag{1}$$

We now derive fail signatures for our technique from these vectors. In order to overcome the drawbacks of fail signature vectors explained in the previous section, a *weight* is assigned corresponding to every element in the fail data. This *weight* refers to the fraction of fail chips that the particular element detects. In other words, *weight* W_j of an *element* is defined as the number of fail chips that the element detects to the total number of fail chips obtained during testing.

$$W_j = \frac{1}{N} \sum_{i=1}^{N} e(i,j),\ j \in [1, M] \tag{2}$$

The *weight* ensures that the element detecting larger number of chips to be faulty has more significance than the one detecting fewer number of chips thereby attaching more importance to those elements that contribute to detecting chips with systematic defects. The new fail signature is defined as the weighted sum of fail signature vector FV_i.

$$F_i = \sum_{j=1}^{M} W_j \times e(i,j),\ i \in [1, N] \tag{3}$$

The fail signature obtained from equation (3) describes the fail data of the chip more efficiently in a single number than the one proposed in [14]. The huge vectors are replaced by a single floating-point number. The various types of fail signatures exhaustively describe the fail data of the chip. The fail signature obtained from all these four types of elements, namely, pattern, scan-flop, standard-cell and node symbolically represents the fail chip in a four co-ordinate system. Thus, with just four attributes describing the fail data of the chip, clustering of the failed chips can be effectively carried out for detecting the systematic defects. The systematic defect can lead to a fault on a net or on a standard cell which are represented efficiently by the proposed fail signature.

IV. WEIGHT-BASED GROUPING ALGORITHM

Given a set of failed chips of the same design, we now propose an algorithm that works on the fail signatures proposed in the previous section. For obtaining the fail signature, firstly the *FAIL* matrix needs to be extracted from the fail chip data. After this, the *weight* for every *element* in the matrix is calculated using equation (2). Finally, the fail signature for each fail chip is computed using the equation (3). The grouping algorithm explained here is for a single element type. In other words, the fail signature is of one dimension namely either pattern based or scan-flop based or standard-cell based or node-based.

Algorithm 1 Grouping Algorithm

Input: Fail Data of the N chips.
Output: n Clusters

N : Number of Chips
M : Number of Elements
n : Number of Clusters
$i \in [1..N]$
$j \in [1..M]$
$k \in [1..n]$
// **Step 1:** Form *FAIL* matrix
for all chip C_i **do**
  for all element E_j **do**
    if E_j contributes in detecting the fault in C_i **then**
      $e(i,j) = 1$
    else
      $e(i,j) = 0$
    end if
  end for
end for
// **Step 2:** Calculate *weight*, W_j
for all element E_j **do**
  $W_j = \{\sum_{i=1}^{N} e(i,j)\}/N.$
end for
// **Step 3:** Obtain *fail signature* F_i
for all chip C_i **do**
  $F_i = \sum_{j=1}^{M} W_j \times e(i,j).$
end for
// **Step 4:** Grouping
Obtain the unique fail signature values
for all unique fail signature value U_k **do**
  for all fail signature value F_i **do**
    if $U_k = F_i$ **then**
      Put chip C_i in $Cluster_k$
      Remove chip C_i from the list
    end if
  end for
end for

The grouping primarily involves clustering all those fail chips that have the same fail signature value. In order to do so, we first find the set of unique fail signatures and secondly cluster all those fail chips corresponding to every fail signature

value in the set. The complexity of the algorithm is of the order of *n* times *N*, where n is the number of unique fail signatures and *N* is the number of chips. The failure being systematic, large number of chips tend to cluster toward very few signature values which leads to $n << N$ and hence the algorithm is linear in the number of fail chips in the average case, thereby improving the run-time during experimentation.

V. Simulation Framework

The simulation framework is developed to mimic the occurrence of the systematic faults in the chips. The fault model for the benchmark circuits is a simple stuck-at fault model. The stuck-at faults are injected over several design files of the same benchmark circuit with the help of Mentor Graphics's tools.

A block diagram of the experimental flow is depicted in

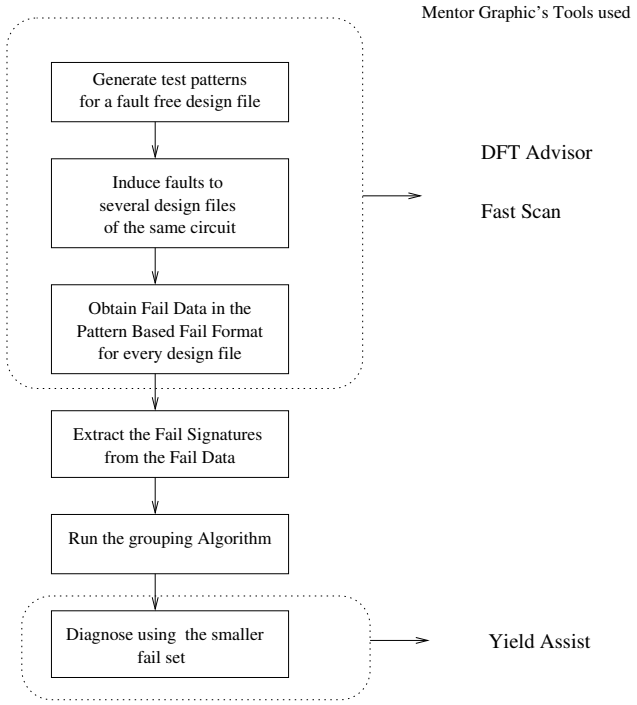


FIGURE 1. Experimental Flow

FIGURE 1. After obtaining the scan-based design from the Mentor's DFTAdvisor, the test patterns are generated for the fault-free design using the FastScan. Additionally, over 300 *fault-injected design files* are simulated for the same benchmark circuit by injecting stuck-at faults at the randomly selected nets in the circuit. These fault-injected design files represent failing chips. The fault distribution over the design files are shown in TABLE II. The typical value for the number of faults to be simulated in a design file (representing number of faults in a defective chip) ranges from 1 to 3 [16] [17].

The simulated faults (stuck-at faults a, b, c, d and e) in the TABLE II are *systematic* in the sense, they manifest

Stuck-at fault	# Fail Files
a	60
b	30
c	40
d	70
e	50
a,b	10
b,c	15
a,b,c	5

TABLE II Fault Distribution

themselves over a large number of design files. A simulated fault is said to be *random* if it manifests itself in small numbers across design files. Around 5 to 30 such random faults are injected, the number of faults in a single run also being random. During simulation, FastScan provides *fail data* for the corresponding fault-injected design files. The fail data consists of the test patterns to be applied to detect the fault as well as scan flops at which the fault can be observed. The fail data thus obtained (in the form of *fail files*) represent the fail data of the faulty chip from the ATE. The fail data is made realistic by removing fail patterns from the data.

The proposed technique extracts fail signature from these fail files and the grouping algorithm is applied. Finally, each group consists of a set of fail files failed due to the same fault occurring in them. Hence for diagnostic purposes, a subset of the fail files are used thereby reducing the run-time of the VYD methodology. With this simulation framework, our weight-based grouping algorithm as well as the clustering algorithm proposed in [14] are compared for all the benchmark circuits.

Cluster Analysis

The quality of the clustering solutions obtained from both the methodologies are compared using the parameter *overall cluster purity*. Let a clustering algorithm when applied on a dataset D provides n-clusters with the size of the cluster $Cluster_j$ being $|Cluster_j|$. Also, let $|Cluster_j|_{class=i}$ denote the number of items of class i in cluster j. The purity of the cluster is defined as,

$$purity(Cluster_j) = \frac{\max_i |Cluster_j|_{class=i}}{|Cluster_j|} \quad (4)$$

The overall purity of a clustering solution is expressed as a weighted sum of individual cluster purities,

$$purity = \sum_{j=1}^{n} \frac{|Cluster_j|}{|D|} purity(Cluster_j) \quad (5)$$

Cluster efficiency is another parameter that is used to compare the methodologies which is defined as the fraction of the chips with systematic defects that is clustered using a methodology.

VI. Experimental Results

The results of the two clustering algorithms on ITC'99, ISCAS'85 and ISCAS'89 benchmark circuits are shown in

TABLES III, IV and V. The tables compare the average *purity* obtained from the two methodologies over 30 runs of the experimental flow. The tables also show the comparison between the run-time values. The signatures used by the two methodologies are *pattern-based* fail signatures.

It can be observed that the weight-based algorithm performs better than or equally good as the Huisman et. al [14] algorithm in terms of overall cluster purity. For the rest of the circuits, the purity values obtained from weight-based closely follows the Huisman et. al [14] cluster purity. In terms of run-time, weight-based algorithm thoroughly outperforms the Huisman et. al [14] for all the benchmark circuits. The proposed linear-time algorithm in terms of number of fail chips is highly efficient in the sense that it maintains the quality of the clustering solution even in reduced run-time.

Circuit	Avg. purity for 30 runs		Time for 1 run (μs)	
	our approach	[14]	our approach	[14]
b01	0.973	0.963	3220	7420458
b02	0.900	0.893	2695	7515433
b03	0.940	0.940	4052	7513778
b04	0.907	0.900	9324	7418418
b05	0.817	0.807	8430	7425969
b06	0.977	0.967	3098	7403849
b07	0.937	0.897	6279	7512281
b08	0.853	0.833	4777	7605997
b09	0.973	0.917	4446	7543745
b10	0.987	0.957	6046	7403404
b11	0.993	0.980	8964	7302712
b12	0.940	0.920	10532	7399974
b13	0.943	0.940	5080	7375065
b14	0.977	0.967	72612	7460847
b15	0.943	0.930	44554	7292204
b17	0.897	0.897	85760	7501813
b18	0.957	0.920	92231	7538283
b20	0.993	0.923	76903	7397438
b21	0.990	0.893	84775	8462466
b22	0.993	0.937	80456	7603565

TABLE III ITC BENCHMARK CIRCUITS

Circuit	Avg purity for 30 runs		Time for 1 run (μs)	
	our approach	[14]	our approach	[14]
s1196b	0.990	0.943	13336	7492025
s1238a	0.923	0.927	13645	7348700
s13207a	0.997	0.933	24494	7428876
s1423a	0.913	0.940	6977	7493458
s1488	0.837	0.817	6739	8237472
s15850a	0.960	0.957	15083	8283926
s27	0.870	0.863	2119	8168263
s298	0.960	0.960	4292	7562654
s344	0.770	0.817	4006	7598608
s35932	0.953	0.933	5960	8638763
s382	0.993	0.957	4927	8380624
s38584a	0.997	0.953	15001	7486959
s386	0.997	0.990	20507	7502573
s400	0.930	0.883	4926	7581175
s444	0.930	0.943	19702	7809367
s510	0.917	0.930	6679	7502994
s526a	0.947	0.917	6752	7982452
s5378a	0.930	0.887	14438	7988727
s641	0.953	0.897	5753	7776690
s713	0.873	0.883	5469	7917087
s820a	0.917	0.983	9958	7618165
s832a	0.960	0.970	17531	7533694
s838	0.853	0.807	12901	7723663
s9234a	0.883	0.883	18285	7843858

TABLE IV SEQUENTIAL ISCAS BENCHMARK CIRCUITS

Circuit	Avg. purity for 30 runs		Time for 1 run (μs)	
	our approach	[14]	our approach	[14]
c1355	0.907	0.870	8300	7470744
c1908a	0.940	0.970	6544	7582367
c2670a	0.967	0.957	9257	7575636
c3540a	0.903	0.907	12112	7985589
c432	0.997	0.987	7185	7423908
c499	0.920	0.920	6347	7644481
c5315a	0.937	0.923	10253	7602392
c6288	0.957	0.960	4716	7631697
c7552	0.953	0.953	15259	7320489
c880a	0.993	0.967	7035	7514399

TABLE V COMBINATIONAL ISCAS BENCHMARK CIRCUITS

The experimental results shown in TABLES III, IV and V has been obtained using pattern-based fail signatures. The same methodology can be followed with the help of other types of fail signatures. For example, when the weight-based algorithm provides clusters with large cluster sizes using *standard-cell* based fail signature, then we can strongly claim the presence of failures due to standard cells in them. On the other hand, when there are no big clusters formed, we can safely overrule the presence of systematic defects of this type.

Validation on Real Industrial Data

The proposed technique has been proved to give remarkable results in the simulation environment for all the benchmark circuits. To validate our scheme, the weight-based grouping algorithm as well as Huisman et. al [14] algorithm [14] have been run on fail data of real *industry* hardware devices. Instead of clustering the fail chips on the exact signature values, the clusters are formed when the fail signature value occurs within a range.

The FIGURES 2, 3 and 4 provide the comparison between

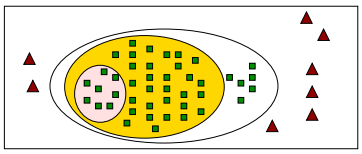


FIGURE 2. Case 1

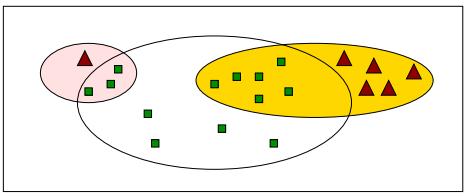


FIGURE 3. Case 2

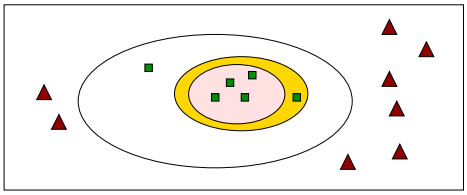
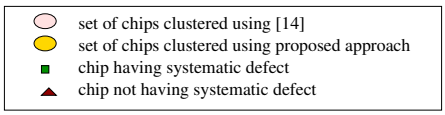


FIGURE 4. Case 3



Lot No.	# Fail Chips	# chips clustered		run-time (μs)	
		ours	[14]	ours	[14]
1	43	38	8	4474	822141
2	13	6	3	48659	473486
3	6	5	4	9729	11778

TABLE VI EXPERIMENTAL RESULTS ON REAL INDUSTRIAL FAILURE DATA

the clusters obtained from the two techniques. In the cases of lots 1 and 3, the cluster purities of both the algorithms are 100% in the sense the predominant cluster obtained from them consists only of the desired fail chips while in case of lot 2, the resultant clusters have few undesired chips. However, the weight-based algorithm is able to group more number of the desired chips than the Huisman et. al [14] algorithm. For example, in lot 1, 88.4% of the desired 43 fail chips are present in dominant cluster obtained from weight-based whereas only 18.6% of the desired chips are present in the cluster provided by Huisman et. al [14] algorithm. The efficiency is less in case of lot 2, even though our algorithm performs better than Huisman et. al [14].

The TABLE VI reveals the comparison of the run-times of both the techniques for every lot. Our experimentation has been carried out using fail chips from 2 to 5 silicon wafers in a lot. The golden reference for the cluster purity and efficiency calculations is the clusters obtained from industry's in-house diagnosis tool. The tool reveals the presence of single predominant systematic defect in every lot. The column 2 of TABLE VI gives the number of fail chips in a cluster with this predominant defect. The comparison of the run-times clearly reveals that the proposed technique is faster than the Huisman et. al [14] algorithm. With the increase in the number of fail chips that have systematic defects, the difference in the run-times can be distinctly seen. TABLE VI re-establishes our claim of reduced run-time by maintaining a better quality of cluster solutions.

VII. CONCLUSIONS

In this paper, we have proposed a weight-based grouping algorithm to cluster chips that have failed due to any systematic defect occurring in them. In doing so, the paper addresses an effective way of describing the fail information in the form of fail signatures. We have demonstrated the superiority of our approach over an existing technique in literature in terms of their run-times. Additionally, the methodology presented here has been proved to provide better quality cluster solutions for 75% of the benchark circuits in the simulation environment. It has also been shown to be equally effective on real industrial data.

REFERENCES

[1] H. Goel and D. Dance, "Yield enhancement challenges for 90nm and beyond," *IEEE/SEMI Advanced Manufacturing Conference*, pp. 262–265, 2003.

[2] M. Karthikeyan, S. Fox, W. Cote, G. Yeric, M. Hall, J. Garcia, B. Mitchell, E. Wolf, and S. Agarwal, "A 65nm random and systematic yield ramp infrastructure utilizing a specialized addressable array with integrated analysis software," *IEEE*, pp. 104–109, 2006.

[3] R. Raina, "What is dfm & dfy and why should i care?" *International Test Conference*, pp. 1–9, 2006.

[4] H.-P. Erb, C. Burmer, and A. Leininger, "Yield enhancement through fast statistical scan test analysis for digital logic," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 2005.

[5] B. Kruseman, A. Majhi, C. Hora, S. Eichenberger, and J. Meirlevede, "Systematic defects in deep sub-micron technologies," *International Test Conference*, pp. 290–299, 2004.

[6] F. Lee, "Advanced yield enhancement: Integrated yield analysis," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, pp. 67–75, 1999.

[7] M. Miller, "Nanometer yield enhancement begins in the design phase," *Electronic Design Magazine*, 2005.

[8] M. Cote and P. Hurat, "Standard cell printability grading and hot spot detection," *International Symposium on Quality Electronic Design*, 2005.

[9] R. Guldi, T. Winter, N. Sridhar, J. Smith, S. PapaRao, J. Garvin, and B. Metteer, "Systematic and random defect reduction during the evolution of integrated circuit technology," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, pp. 2–7, 1999.

[10] J. H. Yeh and A. Park, "Novel technique to identify systematic and random defects during 65 nm and 45nm process development for faster yield learning," *IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, pp. 54–57, 2007.

[11] A. L. Crouch, P. Burlison, and D. Ciplickas, "Processing high volume scan test results for yield learning," *International Symposium on Quality Electronic Design*, 2007.

[12] S. Seike, K. Namura, Y. Ohya, A. Uzzaman, S. Arima, D. Meehl, V. Chickermane, A. Kobayashi, S. Tanaka, and H. Adachi, "Early life cycle yield learning for nanometer devices using volume yield diagnostics analysis," *Asian Test Symposium*, 2006.

[13] C. Hora, R. Segers, S. Eichenberger, and M. Laousberg, "An effective diagnosis method to support yield improvement," *International Test Conference*, pp. 260–269, 2002.

[14] L. M. Huisman, M. Kasaab, and L. Pastel, "Data mining integrated circuit fails with fail commonalities," *International Test Conference*, pp. 661–668, 2004.

[15] C. Schuermyer, K. Cota, R. Madge, and B. Benware, "Identification of systematic yield limiters in complex asics through volume structural test fail data vizualization and analysis," *International Test Conference*, pp. 1–9, 2005.

[16] V. D. Agrawal, S. C. Seth, and P. Agrawal, "Fault coverage requirement in production testing of lsi circuits," *IEEE Journal of Solid State Circuits*, pp. 57–61, 1982.

[17] H. Hashempour, F. J. Meyer, and F. Lombardi, "Hybrid multisite testing at manufacturing," *IEEE International Test Conference*, pp. 927–936, 2003.

[18] M. Rehani, R. Madge, J. Teisher, D.Abercrombie, and J. Saw, "Ate data collection - a comprehensive requirements proposal to maximize roi of test," *International Test Conference*, pp. 181–189, 2004.

100KHz-20MHz Programmable Subthreshold G_m-C Low-Pass Filter in 0.18µm CMOS

S.Ramasamy, B.Venkataramani, R.Niranjini, K.Suganya
Dept. of ECE National Institute of Technology, Tiruchirappalli, INDIA
e-mail-bvenki@nitt.edu

Abstract

This paper proposes a modified, inverter based transconductor using double CMOS pair for implementation of biquad G_m-C low-pass filter with bandwidth tunable from 100 kHz to 20 MHz. This bandwidth range meets the requirements of zero IF receivers for wireless applications. Major contributions of this paper are proposal for operating the G_m stage in sub-threshold region so as to minimize the power dissipation, proposal for switching in both dummy stages and load capacitors (accumulation MOS-Capacitor) to maintain constant capacitance. The centre frequency of the filter is varied by switching in different Gm cells. The proposed filter is designed and implemented on TSMC-0.18µm CMOS process with 1.8V supply using G_m/I_d design methodology. The simulation results demonstrate the tunability of the centre frequency from 100KHz to 20MHz. The power dissipated by the filter is 12µW and 900µW at 100KHz and 20MHz respectively. The SFDR over the entire band is 57dB. The proposed approach guarantees the upper bound on THD to be -40dB for 300mVpp signal swing. The use of inverters with double CMOS pair results in 34dB higher PSRR compared to those using push pull inverter.

1. Introduction

Transconductors have a wide range of applications in the area of analog signal processing [1], [2]. Continuous time filters implemented with transconductance amplifiers and capacitors are known as G_m-C or OTA-C filters and are quite popular for a host of applications such as IF filters, hard disk drive, linear phase filters, LC-oscillators and RF filters. A software defined radio (SDR) demands multitude of standards capable of adjusting their configuration and reception mode depending on the standard and demanded quality of service. In this context the flexible analog baseband filtering is required.

A number of architectures have been proposed in the literature for implementing the transconductor. Push-pull inverters are proposed in [3] for realizing the transconductor. This does not have any internal node and results in large bandwidth. However, for realizing programmable filters, this scheme requires the power supply voltage to be varied. This is not suitable for low voltage applications and it results in poor power supply rejection ratio (PSRR). To solve this problem, floating battery implementation is proposed in [4]. But this scheme requires a large bias resistor, which introduces an additional pole in the region of interest. In [5], a novel switching technique is proposed in Nauta's transconductor to change the transconductance and the input capacitance independently.

The transconductor using double CMOS pair is proposed in [6]. A digitally programmable biquad G_m-C band pass filter for FM application with independent centre frequency tuning and Q-tuning is proposed in [7]. The scheme in [7] requires two supply voltages, one for the filter core and the other for bias voltage circuitry meant for tuning. In this paper, sub-threshold G_m stage using double CMOS pair is proposed to minimize the power dissipation and also to use a single supply voltage.

This paper is organized as follows. Section 2 explains the structure of a sub-threshold G_m stage using double CMOS pair. Section 3 presents the structure of the proposed digitally tunable second order low pass filter. The F-tuning used are also discussed. The filter design using G_m/I_d method is illustrated in Section 4. The simulation results are given in Section 5 followed by the conclusions in Section 6.

2. Sub-threshold G_m stage using double CMOS pair

In this section, the operation of the transconductor using double CMOS pair is described first. The operation of the common mode control and dc gain enhancement of the transconductor is described next. The two-transistor circuit shown in Fig.1 is referred to

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.19

as the CMOS pair. It may be considered to be a single transistor.

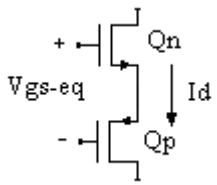


Fig 1. CMOS pair

Assuming that both transistors are operated in weak inversion saturation (sub-threshold) region, the current I_D for single PMOS/NMOS can be written as [9],

$$I_D = 2.n.k.\Phi_T^2.\exp\left(\frac{V_{GB}-V_{T0}}{n.\Phi_T}\right)\left[\exp\left(\frac{-V_{SB}}{\Phi_T}\right)-\exp\left(\frac{-V_{DB}}{\Phi_T}\right)\right] \quad (1)$$

where 'n' is sub threshold slope factor, $k = \beta .(W/L)$ is transconductance parameter, $\beta = \mu.C_{ox}$ is the process gain factor, 'μ' is mobility and 'C_{ox}' is the oxide capacitance of the transistor. The thermal voltage 'Φ_T' is 25.9 mV at room temperature, V_{GB}, V_{SB}, V_{DB} are gate, source, drain voltages w.r.t. bulk respectively, V_{T0} is the threshold voltage when V_{SB} is Zero. When $V_{DB} > 5\ \Phi_T$ and $V_{SB} = 0$, an approximate expression of I_D for single transistor is given by (2).

$$I_D = 2.n.k.\Phi_T^2.\exp\left(\frac{V_{GS}-V_{T0}}{n.\Phi_T}\right) \quad (2)$$

The current I_1 and I_2 in the double CMOS pair is given by (3) and (4) respectively

$$I_1 = 2k_{eff}\, n\phi_t^2 \exp\left(\frac{v_{GSeq\,1}-v_{teq\,1}}{n\phi_t}\right) \quad (3)$$

$$I_2 = 2k_{eff}\, n\phi_t^2 \exp\left(\frac{v_{GSeq\,2}-v_{teq\,2}}{n\phi_t}\right) \quad (4)$$

where, $\frac{1}{\sqrt{K_{eff}}} = \frac{1}{\sqrt{K_n}} + \frac{1}{\sqrt{K_p}}$

and $V_{T\text{-}eq} = V_{Tn} + |V_{Tp}|$

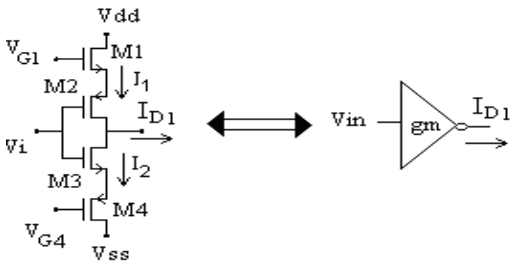


Fig. 2. Double CMOS pair

The circuit in Fig.2 shows the double CMOS pair [6], which acts as a transconductance cell. Assuming that all the MOS devices are operated in the weak inversion saturation region and neglecting the channel length modulation, the output current can be expressed as,

$$I_{D1} = I_1 - I_2 = 2k_{eff} n\phi_t^2 * \left[\exp\left(\frac{v_{GSeq1}-v_{teq1}}{n\phi_t}\right)-\exp\left(\frac{v_{GSeq2}-v_{teq2}}{n\phi_t}\right)\right] \quad (5)$$

where, $v_{GSeq1} = v_{g1} - v_i$, $v_{GSeq2} = v_i + v_{g4}$,

$v_{teq1} = v_{tn1} + |v_{tp2}|$, $v_{teq2} = v_{tn3} + |v_{tp4}|$

Assuming that the n-well process is used, the P-MOS transistors can have their bulks connected to their own source terminals and this eliminates the body effect. The bulks of M_1 and M_3 should be connected to the most negative supply (Vss), resulting in a rise in threshold voltage. This bulk effect causes the relationship between V and I to be non-linear. However, when the CMOS pair is used in Nauta's structure, this problem is eliminated as explained below.

I_{D1} and I_{D2} are the currents coming out of g_{m1} and g_{m2} (fig. 4) respectively. The differential current coming out of the G_m cell $I_0 = I_{D1} - I_{D2}$ is given by

$$I_0 = -4k_{eff} n\phi_t^2 * \left[\exp\left(\frac{v_{GSeq\,1}-v_{teq\,1}}{n\phi_t}\right)-\exp\left(\frac{v_{GSeq\,2}-v_{teq\,2}}{n\phi_t}\right)\right] \quad (6)$$

and g_m can be derived by taking partial derivative of (5) w.r.t V_{GSeq} and is given by

$$g_{m1} = \frac{I_{D1}}{n\phi_t} \quad (7)$$

Thus the g_m can be varied by varying the bias voltage V_{g1}, V_{g4} and the aspect ratio of the MOS transistors. In order to ensure that all transistors in g_m cell remain in the weak inversion saturation region, the input voltage Vi and bias voltage V_G must satisfy the inequalities given in (8)

$$v_{g1} - v_{teq1} \le v_i \le v_{g2} + v_{teq2} \quad (8)$$

V_{G-} and V_{G+} should be lesser than or equal to V_{ss} and V_{dd} respectively.

2.1. Common mode control and DC gain enhancement

To operate filters at high frequencies, we need a transconductor with high DC gain of at least 40 dB and the parasitic poles should be located far from the cutoff frequency [3]. To increase the DC gain, the negative resistance loading is proposed in [3]. Let us consider

two transconductance cells which are connected in a cross coupled manner as shown in Fig. 3.

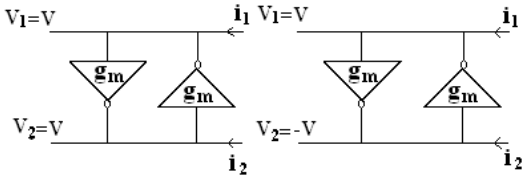


Fig. 3. Cross coupled g_m cell

Applying a common mode voltage V at both nodes results in output currents $i_1 = i_2 = g_m V$, yielding common mode resistance $R_{ocm} = 2V/(i_1+i_2) = 1/g_m$. Application of differential voltage at the nodes results in output currents $i_1 = -g_m V$, $i_2 = g_m V$. The differential mode resistance is $R_{odm} = 2V/(i_1-i_2) = -1/g_m$. Thus the differential signal sees negative load resistance.

The structure of a complete transconductor block (G_m Block) using CMOS pair (g_m cell) with differential input and output is shown in Fig. 4. This is similar to Nauta's transconductor [3], but in this design, each g_m cell is constructed with a double CMOS pair and g_m tuning is done by combining the outputs of multiple transconductors using the switched transconductance cell technique [8]. The g_m cell 1 and g_m cell 2 act as the main transconductor, g_m cells 3-6 provide common mode control and dc gain enhancement.

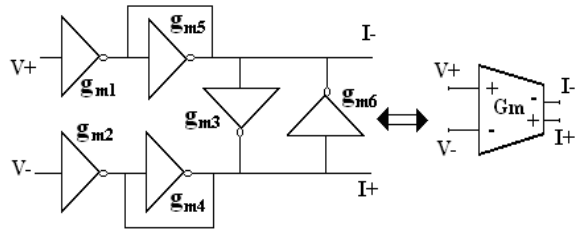


Fig. 4. CMOS Pair based balanced Gm block

This transconductor has a differential architecture made of two identical sub circuits: g_m cells 1,5,6 and g_m cells 2,4,3 respectively. Thus all equations which correspond to the first set of g_m cells hold good for the second set as well, with the indexes in the given order. The common-mode output resistance at the node I- is,

$$R_{ocm} \cong 1/(g_{ds1}+g_{ds5}+g_{ds6}+g_{m5}+g_{m6}) \qquad (9)$$

while the differential-mode output resistance at the node I- is,

$$R_{odm} \cong 1/(g_{ds1}+g_{ds5}+g_{ds6}+g_{m5}-g_{m6}) \qquad (10)$$

where, $g_{mi} = 2(g_{mni} * g_{mpi}) / (g_{mni} + g_{mpi})$

$g_{dsi} = 2g_{dspi} + 2g_{dsni} \approx I_{Di} / V_{Ai}$

g_{mn}, g_{mp} denote the g_m of NMOS and PMOS transistor respectively. Similarly g_{dsp}, g_{dsn} denote the output transconductance of NMOS and PMOS transistors respectively. I_{Di} and V_{Ai} are the drain current and the early voltage of the i^{th} transistor.

Assuming that for all the transconductance cells, $g_{dsi} = g_d$ and $g_{mi} = g_m$, the common-mode dc gain (A_{cm}) and differential-mode dc gain (A_d) at the output nodes are computed to be [3],

$$A_{cm} = \frac{g_m}{3g_d + 2g_m} \qquad (11)$$

$$A_d = \frac{g_m}{3g_d} \qquad (12)$$

Since A_{cm} is less than unity, common-mode stability is maintained. The differential mode gain can be boosted by choosing $g_{m5} \cong g_{m6} - (g_{ds1}+g_{ds5}+g_{ds6})$.

3. Digitally tunable second order Butterworth low pass G_m-C filter

In programmable continuous time filters, the center frequency and the quality factor of the filter can be tuned by varying G_m, which in turn is controlled by changing either the bias current or the device dimensions. Unary weighted G_m cells including dummy elements are connected in parallel to realize a digitally programmable transconductor. Programmability using a parallel connection of conventional differential pairs has been already reported in [8]; however, these structures are not suitable for low-voltage supply. In this paper, the centre frequency (f_c) of the filter is tuned by varying the no. of transconductance cells used and also by varying the value of the capacitor used. A compact dummy based switching scheme is proposed to ensure linear frequency tuning and constant dynamic range irrespective of the no. of g_m cells, switched in and out. This is achieved by maintaining input capacitance to be constant. Capacitors are realized by operating MOS transistor in accumulation region. This scheme dispenses with the need for switches in the signal path. The transistors M_1 and M_4 of the double CMOS pair (Fig.2) acts as switches to include or exclude the g_m cell to achieve the desired programmable transconductance values.

The differential G_m-C realization of second order low pass filter structure based on double CMOS pair, with digitally assisted centre frequency tuning (F-tuning) is shown in Fig. 5. In this parallel resonance circuit, G_{m1} is the V-I converter, the resistor is realized by G_{m2}, the inductor is realized by the Gyrator (G_{m3},G_{m4} and C_2) and C_1 is the capacitor of resonant circuit.

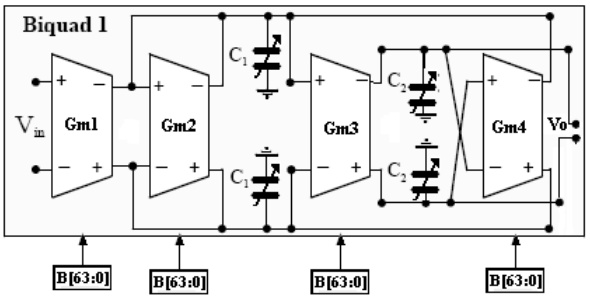


Fig. 5. Schematic diagram of tunable second order low pass filter

The low pass transfer function of the above biquad structure [2] is given by,

$$H(s) = G_{m1}/(s^2C_1C_2 + sC_2G_{m2} + G_{m3}G_{m4}) \quad (11)$$

From (11), the center frequency and the quality factor, Q, are given by,

$$\omega_0 = \frac{\sqrt{G_{m3}G_{m4}}}{\sqrt{C_1C_2}} \quad (12)$$

$$Q = \left(\frac{1}{G_{m2}}\right)\frac{\sqrt{G_{m3}G_{m4}C_1}}{\sqrt{C_2}} \quad (13)$$

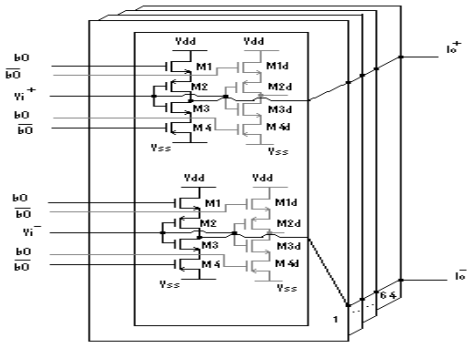


Fig.6 Programmable transconductor cell with dummies

From (12), the center frequency of the filter can be varied either by the constant-C or constant-G_m method. In constant-C technique, the load capacitance is maintained constant and the value of G_m is changed to alter the centre frequency of the filter. In constant-G_m technique, G_m is kept constant and the value of load capacitance is changed to alter the centre frequency of the filter. Detailed analysis of these two approaches is carried out in [8] based on noise, total capacitance required and power dissipation. It suggests that constant-C approach is suitable for tunable filter realizations. We follow the constant-C approach. To get Butterworth response (Q=0.707), the value of $G_{m2} \approx \sqrt{2}\, G_{m1}$ [5]. The value of G_{m1} determines the DC gain of the filter. The transistor level representation for g_{m1}/g_{m2} with dummies is shown in Fig. 6. From Fig.6, it may be noted that when b0=1 (+0.9v) and b0'= 0 (-0.9v) the main transconductor will be switched on and the dummy will be in off state and vice versa. Thus capacitance at the input is maintained constant over the entire bandwidth. In Fig. 5, G_{m3} and G_{m4} are constructed by fully balanced architecture as in Fig. 4. G_{m1} and G_{m2} blocks uses only main transconductor cells g_{m1} and g_{m2} of Fig 4.

3.1. F-tuning

The centre frequency is tuned by suitable switched transconductance cell. The digital controller generates the respective bit streams depending on the cutoff frequency which in turn decides the number of cells to be switched in. Sixty four unary g_m cells are used to cover the entire bandwidth. The dimensions of M_1-M_4 transistors (N-MOS / P-MOS) in double CMOS pair are 1.2µm/0.36µm. The capacitor of 1pF is realized by MOS capacitor. When all the cells are switched in, the maximum centre frequency tuned is only 17.6MHz. The output parasitic capacitance increases with more number of g_m cells. To vary the bandwidth in the range 17.6MHz - 20MHz, the capacitance at the output terminals is varied by using switchable MOS capacitors. The source, drain and bulk terminal shorted together forms one end of the capacitor (gnd). The gate forms the other terminal of the capacitor. If Vgb>Vt then the MOS transistor will act as capacitor (Accumulation).

Table 1 summarizes the F-tuning procedure for the entire band. Table 2 lists the dimension of transistor used for realizing the capacitor.

Table.1. F-Tuning Summary

Sl.No	No.of g_m cells	G_m (µS)	f_c (MHz)
1	1	2.8	.102
2	32	70	1
3	48	112	11.2
4	64	188	17.6

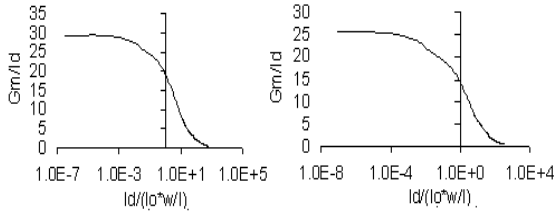
Table.2. MOS capacitor

Sl.No	Dimension of Transitor(µm)	Capacitance (pF)
1	21.6/2.16	0.1
2	21.6/4.32	0.2
3	21.6/6.48	0.3
4	21.6/8.64	0.4
5	21.6/10.8	0.5
6	21.6/12.96	0.6
7	21.6/15.12	0.7
8	21.6/17.28	0.8
9	21.6/19.44	0.9
10	21.6/21.6	1

4. Filter design

Majority of methods used for analytical synthesis of analog circuits assume that the MOS transistors are either in strong inversion or weak inversion region. The design methodology, based on the G_m/I_d characteristics [10] allows a unified synthesis technique which is valid in all regions of operation of the MOS transistor. Detailed explanations on G_m/I_d method are given in [10].

The extracted G_m/I_d versus $I_d/(I_0W/L)$ curves for both NMOS and PMOS transistors of TSMC 0.18µm process are shown in Fig. 7(a) and 7(b). I_0 is the specific current [10] given by $I_O = 2nKU_T^2$.
where K is the process parameter(μC_{OX}), U_T is the thermal voltage and n is the slope factor[10].



(a) NMOS (b) PMOS

Fig. 7. G_m/I_d Vs I_{norm} curve for transistor

Here G_m/I_d value (>25) is chosen such that the transistor is in weak inversion region.
The procedure e used in the design of the G_m stage .

1. Assume the unity gain frequency f_T and the load capacitance C_L. The required G_m can be calculated using $G_m=f_T2\pi C_L$. (At f_{cmax}, based on noise specification capacitance required is calculated as 1pF).
2. Assume G_m/I_d based on the region of operation of the transistor and calculate the value of drain current as G_m /(G_m/I_d).
3. Find the value of normalized drain current $I_d/(I_OW/L)$ from the G_m/I_d curve corresponding to the assumed G_m/I_d.
4. Calculate the W/L corresponding to this normalized current.
5. Once the W/L values are determined, their lengths are chosen based on both gain and area requirement and then the corresponding width is found. Here the length of the transistor is chosen as 0.36µm. The transistor dimension of the unit g_m cell is 1.2 µm/0.36 µm.

5. Simulation results

This section presents the pre-layout simulation results of the low pass filter, obtained using Mentor Graphics tools, Eldo and ezwave for the TSMC 0.18µm CMOS technology model. Simulated results are summarized in Table 3. THD of -41dB is obtained for a signal swing of $300mV_{pp}$ with PSRR of 55 dB at 20MHz. The DC transfer characteristics (I_{od} vs V_{id}) for the various bias voltages are shown in Fig.8. From this figure, it can be observed that the output current is linearly proportional to differential voltage of up to ±100mV. Simulated transconductance values versus differential input voltage (V_{id}) for various bias voltages are shown in Fig.9**.** PSRR characteristics for the supply voltage of ± 0.9 V is shown in Fig. 10. Fig. 11 shows the simulated frequency response of the low pass filter as the centre frequency is varied from 100 KHz to 20 MHz. The simulated results of MOS capacitor is shown in Fig 12. SFDR plot for minimum and maximum frequencies are shown in Fig. 13 for 250mVpp input signal. From the simulation results, the power consumed by the filter is found to be less than a 1mW which is ten times lesser than that of [5].

A monte Carlo simulation for thousand runs was carried out for evaluating the sensitivity of common mode range and centre frequency of the low pass filter with 5% Gaussian variation simultaneously for parameters W/L, t_{ox} and $V_{th.}$ The variation of centre frequency is found to be less than 4%.

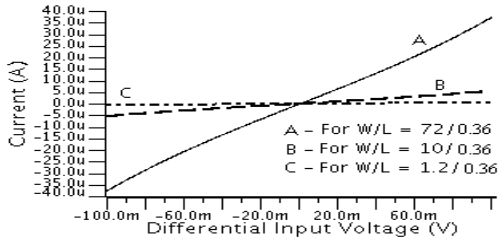


Fig 8. DC Transfer characteristics for Differential input voltage

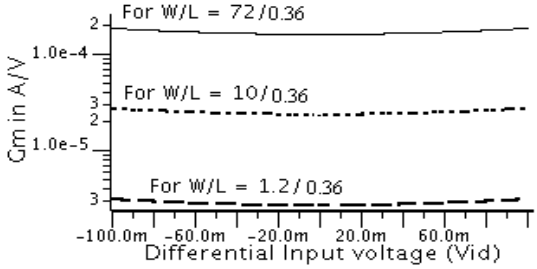


Fig.9. Transconductance values for three different transconductance cells

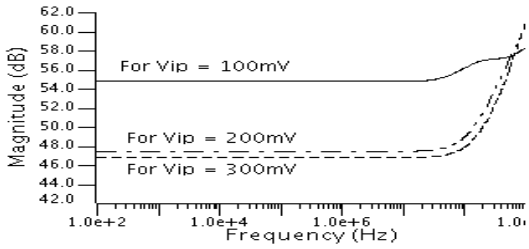


Fig. 10. PSRR characteristics (± 0.9V)

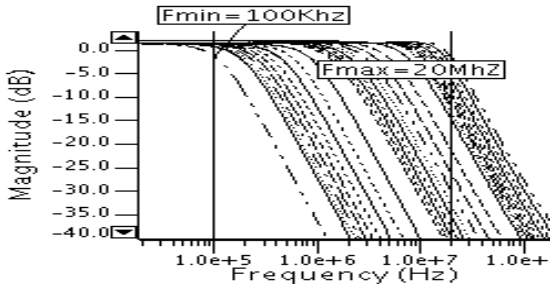


Fig.11. Low pass filter response for F-tuning

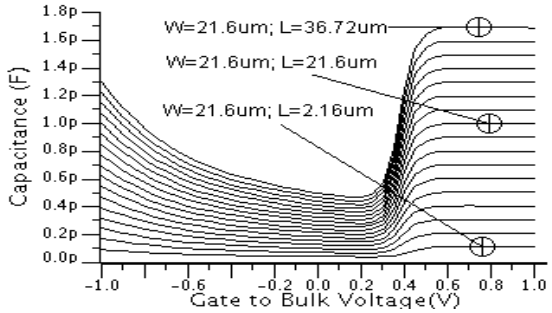


Fig 12. MOS capacitor in accumulation region

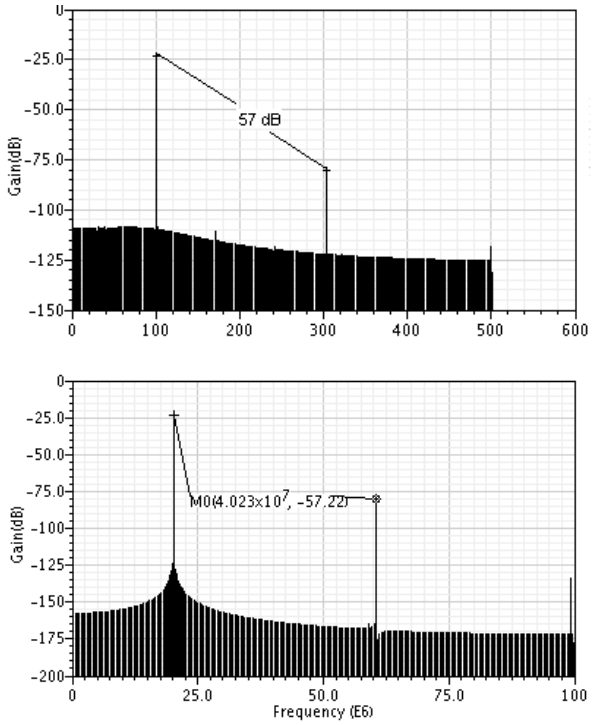


Fig 13. SFDR plot for 100Khz and 20MHz

Table 3. Simulation results summary

Technology	TSMC 0.18 µm CMOS process
Supply voltage	± 0.9V for Filter
Filter type	second order, Butterworth
Frequency tuning	100KHz -20MHz
THD @300mV$_{pp}$	-40.7dB
Group delay	<2ns
PSRR (± 0.9V) at 20MHz	55dB
Input referred noise @20MHz	27nv/sqrtHz
SFDR @20MHZ	57dB
Power consumption	15 µW – 900 µW
Load capacitance	0.1pF – 1pF

6. Conclusions

A tunable second order G_m-C low pass filter based on double CMOS pair is implemented in TSMC 0.18 µm digital CMOS process. Thanks to the sub-threshold G_m stage and compact dummy based switching scheme, more than ten fold reduction in power and fifteen fold reduction in area are achieved respectively. The designed low pass filter features a good center frequency tuning range from 100KHz to 20MHz which covers the frequency range corresponding to analog baseband filters used in the physical layer of various the wireless networks such as WLAN a/b/g , UMTS, Bluetooth, GSM and CDMA.

References

[1] M.Ismail, and T.Fiez, *Analog VLSI Signal and Information Processing*, McGraw-Hill, New York, 1994.

[2] David A.Johns, and Ken Martin, *Analog Integrated Circuit Design*, Wiley & Sons Inc, 1997.

[3] B. Nauta, "A CMOS transconductance-C filter technique for very high frequencies", *IEEE J. Solid-State Circuits*, vol. 27, pp. 142–153, Feb.1992.

[4] F. Munoz, A. Torralba, R. G. Carvajal, and J. Ramirez-Angulo, "Two new VHF tunable CMOS low- voltage linear transconductors and their application to HF gm-C filter design", in *Proc. ISCAS*, May 2000, pp.V-173–176.

[5] P. Crombez, J. Craninckx, Piet Wambacq and M. Steyaert, "A 100-kHz to 20-MHz Reconfigurable Power-Linearity Optimized Gm–C Biquad in 0.13µm CMOS", *IEEE Transactions on Circuits and Systems* II: EXPRESS BRIEFS, VOL. 55, NO. 3, MARCH 2008, 33, pp. 224- 228.

[6] C.S. Park, and R.Schaumann, "A High-Frequency CMOS Linear Transconductance Element", *IEEE Transactions on Circuits and Systems*, 33, pp. 1132- 1137, 1986.

[7] S.Ramasamy, B.Venkataramani, K.Anbugeetha "VLSI Implementation of a digitaly tunable Gm-C Filter with double CMOS pair" 21st International Conference on VLSI Design, January 2008, pp. 317-322.

[8] Shanthi Pavan, Yannis.P. Tsividis, and Krishnaswamy Nagaraj, "Widely programmable high frequency continuous time filters in digital cmos technology",*IEEE Journal of Solid-State Circuits*,Vol-35,April 2000.

[9] Yannis Tsividis, Mixed Analog Digital VLSI Devices and Technology, McGraw-Hill, New York, 1995, pp. 66.

[10] Daniel Foty, David Binkley, and Mathias Bucher, "Starting Over:G_m/Id -Based MOSFET Modeling as a Basis for Modernized Analog Design Methodologies", *Nanotech 2002*, Vol.1, Chapter 13, pp. 682 – 685.

Session 2A

Analog and Mixed Signal I

A 20MS/s 5.6 mW 6b asynchronous ADC in 0.6μm CMOS

Theja Tulabandhula*
HPA, Texas Instruments India, *and*
Dept. of Electrical Engineering, IIT Kharagpur
t.theja@iitkgp.ac.in

Yujendra Mitikiri
NyADC, High Performance Analog Group
Texas Instruments India
yuju@ti.com

Abstract

The design of an N-comparator based asynchronous Successive Approximation Analog-to-Digital Converter (SAR ADC) is described (with $N = 6$) working at 20 MS/s and consuming only 5.6 mW for low power high speed applications like communication systems. Resetting the comparators in each conversion cycle is avoided (reducing power consumption compared to [1]) and only N latches are used overall (incl. comparator latches) for the output code. Further using only N comparators instead of $2^N - 1$ as in [2], leads to huge savings in terms of area at comparable power consumption. For example, a saving of $\sim$90% comparator area is achieved for the 6 bit ADC design when compared to the design in [2].

1. Introduction

There is a constant need for high speed data converters ([3],[4]) in communication systems with low power consumption also being a major concern. The resolutions demanded are about 4-8 bits (e.g., UWB applications [5] & [6]). Asynchronous ADC (Analog to Digital Converter) design seems to be a promising way to meet this goal and has been pursued actively in the past few years ([7],[8],[9]). Of particular interest are those architectures which evolve from the SAR (Successive Approximation) scheme as in [1] and [2]. Exploring asynchronous conversion in this scheme has the advantage of retaining some of its simplistic features and at the same time one can design a class of ADCs whose performance matches the flash or the folding flash versions. A flash ADC requires as many comparators in a conversion operation as the number of quantization levels and thus taxes area and power exponentially with increasing resolution.

In this paper, an N-comparator based asynchronous SAR ADC is proposed which strikes a balance between single comparator [1] and $2^N - 1$ comparator [2] based asynchronous designs. The single comparator based ADC needs its comparator to be reset after every comparison requiring a complex clock generation block and extra resetting time as a result. The $2^N - 1$ comparator based design requires calibration and trimming to achieve all $(2^N - 1)$ embedded thresholds at its comparators and its resolution cannot be increased without an accompanying exponential increase in area. On the other hand, the N comparator based ADC proposed here doesn't require resetting of the comparators and thus can be much faster in operation. Also, this asynchronous design scales linearly with resolution facing only the same problems as a regular SAR ADC. It offers comparable performance benefits as the $2^N - 1$ design without having to lose much area. The proposed design makes use of minimal number of latches and doesn't need an encoder to output the codes. Note that the maximum input clock rate required will be the same as the conversion rate, reducing power and complexity when compared to similar synchronous designs.

In Section 2, the new asynchronous architecture along with the single and $2^N - 1$ comparator architectures is explained. Section 3 provides the implementation details of the proposed architecture for $N = 6$. Simulation results are briefed in Section 4 and finally Section 5 presents the concluding remarks.

2. Asynchronous ADC

An N bit synchronous SAR ADC which has a conversion rate of M samples/second generally needs a clock of at least $(N + 1) * M$ Hz. It works in two phases: signal track and conversion phases. In the signal track phase (1 cycle), the input is applied to the ADC. In the succeeding conversion phase (N cycles), each bit of the output code is resolved in a binary fashion mapping to one of the $2^N - 1$ quantization levels. It generally has one comparator which is reused in each of the N comparison cycles of the conversion phase. At least once in each conversion phase (i.e. in each ADC operation), the inputs to the comparator will be so close that they will cause a high resolving time. To ensure proper functioning, the cycle of the system clock

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.56

$((N+1) * M$ Hz) should be greater than this time for minimum resolving. This clock period has to be maintained even if some of the comparisons have settled beforehand (this happens when the input difference is large) thus losing potential time savings. This limitation is done away with in an asynchronous approach, where as soon as each comparison resolves one bit of the output code, it triggers the execution of the next comparison thereby saving time. A first order upper and lower limit on the time savings when such a configuration of dynamic comparators is used is shown in [1] and elaborated in Section 2.2 where it can be seen that the order of the savings is almost $1.5 - 2$ times. The asynchronous approach also doesn't require an input clock greater than the conversion rate (M) itself. A start signal is still required for the entire analog to digital conversion process to begin in both asynchronous and synchronous cases. This can be the input clock edge itself in the asynchronous case.

A flash is a true asynchronous ADC since it generates the output code in one clock span. But due to the thermometric structure where input is compared with all the $2^N - 1$ reference levels, it consumes more power than a SAR ADC which does N comparisons over a period of N cycles (in the synchronous case). Two completely different variations of the SAR scheme retaining the N sequential comparisons/sample feature implemented in an asynchronous way are reported in [1] and [2]. We also report an architecture falling into this class. A comparison of [1] and [2] with the proposed architecture is done next. Such a comparison is justified even though their architectures might differ in some aspects because they still take similar times to perform the conversion (with N sequential comparisons) assuming all else being same.

2.1. Single, N and $2^N - 1$ comparator Architectures

Figure 1 shows the asynchronous SAR ADC proposed in [1]. In addition to the standard charge-redistribution DAC, SAR logic block and a comparator, it also has a ready signal generator, a multi-phase clock generator and additional latches. Each comparison output is stored in the additional latches, and a ready signal is generated simultaneously. This drives the multi-clock generator which apart from resetting the comparator also generates the signals used to control the SAR logic block and prepares the system for the next comparison.

In [2], the limitation of using a single comparator is overcome by employing a binary tree of $2^N - 1$ comparators with embedded threshold as shown in Figure 2. Input is applied to all the comparators all the time. The root comparator is turned on by the external start signal (same as the input clock). Depending on the comparison it enables either the

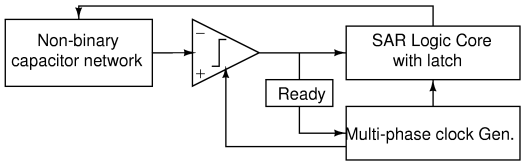


Figure 1. Single comparator asynchronous SAR ADC architecture

left child comparator or the right child comparator. Once one of the child comparators in the tree is turned on, it starts doing the next comparison making use of the embedded thresholds and then triggers one of its children. Embedded threshold in each of the $2^N - 1$ comparators is achieved by using intentional transistor mismatch and loading capacitor mismatch ([5],[10]). The N comparators which are turned on during the conversion phase are reset together at the start of the succeeding signal track phase (i.e. next input clock edge).

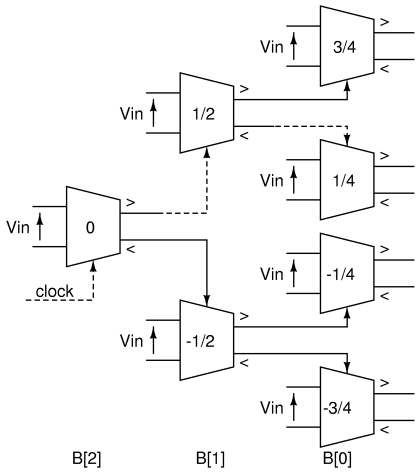


Figure 2. $2^N - 1$ comparator asynchronous SAR ADC architecture

The architecture proposed in this paper consists of a self clocked chain of N comparators and a standard charge-redistribution DAC as shown in Figure 3 (where $N = 3$). The DAC is controlled by a latchless SAR logic core. The input is applied to all the comparators simultaneously and the first comparator is turned on by the external start signal (again same as the input clock). Once this comparator resolves, it generates a next-state trigger signal which turns on the succeeding comparator in the chain. It also causes a change in the combinational SAR logic and steers the charge redistribution DAC accordingly. The output of each dynamic comparator gives one bit of the digital code

and no further encoders or additional latches are necessary. The design thus does not require high complexity calibrations to set voltage thresholds as in the $2^N - 1$ case ([2]). Since it also doesn't require its comparators to be reset right after their comparisons are done, no additional resetting logic needs to be incorporated along with saving of resetting time. In [2], power consumption is made the same as that required by a typical SAR ADC by turning on only N comparators out of the $2^N - 1$ in each analog-to-digital conversion cycle. Though in this case power scales linearly with resolution, area on the other hand still scales exponentially. In the proposed N-comparator architecture however, both power and area scale linearly with resolution. Note that the ADC in [2] looks closer to a flash configuration than a SAR configuration at a first glance. Nonetheless, use of embedded thresholding seems to be the only diversion from a conventional SAR design too. All the three configurations described and compared here take a sum of N comparison times ($\sum_{i=0}^{N-1}(t_{i^{th}comparison})$) to do a conversion unlike the flash architecture where conversion rate is determined by the slowest comparison time ($max_{i=0,\ldots,2^N-1}\{t_{i^{th}comparison}\}$).

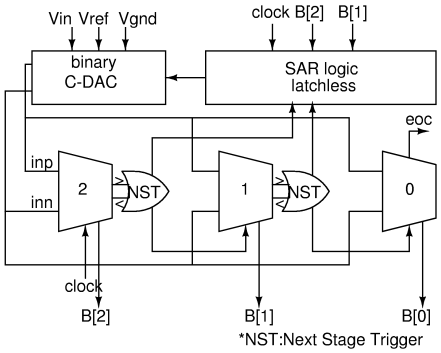


Figure 3. Proposed N comparator asynchronous SAR ADC architecture

2.2 Conversion times

If the relation between input voltage V_{diff} ($= inp - inn$) to a comparator and the resolving time T_c is given as ([1]):

$$T_c = K * ln\frac{V_{FS}}{V_{diff}} \tag{1}$$

Where V_{FS} is the full scale voltage at input, then, one can write the synchronous and asynchronous conversion times (T_{syn} and T_{asyn}) as:

$$T_{syn} = N * K * ln\frac{V_{FS}}{min(V_{diff}(i))} \tag{2}$$

$$T_{asyn} = \sum_{i=0}^{N-1} K * ln\frac{V_{FS}}{V_{diff}(i)} \tag{3}$$

Their ratio for the best case input (staircase like convergence in each step) can be shown to be

$$\frac{T_{asyn}}{T_{syn}} = \frac{1}{2} + \frac{1}{N+1} \tag{4}$$

which means that, for high conversion steps (resolution N), the speed up is almost 2 times theoretically. For N = 4, 5 and 6 this value is 0.70, 0.67 and 0.64 respectively. From a different viewpoint, this is same a saying that in the best case for 4 bits, asynchronous SAR ADCs are only 2.8 times slower than a flash ADC rather than 4 times like a synchronous SAR. Worst case and average speed up can also be worked out similarly.

3. Circuit Implementation Details

The asynchronous ADC proposed here is designed for $N = 6$. It consists of a passive T/H followed by a charge-redistribution capacitive DAC. The top plate of the capacitor array is connected to one input port of all the 6 comparators and the bottom plates of the individual capacitors are connected to three switches each (which in turn are connected to the output of the T/H (v_{in}), v_{ref} and v_{gnd}). The next stage triggering signal is generated from the *'less than'* and *'greater than'* outputs of each comparator using a NOR gate. In addition to enabling the succeeding comparator, this trigger signal is fed into the combinational SAR logic. Bitlines are resolved using the *'less than'* and *'greater than'* signals and serve as the only 'registers' in this SAR scheme as shown in Figure 3 (example with $N = 3$).

3.1. Comparator and Next-Stage Trigger

Each comparator triggers a succeeding comparator in the chain as soon as it resolves a bit as shown in Figure 4 for the actual design with $N = 6$. A capacitive DAC (see 3.2) connects to the inp and inn terminals of each of the comparators. Note that, the input load capacitance is reduced from being $\propto 2^N - 1$ to N from the design of [2]. All the comparators are dynamic with cross-coupled inverters. A pmos driven implementation is chosen similar to [2],[11] and [12] with a few variations. The comparator circuit diagram is shown in Figure 6. When the enabling signal ϕ is HI, both the latch outputs are pulled down to ground. An nmos is kept at the top of the comparator as in [2] to pull that node to a deterministic value and also to linearize the input capacitance. The pmos pair is kept matched and embedded thresholding is completely done away with. Embedded thresholding is incompatible with the proposed architecture and necessitates the use of $2^N - 1$ comparators

like a flash ([5]) or 2^{N-1} comparators as in folding flash schemes ([13],[14]). When 'ϕ' goes LO, the pmos at the top is turned on enabling the latch in the process. Further, because of the imbalance at the input terminals the currents through the two branches in the comparator change, and the latch resolves to a logic state. The comparator in [1] is accompanied by a preamplifier which has been dispensed with in this design, lowering power consumption further. The Next-Stage Trigger is a logic signal which can differentiate between the outputs (LO,LO) in the reset phase with any other state. Note that, if the comparator doesn't go into metastable state, its latch outputs will resolve into either (HI,LO) or (LO,HI) depending on the input voltage imbalance. Even when they have entered into a metastable state as shown in Figure 5, the input common mode voltage ensures that both the outputs of the latch rise from (LO,LO) to some value above LO determined by circuit conditions. A NOR gate with input threshold below this value helps in giving out a trigger signal in this case along with the normal cases (this is similar to NAND used in [1]). The comparator may remain unresolved but is prevented from stalling the chain. The bitline will reflect a previously resolved bit till the comparator makes a decision. This decision might not be the same as the decision which was assumed just before triggering the next comparator ($\sim 50\%$ of the times). The comparator which was sensed to be in metastable state thus needs to be disabled from taking a decision. This can easily be done using an additional logic gate (not shown here).

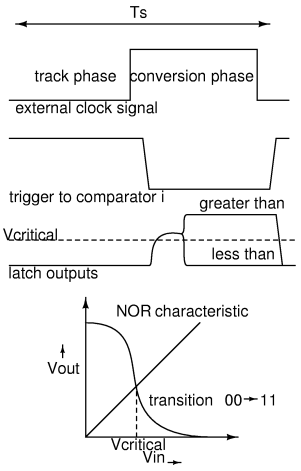


Figure 5. A timing diagram showing the operating principle of next stage triggering

There are high threshold buffers at the two outputs of the comparator depicted in Figure 6 to drive the bitline as well as the succeeding comparator through the NOR gate. High threshold avoids bus contention or instances where both the 'bitline nmos transistors' get turned on. [2] uses additional OR encoders to get the final bitline (since there are $2^N - 1$ comparators and only N final bitlines corresponding to a N bit output code). For a 6 bit design of this type, the OR encoder ties up 32 bitline nmos transistor pairs of the 32 (2^{N-1}) comparators together at the LSB stage. And only one comparator's bitline nmos transistor (either the top or the bottom one which got switched on) has to drive all the other 63 transistors (or equivalent load capacitances). This issue has been avoided in the present architecture since exactly N bitlines are present.

The output code is preserved till the beginning of the next ADC conversion phase irrespective of resetting because resetting the comparators in the signal track phase will make the bitlines float and keep the voltage value unchanged. Note that the comparator next in the chain is triggered by the one before it after a delay as shown in Figure 7. This delay is to account for DAC settling and the speed of the combinational SAR logic core.

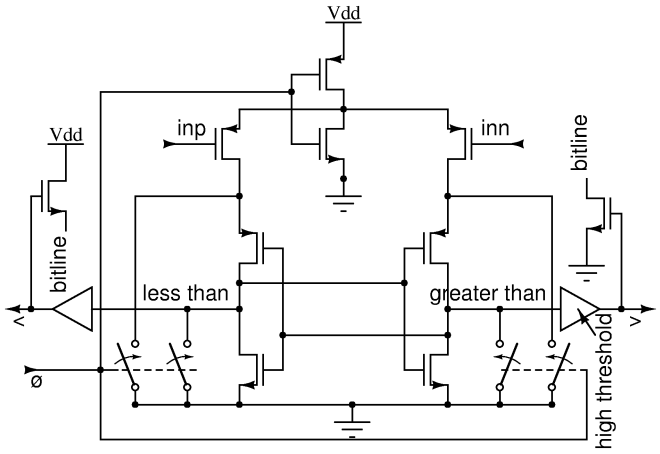


Figure 6. Circuit schematic of the comparator

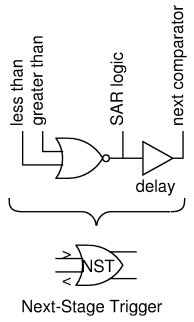


Figure 7. Next stage trigger

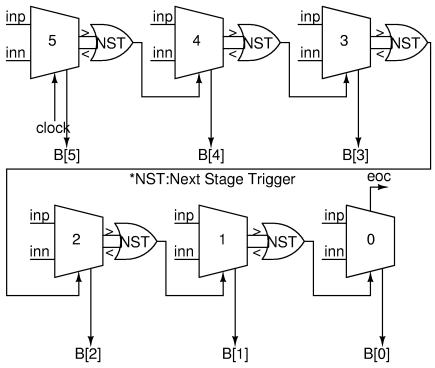


Figure 4. The N comparators connected as a chain

3.2. Binary C-DAC (Capacitor based Digital to Analog Converter)

Switched capacitor (or charge-redistribution) DAC ([15]) has been used in a straight forward manner here. The advantage w.r.t a resistive one is that the accuracy and linearity of the DAC (and in turn of the ADC) is achieved by default by the high-accuracy photolithography process (which controls the capacitor plate area and capacitance matching). For higher resolutions however, binary weighted capacitor array becomes too large for switching. One might have to go for modified DACs and switching sequences (for example, by using a multi-stage network [16] or a coupling capacitor). A dummy capacitor array is connected to the second input of the comparators to neutralize loading effects when the input is applied and removed on the bottom plates of the capacitor array connected to the first input. Input capacitance depends on the unit capacitance one uses, multiples of which will constitute the capacitive DAC. Since this portion has been implemented in a conventional manner with little optimization in an analog 0.6μm technology, the total input capacitance turns out to be high (see Section 4 and Table 2).

3.3. Digital Core without Latches

Significant power gains have been achieved by removing redundant latches/registers in the SAR logic core. The only latches present are those in the dynamic comparators which directly set the bitlines. Loading effects on the C-DAC switch driving lines differ and have been taken into consideration. For the 6 bit design, 40 basic gates constitute the SAR logic as listed in Table 1.

Gate	Count
And	18
Inverter	15
Or	5
Nand	2
Total	40

Table 1. Gates used in SAR Logic

4. Results

The 6 bit design was simulated in a 0.6μm $5V$ analog CMOS process with a full scale voltage of $2.5V$. Power consumption and other features are listed in Table 2 along with those reported in the literature. The important difference is that the other two implementations are in digital $\sim 1V$ CMOS thus giving better performances (e.g., power). Nonetheless, the performance results of the design using the 0.6μm 5V analog CMOS process (this technology is preferred for high resolution SAR ADCs) suggest that significant gains can further be made by voltage and technology scaling. No specific effort has been made to reduce the input capacitance (which will require a change in the DAC structure). The offsets of the comparators due to size mismatch and loading mismatch limits the resolution of the present scheme, among other things. Note that the difference in the offsets of different comparators is avoided in [1], and at the same time, is purposefully introduced and indispensable to the working of the state-of-the-art Asynchronous ADC in [2]. SFDR (Spurious-Free Dynamic Range) and PSD plots at simulation stage are not very relevant as post fabrication metrics can significantly differ and hence have not been shown here. [1] presents a time interleaved (TI) ADC housing two single-comparator sub-units and hence its performance metrics are slightly different (refer Table 2). Time interleaving has not been done here since it is a standard concept and requires considering analog or digital calibration to counter offset mismatch, gain mismatch and phase skew across channels.

Architecture (comparators)	Single ([1])	$2^N - 1$ ([2])	N (proposed)
Resolution	6	7	6
Process (CMOS)	130nm 1.2V digital	90nm 1V digital	**600nm 5V** analog
Input Cap.(pF)	0.09	0.25	3.2
Conv. Rate (MS/s)	600 (TI)	150	20
Peak SNDR (dB)	34	40	35.2
Efficiency (pJ/step)	0.22	0.01	5.985
Power(mW)			
Analog	1.2	0.089	0.75
Digital	4.1	0.044	4.84

Table 2. Comparison of the three asynchronous SAR architectures. Note that [1] is Time Interleaved.

5. Conclusion

Design of an asynchronous ADC which takes advantage of faster comparison cycles to perform the conversion has been detailed. With clock requirement equal to the sampling rate (unlike synchronous SARs), its area and power consumption scale linearly with resolution. Comparison with two other similar architectures has been done and a 6bit version has been simulated which works at 20MS/s. The limitation on the the clocking in synchronous case was governed by the slowest resolving time. Here, this bottleneck has been done away with. This ADC is faster than synchronous SAR ADCs and can be increasingly relevant in non-uniform sampling systems in areas like communication and imaging (e.g. MRI). When compared with a 4 bit flash for example, such an ADC is about 2.8 times slower instead of being 4 times slower like a synchronous SAR (see Section 2.2).

References

[1] S.-W. M. Chen and R. W. Brodersen. A 6-bit 600-ms/s 5.3-mw asynchronous adc in 0.13-μm cmos. *IEEE Journal of Solid-State Circuits*, 41(12):2669–2680, December 2006.

[2] G. V. der Plas and B. Verbruggen. A 150ms/s 133μw 7b adc in 90nm digital cmos using a comparator-based asynchronous binary-search sub-adc. *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 242–243, February 2008.

[3] B. P. Ginsburg and A. P. Chandrakasan. Highly interleaved 5b 250ms/s adc with redundant channels in 65nm cmos. *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 240–610, February 2008.

[4] V. Giannini, P. Nuzzo, V. Chironi, A. Baschirotto, G. V. der Plas, and J. Craninckx. An 82opw 9b 4oms/s noise-tolerant dynamic-sar adc in 90nm digital cmos. *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 238–239, February 2008.

[5] G. V. der Plas, S. Decoutere, and S. Donnay. A 0.16pj/conversion-step 2.5mw 1.25gs/s 4b adc in a 90nm digital cmos process. *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, pages 2310–2319, February 2006.

[6] B. Verbruggen, J. Craninckx, M. Kuijk, P. Wambacq, and G. V. der Plas. A 2.2mw 5b 1.75gs/s folding flash adc in 90nm digital cmos. *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 252–253, February 2008.

[7] E. Allier, G. Sicard, L. Fesquet, and M. Renaudin. A new class of asynchronous a/d converters based on time quantization. *Ninth International Symposium on Asynchronous Circuits and Systems*, pages 196–205, May 2003.

[8] Y. W. Li, K. L. Shepard, and Y. P. Tsividis. A continuous-time programmable digital fir filter. *IEEE Journal of Solid-State Circuits*, 41(11):2512–2520, November 2006.

[9] B. Schell and Y. Tsividis. A clockless adc/dsp/dac system with activity-dependent power dissipation and no aliasing. *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 550–551, February 2008.

[10] A. Nikoozadeh and B. Murmann. An analysis of latch comparator offset due to load capacitor mismatch. *IEEE Transactions on Circuits and Systems II*, 53(12):1398–1402, 2006.

[11] T. Kobayashi, K. Nogami, T. Shirotori, and Y. Fujimoto. A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architecture. *IEEE Journal of Solid-State Circuits*, 28(4):523–527, April 1993.

[12] B. Wicht, T. Nirschl, and D. Schmitt-Landsiedel. Yield and speed optimization of a latch-type voltage sense amplifier. *IEEE Journal of Solid-State Circuits*, 39(7):1148–1158, July 2004.

[13] U. Fiedler and D. Seitzer. A high-speed 8 bit a/d converter based on a gray-code multiple folding circuit. *IEEE Journal of Solid-State Circuits*, SC-14(3):547–552, June 1979.

[14] J. C. Bob Verbruggen, M. Kuijk, P. Wambacq, and G. V. der Plas. A 2.2mw 5b 1.75gs/s folding flash adc in 90nm digital cmos. *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 252–253, February 2008.

[15] J. McCreary and P. Gray. All-mos charge redistribution analog-to-digital conversion techniques. i. *Solid-State Circuits, IEEE Journal of*, 10(6):371–379, Dec 1975.

[16] Y. Yee, L. Terman, and L. Heller. A two-stage weighted capacitor network for d/a-a/d conversion. *IEEE Journal of Solid-State Circuits*, 14(4):778–781, Aug 1979.

Design of a Low Power, Variable-Resolution Flash ADC

Sreehari Veeramachanen, A. Mahesh Kumar, Venkat Tummala* ,M.B. Srinivas
Centre for VLSI and Embedded System Technologies(CVEST),
International Institute of Information Technology (IIIT),
Gachibowli, Hyderabad, 500032, India.
** San Jose State University*
Email: srihari@research.iiit.ac.in, maheshkumar_a@research.iiit.ac.in, ven646@gmail.com srinivas@iiit.ac.in.

Abstract: **In this paper, a low power and variable resolution (adaptive) flash ADC is proposed. The ADC enables exponential power reduction while the reduction in resolution is linear. In the proposed design, unused parallel voltage comparators are switched to standby mode leading to consumption of only the leakage power. The ADC, capable of operating at 4-bit, 5-bit, and 6-bit precision, dissipates 6mW at 4-bit and 12mW at 6-bit, and operates at a sampling frequency of 1 to 2 GSPS. The ADC has been designed and simulated in standard 65nm CMOS technology using Cadence tools.**

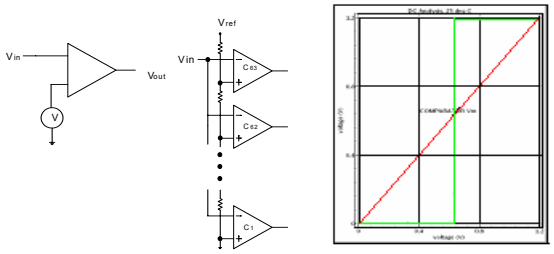
1. INTRODUCTION

Analog-to-digital converter (ADC) is a fundamental block in mixed-signal VLSI circuits. For high-speed applications, a flash ADC is often used. Resolution, speed, and power consumption are the three key parameters for an analog-to-digital converter (ADC). These parameters cannot be changed once an ADC is designed. While one can use 6-bit precision from an 8-bit ADC, it is non-optimal resulting in slower speed and extra power consumption due to full 8-bit internal operation.

In this paper, a new flash ADC design is proposed that is a true variable-power and variable-resolution ADC. It can operate at higher speed and will consume less power when operating at a lower resolution. Such features are highly desirable in many wireless and mobile applications. For example, the strength of a radio frequency (RF) signal varies greatly depending on geographic location. Optimally, the ADC resolution can be reduced upon the reception of strong signal and can be increased upon the reception of weak signal. Substantial reduction in power consumption at lower resolution will prolong the battery life.

2. Background

Low power ADC architectures are implemented with pipelined, successive approximation, and sigma-delta modulators. These are all useful for the medium speed conversion and high resolution applications. On the other hand, the flash architecture is suitable for high speed conversion and low resolution applications due to its parallel architecture. Figure 1 represents the conventional flash ADC with dc characteristics, which requires many analog comparators and the complexity and power dissipation become very high. Moreover, the accuracy of dividing resistors requires high value for reference voltage if the conversion resolution is high. Because many comparators compare the reference voltage with input voltage at the same time, power consumption in the flash architecture is much larger than for the others. Controlling the power consumption in the comparator is the key to reducing the overall power consumption in a flash ADC.



(a) (b) (c)

Figure 1: Differential voltage comparator

In this work, the authors propose a new, low-power and low-voltage flash ADC based on an existing approach [1-3,8] of TIQ-based ADC in which the ratio of channel length and width are designed by which the transition threshold of the CMOS inverters is varied to detect input analog signal. The values are then encoded into the digital code. The advantages are that the ADC circuit does not need any resistor and uses simple CMOS inverters rather than analog comparators.

3 Proposed ADC Design

The key feature of the comparator in the proposed design is the fact that the comparator can easily and quickly switch from active mode to standby mode. It consists of following four sub blocks followed by their integration.

3.1 Bias block
3.2 Peak detector
3.3 comparator
3.4 Decoder block

3.1 Bias block

Bias block consists of band-gap reference and voltage regulator generating reference voltages as shown in the figures 2. The band-gap reference block generates 0.7V and

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.62

voltage regulator generates reference voltages V1=1.1, V2=0.9 and V3=0.7 in this application.

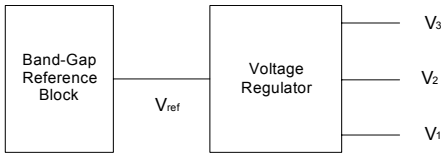


Figure. 2. Bias block

3.2 Peak detector

The function of the peak detector is to compute the peak value of the input. The circuit follows the voltage peaks of a signal and stores the highest value on a capacitor. If a higher peak value signal comes along, this new value is stored. The highest peak value is stored until the capacitor is discharged.

Consider the circuit of figure 3. When input V_{in} exceeds Vc, the voltage across capacitor, the diode D is forward biased and the circuit becomes a voltage follower. Consequently, the output voltage Vo follows V_{in} as long as V_{in} exceeds Vc. When V_{in} drops Vc, the diode becomes reverse-biased and the capacitor holds the charge till input voltage again attains a value greater than Vc. Figure 4 shows the different voltage wave shapes for the peak detector. The advantage of peak detector is self-triggering, self-sparsifying and timing output.

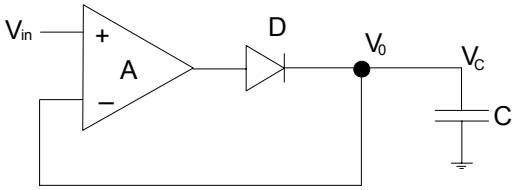


Figure. 3. Peak detector

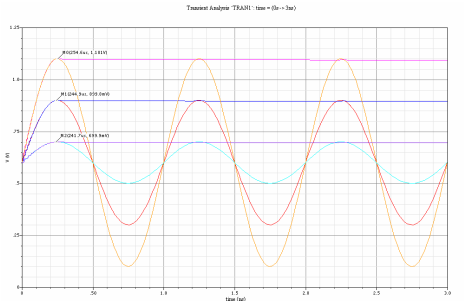


Figure. 4. Different Voltage wave shapes for the peak detector

3.3 Comparator

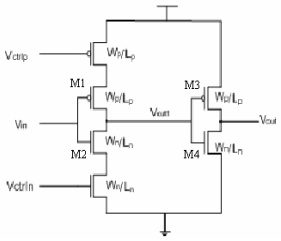
The comparator is the most important component in the ADC architecture. Its role is to convert an input voltage (Vin) into a logic `1' or `0' by comparing a reference voltage (Vref) with the Vin. If Vin is greater than Vref, the output of the comparator is `1', otherwise `0'.

The proposed comparator uses two cascading CMOS inverters as a comparator for high speed and low power consumption. This is a modification of Tangel [1-3] who used TIQ comparator for implementing a high speed flash ADC.

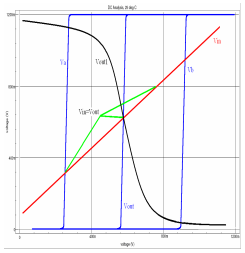
3.3.1 CMOS Inverter as a Comparator

The inverter threshold (Vm) is defined as the Vin = Vout in the VTC of an inverter. Mathematically,

$$V_m = \frac{r\left(V_{DD} - \left|V_{Tp}\right|\right) + V_{Tn}}{1+r} \quad \text{with} \quad r = \sqrt{\frac{k_p}{k_n}} \tag{3.1}$$



(a) Inverter schematics diagram



(b) Inverter VTC

Figure. 5. Inverter schematic and VTC

Where VTp and VTn represent the threshold voltages of PMOS and NMOS devices, respectively. Figure 5 shows the schematic of an inverter and its VTC from the simulation.

The first inverter consists of controllable inputs Vctrlp and Vctrln to operate inverter in stand-by mode or active mode as shown in figure 5(a). At the input the analog signal quantization level is set by Vm depending on the W/L ratios of PMOS and NMOS, the control voltages Vctrlp=0 and Vctrln=1 for active mode and Vctrlp=1 and Vctrln=0 for stand-by mode . The second inverter is used to increase voltage gain and to prevent an unbalanced propagation delay. In Figure 5(b), the slope of Vout is shown larger than the one of Vout1. The inverter threshold depends on the transistor sizes. The inverter VTC Va and Vb show the difference from the VTC of Vout. With a fixed length of the PMOS and NMOS devices, we can get desired values of Va and Vb by increasing only the width of the PMOS and NMOS transistors, respectively.

This result can be confirmed by the following equation of the inverter threshold

$$V_m = \frac{\sqrt{\frac{\mu_p W_p}{\mu_n W_n}}\left(V_{DD} - \left|V_{Tp}\right|\right) + V_{Tn}}{1 + \sqrt{\frac{\mu_p W_p}{\mu_n W_n}}} \tag{3.2}$$

where µp and µn are the electron and hole mobility, respectively. To derive above Equation, we assume that both

transistors are in the active region, the gate oxide thickness (Cox) for both transistors is the same, and the lengths of both transistors (Lp and Ln) are also the same. From Equation 3.2, we know that Vm is shifted depending the transistor width ratio (Wp/Wn). That is, increasing Wp makes Vm larger, and increasing Wn results in Vm being smaller on the VTC. This changing of the widths of the PMOS and NMOS devices with a fixed transistor length is the idea of the TIQ comparator[1-3]. We can use the inverter threshold voltage as an internal reference voltage to compare the input voltage. However, to use the CMOS inverter as a voltage comparator, we should check the sensitivity of Vm to other parameters, which are ignored in Equation 3.2, for correct operation of the proposed flash ADC.

3.4 Decoder block

A decoder for flash analog-to-digital converter with short critical path, regular structure, and small area has been suggested earlier [4]. In this work, it has been modified to make its operation efficient.

The decoder is based on 2:1 multiplexers connected as a tree. Each level of the tree divides the input thermometer scale in two and calculates one of the bits in the binary output. In comparison with the Wallace tree decoder and the folded decoder the length of the critical path is approximately reduced to one third and one half, respectively. The amount of hardware is also reduced, which will translate to a power saving, compared with the Wallace tree decoder and the folded decoder.

The multiplexer-based encoder can be found by observing that the most significant bit (MSB) of the encoder output is equal to the middle digit in the thermometer code. The second highest bit is found in the same way, but only considering a part of the whole thermometer code. The lower half if the middle digit is zero, otherwise the upper half. This is continued until all output bits are found. The algorithm is illustrated by the figure 6 below to the left and can be realized by the circuit below to the right, which entirely consists of 2-to-1 multiplexers. Hence, it has a regular structure, which is an advantage during layout [5]. A comparison between different types of encoders as shown in table 1, also indicate that this encoder has the lowest hardware cost among all and shortest critical path. Hence, it is probable that it also is the fastest encoder. In the following figure 6, block diagram is shown with an example.

Type of encoder	N.o. MUX's	Critical path
Wallace tree	171 MUX	18 t_{mux}
4-level folded	81 MUX	12 t_{mux}
Multiplexer based	57 MUX	5 t_{mux}

Table 1 Different types of encoders

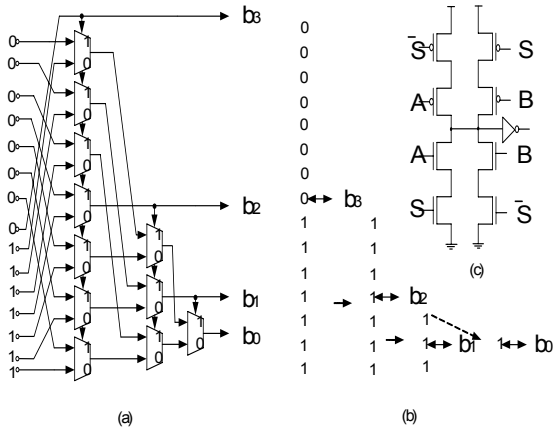


Figure 6 Decoder block diagram with example

Figure 6 (c) represents the CMOS based 2-to-1 multiplexer. The advantage of this multiplexer are

1. Acts as gain booster in the ADC.
2. Low leakage operation
3. Two legs are symmetrical
4. Layout area of the MUX is less and symmetry
5. Power dissipation is less

3.5 Integration of Blocks

The proposed flash-ADC features modified threshold inverter quantization (TIQ) technique [1-3] for low power and variable resolution using standard CMOS 65nm technology. Figure 7(a) and 7(b) shows Peak Detector for Reconfigurability and block diagram of the proposed ADC . The proposed ADC consists of bias block generating reference voltage V1, V2, V3 (V1>V2>V3) which are used as reference levels to determine the voltage levels of the analog input signal, peak detector for determining the voltage levels, 63 inverters, decoder s (4-bit, 5-bit and 6-bit), digital block and output multiplexer.

The analog input signal Vin is given to the peak detector and the inverters. Peak detector consists of an amplifier, three diodes D1, D2, D3, comparators and multiplexers. When the input signal Vin applied to the peak detector the comparator output comp1 is high and mux1 is short circuited. when Vin exceeds V_{k1} then diode D1 is forward biased and the circuit becomes a voltage follower. Consequently, the output V_{k1} follows Vin as along as Vin exceeds V_{k1}. When Vin drops below Vc, the diode becomes reverse biased and the capacitor holds the charge till input voltage again attains a value greater than Vc. when diode D1 is reverse biased and the voltage V_{k1}=Vc is greater than V1 then comparator Comp1 triggers and output is '0'. In this case mux1 open circuited and the peak voltage is stored on to the capacitor (Vc). The control signals C1=0 and C2=C3=1 and hence peak detector acts as V1 voltage peak detector.

The control voltages C1, C2 and C3 are given to a digital block to generate control voltages (A1, A2 and A3) for decoders as shown in table 2.

INPUTS			OUTPUTS		
C1	C2	C3	A1	A2	A3
0	1	1	0	1	1
0	0	1	1	0	1
0	0	0	1	1	0

Table 2. Control Signals for inverters, Decoders and Multiplexer

6-bit inverters are designed with different threshold voltages i.e., for highest input peak voltage signal the control signal values are C1=0, C2=C3=1 which will turn on (active mode) LSB inverters from 0 to 15 and turn off (stand-by mode) from 16 to 63. In the decoder section we have 4-bit, 5-bit and 6-bit decoders separately. The control signal A1=0, A2=A3=1 which will select 4-bit decoder and output multiplexer such that proposed ADC operates as 4-bit ADC with highest input analog signal.

For the second highest peak input analog signal, the signal is compared with reference voltage V2. During this comparison if V_{k2}=Vc is greater than V2 then comparators Comp1, Comp2 triggers and outputs of comparators go to '0' therefore C1=C2=0, C3=1 and digital block outputs are A1=A3=1, A2=0. Hence peak detector acts as V2 voltage peak detector. The control signals C1, C2, C3 will turn on (active mode) inverters from 0 to 31 and turn off (stand-by mode) from 32 to 63. The control signals A1, A2 and A3 will select 5-bit decoder and output multiplexer such that proposed ADC operates as 5-bit ADC.

For the Lowest peak input analog signal, the signal is compared with reference voltage V3. During this comparison, if V_{k3}=Vc is greater than V3 then comparators Comp1, Comp2, Comp3 triggers and outputs of comparators go to '0' therefore C1=C2=C3=0 and digital block outputs are A1=A2=1, A3=0. Hence peak detector acts as V3 voltage peak detector. The control signals C1, C2, C3 will turn on (active mode) inverters from 0 to 63. The control signals A1, A2 and A3 will select 6-bit decoder and output multiplexer such that proposed ADC operates as 6-bit ADC.

The programmable resolution based upon analog input peak voltage and its corresponding power consumption values and performance parameters are given in table 3.

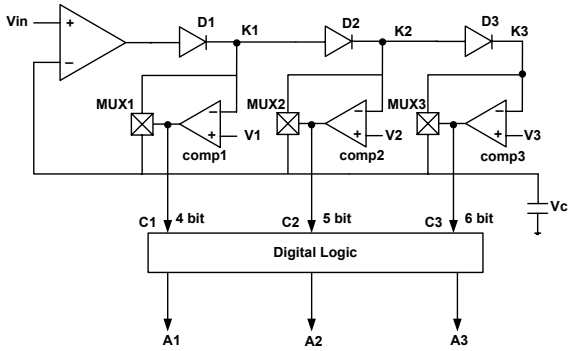


Figure 7(b) Peak Detector for Reconfigurability

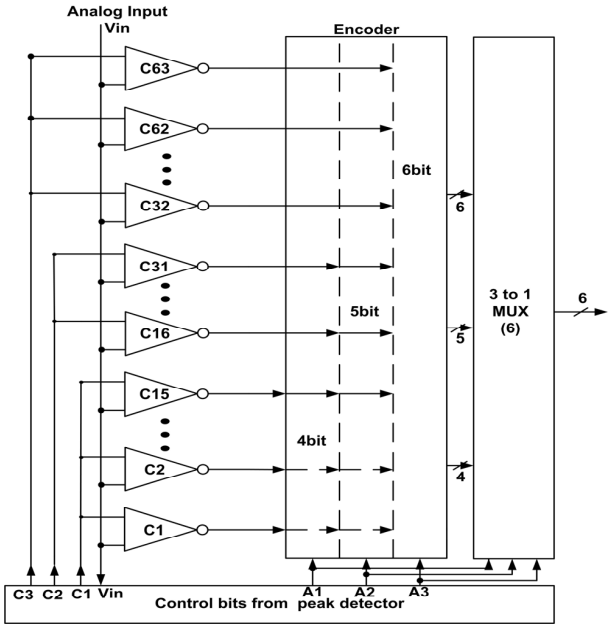


Figure 7(b) Block diagram of Proposed ADC

4 Simulation Results

This section describes the functional simulation of the proposed ADC and characterization to verify that is suitable for high speed, low voltage application.

4.1 Functional simulation

A transient and DC analysis of the ADC is done by giving a ramp input going from 0.264V to 0.888V (which is full scale range of the ADC) and with each LSB voltage level (VLSB) of 10mv. The digital codes were obtained correctly going from 0 to 63 at the output with one VLSB of 10mv, indicating that the ADC was functionally correct. The simulation results are shown in figure 8(a) and figure 8(b).

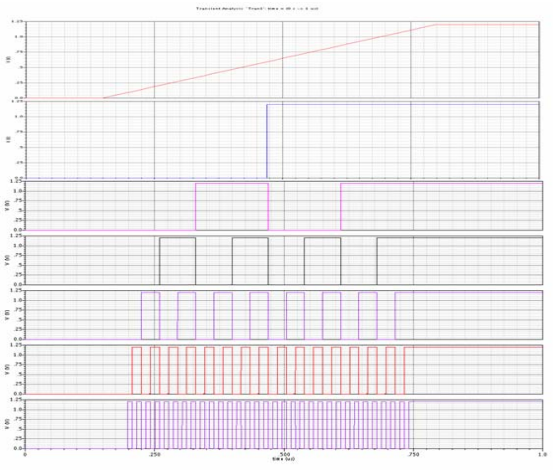


Figure 8(a) Transient analysis of the 6-bit ADC to prove functional correctness

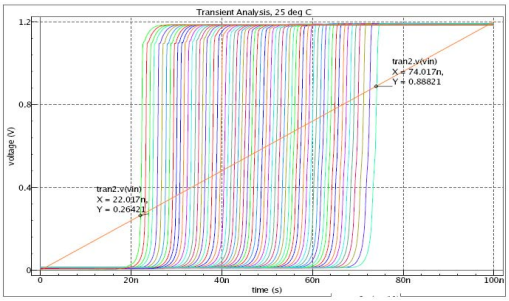


Figure 8(b) Inverter threshold levels

4.2 Characterization

The proposed ADC (4-bit, 5-bit and 6-bit) is designed in 65nm technology, characterized for parameters like differential non-linearity (DNL), integral non-linearity (INL) (Static performances), signal-to-noise ratio (SNR) , signal to noise and distortion ratio (SNDR) and effective number of bits (ENOB) (Dynamic performances) [6-8] as shown in table 3.

	4-bit	5-bit	6-bit
DNL (LSB)	0.3	0.36	0.4
INL (LSB)	0.28	0.32	0.35
Input frequency	1.2Gs/sec	1Gs/sec	800Ms/sec
SNR	30dB	30dB	34.5dB
SNDR	29.5dB	29.5dB	34dB
ENOB	4.6	4.6	5.34
Avg.Power(mw)	6mw	9mw	12mw
Layout Area (um2)	320x320	400x400	500x500
Power supply (v)	1.2	1.2	1.2

Table 3 Proposed ADC features for 4-bit, 5-bit and 6-bit.

The proposed ADC designed for 6-bit is compared with conventional 6-bit comparator based ADC for static performance, dynamic performance and power consumption as shown in table 4.

	Comparator based ADC Design	**Proposed ADC Design**
DNL (LSB)	0.8	0.4
INL (LSB)	0.5	0.35
Input frequency	800Ms/sec	800Ms/sec
SNR	32.3dB	34.5dB
SNDR	31.86dB	34dB
ENOB	5.0	5.34
Avg.Power(mw)	70mw	12mw
Layout Area(um2)	900x900	500x500
Power supply	1.2V	1.2

Table 4 Comparison of 6-bit comparator based and Proposed ADC design

DNL and INL testing is done by including verilog-A block which generates a slowly varying full scale range ramp is given as input to the proposed flash ADC, which completes the full scale range in 63 steps for transistor level implementation. The values of the each code are compared with ideal value and store the difference value. The results show that the ADC exhibits a maximum DNL of 0.4LSB. and INL of 0.35LSB as shown in the Fig 9 and Fig 10 respectively.

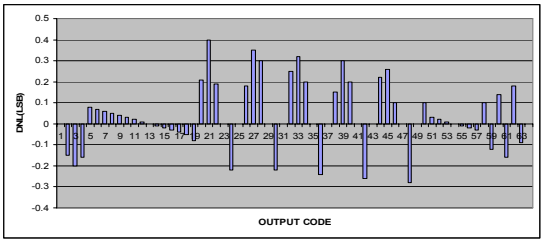


Figure 9: DNL plot of the 6-bit ADC

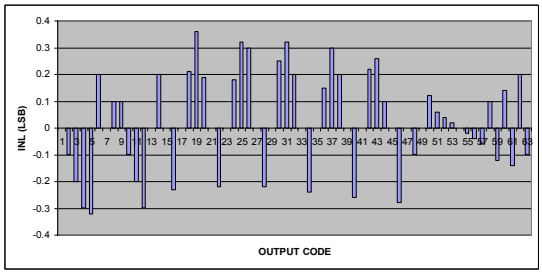


Figure 10 INL plot of the flash ADC

The SNR and SNDR of the designed ADC have been measured at an input frequency of 400MHz. The flash ADC is fed a sinusoidal input which covers the entire full scale range, and the output is fed to an ideal DAC is a reconstructed, digitized sine wave, at 400MHz. The FFT of this sine wave is plotted, from which SNR and SNDR values at different input frequency range is shown in the figure 11. The SNR and SNDR was found to be 34.5 dB and 34.0 dB.

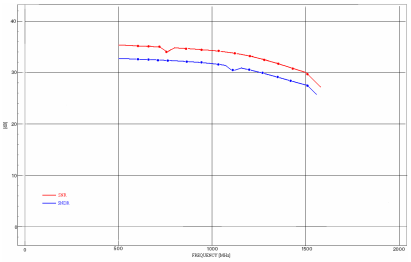


Figure 11 SNR and SNDR plot

The effective number of the bits (ENOB) shows an ADC's performance at a specific input frequency. The ENOB is really related to the input frequency. If the input frequency is increased, then the ENOB degrades. The ENOB can be calculated with SNDR as shown in equation.

$$ENOB = (SNRD - 1.76) / 6.02$$

ENOB for five different frequencies is shown in the table 4.

Table 4 Different ENOB values with respect to frequency.

I/P Frequency	No of bits	ENOB
100MHz	6-bit	5.6
200MHz	6-bit	5.5
300MHz	6-bit	5.4
400MHz	6-bit	5.34
800MHz	6-bit	5.2

4.6 Power simulation

Figure 12 shows the comparison of power consumption for conventional 6-bit flash ADC and proposed 6-bit ADC design. We observe that peak powers are 70mW and 12mW for conventional ADC and proposed ADC respectively.

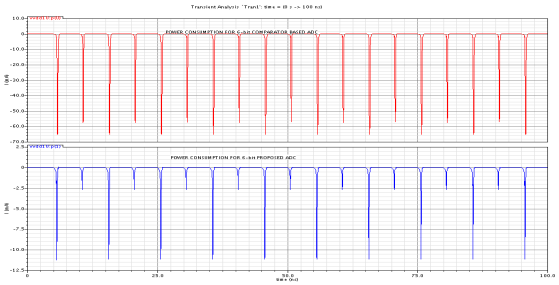


Figure 12 Instantaneous power plot of the flash ADC

The flash ADC is fed a sinusoidal input operating at a frequency of 800MSPS which covers the entire full scale range, and the output is fed to an ideal DAC is a reconstructed, digitized sine wave, at 800Ms/sec is shown in the figure 13 .

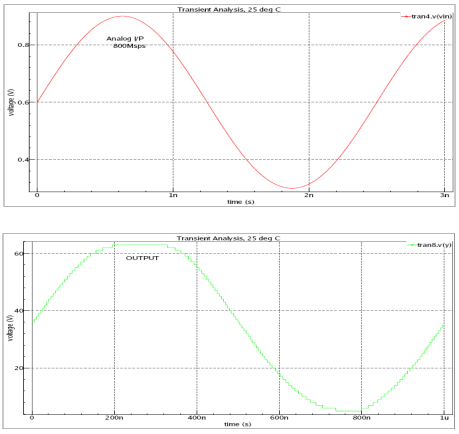


Figure 13 800Msamples/sec 6-bit ADC Input and output waveform.

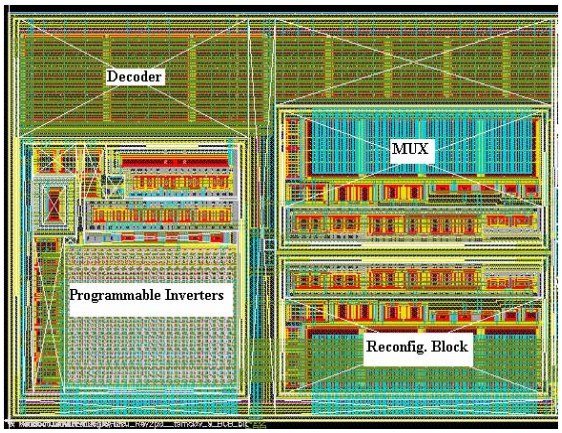


Figure 14 Layout of Proposed ADC

Figure 14 represents the layout view of the Proposed ADC. ADC footprint is made such that all input pins are brought on to the left side and output pins are on the right side of the layout.

Conclusion

The proposed low power and variable resolution flash ADC design operates at high speed with programmable resolution based upon analog input peak voltage. It operates at higher speed, lower resolution and consumes less power. The advantage of proposed ADC is built in peak detector which will detect the peak level of the analog input signal and provides programmable feature (i.e., at highest input voltage peak level ADC will operate as 4bit and for lowest input voltage peak level ADC will operate as 6-bit).

References

[1] A. Tangel, "VLSI implementation of the threshold inverter quantization (TIQ) technique for CMOS flash A/D converter applications." Ph.D. Dissertation, The Pennsylvania State University, Aug. 1999.

[2] J. Yoo, "A TIQ Based CMOS Flash A/D Converter for System-on-Chip Applications", Ph.D Thesis, The Pennsylvania State University, May 2003.

[3] Tangel, A.; Choi, K, "'The CMOS Inverter as a Comparator in ADC Designs", spinger Analog Integrated Circuits and Signal Processing, Vol.39, pp.147-155,2004.

[4] E. Säll, "Implementation of Flash Analog-to-Digital Converters in Silicon-on-Insulator Technology," Linköping Studies in Science and Technology, Thesis No. 1213, ISBN 91-85457-79-5, Linköping, Sweden, Dec. 21, 2005.

[5] J. M. Rabaey, A. Chandrakasan, and B. Nikolic′, "Digital Integrated Circuits", 2nd Edition, 2003.

[6] Maxim Integrated Products, INL/DNL Measurements for High-Speed Analog to-Digital Converters (ADCs).

[7] Maxim Integrated Products. Defining and Testing Dynamic Parameters in High-Speed ADCs, 2001

[8] Rudy J. van de Plassche, "CMOS Integrated Analog-to-Digital and Digital-to-Analog Converters", 2nd Edition,2005.

Session 2B

Floorplanning and Analog Layout

2009 22nd International Conference on VLSI Design

Floorplanning for Partial Reconfiguration in FPGAs

Pritha Banerjee
Indian Statistical Institute
ACMU, Kolkata, India
pritha_r@isical.ac.in

Megha Sangtani
Nvidia Graphics Pvt. Ltd.
Pune, India
meghasangtani@gmail.com

Susmita Sur-Kolay
Indian Statistical Institute
ACMU, Kolkata, India
ssk@isical.ac.in

Abstract

Partial Reconfiguration on heterogeneous Field Programmable Gate Arrays (FPGA) with millions of gates yields better utilization of resources by swapping in and out the active modules of one or more applications at an instant of time. Given a schedule of sub-task instances with each instance having a netlist of active modules, a global floorplanning method is essential to reduce the reconfiguration overhead by fixing the position and shapes of common modules across all instances, while optimizing the performance. Here we propose a global floorplan generation method to obtain same positions for the common modules across all instances such that the heterogeneous resource requirements of all modules in each instance are satisfied, and the total wirelength (HPWL) over all instances is minimal. We also provide experimental results in support.

1. Introduction

Modern FPGA architectures like Xilinx' Virtex series allow partial dynamic reconfiguration, i.e., inactive parts of a design implemented on FPGA chip can be replaced by other designs while the remaining part of FPGA continues to execute. Thus, partial reconfiguration helps executing a large application in the same piece of hardware by swapping in and out the active and inactive parts of design when the whole application does not fit completely on the chip. This incurs an additional partial reconfiguration overhead each time a new part is swapped in and out of the FPGA chip. Hence an appropriate scheduling of task/application/design is necessary to reduce the partial reconfiguration overhead such that common tasks/designs need not be swapped in and out again and again. Given a schedule of instances consisting of a set of common as well as other tasks, the resources on the chip may get fragmented due to arbitrary placement of tasks on the chip. The tasks of the consecutive instances might not fit contiguously in the fragmented resources scattered across the chip. This might lead to reconfiguration of the whole chip to make contiguous space for each task incurring reconfiguration overhead defeating the whole purpose of partial reconfigurability. As modern FPGAs are heterogeneous in nature with preplaced blocks like RAM, Multipliers along with array of CLBs, the mapping of tasks for each instance allocating heterogeneous resources contiguously to each task while meeting the performance objective, becomes more complex. In this paper, we propose a fast performance aware global floorplan generation method for the tasks/modules of each instance of a given schedule such that the common tasks/modules across instances take same position and shape on the target FPGA chip resulting in minimal reconfiguration overhead and the total semi-perimeter wirelength over all instances is optimized by placing the other modules considering their connectivity and position of common modules.

In the rest of the paper, Section 2 reports related earlier works. While Section 3 has the problem formulation, the steps of the proposed method are detailed in Section 4, 5 and 6 respectively. Section 7 reports the experimental results with concluding remarks in Section 8.

2. Previous works

In the FPGA literature, floorplanning a set of modules in a single instance itself is only a handful. Cheng and Wong [4], Feng and Mehta [5] have proposed simulated annealing based methods whereas Banerjee et. al. [2] have proposed deterministic topology generation and node sizing for floorplanning the heterogeneous FPGAs. The earliest work on floorplanning for partial reconfiguration was formulated as a 3D template placement problem in [3]. Singhal and Bozorgzadeh [12] have introduced a new multi layer sequence pair representation based floorplanner which maximizes the overlap of common components of multiple designs thereby reducing reconfiguration overhead. The commercial tool like Xilinx' Planahead [8] requires manual placement of the common modules beforehand. Ahmadinia et. al [1] have proposed an algorithm for on-line optimal free space management and routing conscious dynamic placement for reconfigurable devices. Our fast deterministic global topology generation is based on slicing tree and its sizing unlike [12] which minimizes reconfiguration overhead by placing

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.36

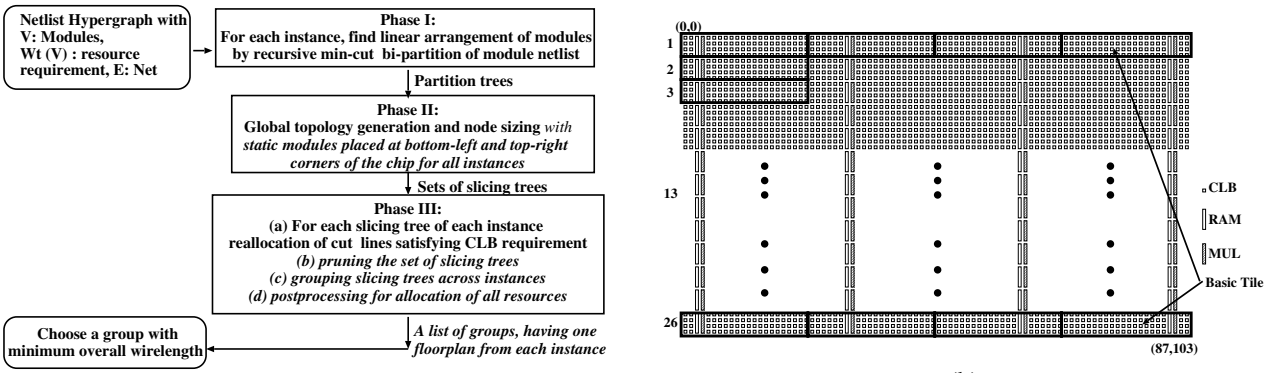


Figure 1. (a) Flow of the proposed method; (b) Spartan-3 XC3S5000 FPGA Architecture, tessellated with a basic tile.

the common modules of same shape across all instances at specific position on the chip.

3. Floorplanning for partial reconfiguration

Definition 1 *Resource Requirement Vector [4]: A 3-tuple vector R_m = (m_{clb}, m_{ram}, m_{mul}) represents the number of CLBs, RAMs and MULs required by a module m.*

Definition 2 *Static and Dynamic modules: Given a schedule of instances, modules which are common and remains active in all instances are called static modules. The rest of the modules which are swapped in and out of an instance, are called dynamic modules.*

Floorplanning problem for partial reconfiguration is essentially the generation of a global floorplan where each floorplan corresponding to each instance of a schedule is a feasible floorplan and the common modules are placed at the same location with same shape in each of the instances, while the total HPWL across all instances is minimal. We build upon the single instance floorplanning problem for heterogeneous FPGAs of [2] as follows. In a given schedule, let there be k instances $I_1, I_2, \cdots I_k$. Let $s_1, s_2, \cdots s_m \in SM$, be m static modules that remain active in all instances. For each $I_i, 1 \leq i \leq k$, let there be n_i modules $d_{i1}, d_{i2}, \cdots d_{in_i}$. The connectivity of the modules C_i, in each instance i is also given. The objective is to find floorplans for all instances, such that (i) the resource requirement of a module R_{m_i} is satisfied within a region $(x_{min}, y_{min}, x_{max}, y_{max})$ in each instance without overlap, (ii) the location and shape (i.e., width and height) of each static module is same across all instances, (iii) the half-perimeter wirelength (HPWL) of netlist for all instances is minimized.

Our method consists of three phases as shown in Fig. 1. In the first phase a linear arrangement of modules for each instance is obtained such that the static modules are fixed to the same position across instances. In the second phase, a list of global slicing topologies is generated for each instance such that the positions of static modules are fixed at diagonally opposite corners of all floorplans leaving the maximal contiguous space for dynamic modules. Finally, on the basis of similarity between slicing trees of different instances, a set of groups, each having a set of slicing trees is generated. For each slicing tree in each group, a rectangular region is assigned to every module, which respects the cut direction and the actual resource requirement of the modules. The group with least total wirelength is the final solution.

For soft modules with homogeneous resource requirement such as only CLBs, the requirement can be factorized to generate a set of possible shapes (i.e, width and height), which can be later used for node sizing in traditional topology generation when floorplans are represented as slicing trees [11]. For heterogeneous resource requirements, where each resource type have specific location on the board, shapes can not be generated from the resource requirement vector R_m. Thus, *basic tile* [2], an uniform entity is defined to compute the resource requirement of each module, which could be easily adapted for generation of shapes during node sizing. Thus the chip is composed of $T_w \times T_h$ basic tiles arranged in h rows and w columns. In Fig. 1(b), The basic tile $A = (80, 1, 1)$ consists of 20×4 CLBs, 1 RAM and 1 MUL and $T_w \times T_h$ is 4×26.

4. Linear arrangement of Modules

In order to minimize the wirelength [2], we obtain a linear arrangement of modules, taking the left to right order of the leaves of a partition tree obtained by recursive partitioning of module netlist. The partition tree is the baseline of slicing tree generation in the next phase. For every

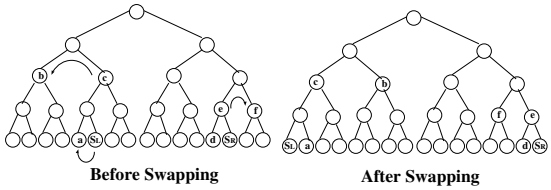


Figure 2. Swapping of static super modules to extreme ends of the partition tree; the arrow shows the partitions to be exchanged

instance of the given schedule, we use a balanced mincut bi-partitioning tool hMetis [6] to partition the modules of a netlist (represented as *hypergraphs*) by extending the partitioning of [2]. The weight associated with a vertex is the number of basic tiles required to satisfy the resource requirement of the corresponding module.

As static modules must have the same shape and location across all instances, it is beneficial to place all the static modules at two diagonally opposite corners of the floorplan. This provides the maximal contiguous space to place the rest of the dynamic modules.

Observation 1 *In a slicing tree representation of a floorplan, the modules at left most and right most leaves in a slicing tree always correspond to the two diagonally opposite corners of the floorplan.*

From Observation 1, if the static modules are placed at the left most and right most ends of the partition tree, the modules will definitely be on the opposite corners on the floorplan. If the netlist of modules of each instance is considered separately for linear arrangement, the static modules might go anywhere in the linear arrangement of each instance. We formulate a constrained linear arrangement of modules of each instance such that, in every instance, the positions of static modules are same in the partition tree and thus in linear arrangement.

First we extract the static modules and the corresponding netlist from the given schedule. Then we bi-partition the static modules into two partitions S_L and S_R and call each of them a *super module*. For each instance, the two super modules along with dynamic modules and their netlist is bi-partitioned recursively based on balanced min-cut until each partition contains at most one module/super module per partition. In the first level of recursive bi-partitioning, we force S_L and S_R to be in different partitions for every instance, so that they can be pushed to extreme left and right positions respectively during further recursive partitioning. Since swapping of partitions in a partition tree does not affect the min-cut in the tree, during each recursive bi-partition the left and right partitions are swapped such that partitions S_L and S_R are always pushed to the extreme left and extreme right of the partition tree. The swapping of partitions with static super modules is shown in Fig. 2. Thus we get one partition tree for each instance of the given schedule where the static super modules are at the extreme left and right leaves of each partition tree.

5. Global floorplan topology generation

In this step a set of sliceable floorplan topologies is generated for each instance simultaneously by appropriate horizontal and vertical node sizing starting from a set of possible shapes (in terms of tiles) of each module.

A list of *irredundant shapes* are generated [2] for every module and the super modules m_i, of all instances by factorizing T_{m_i}. Thus, each leaf node of the partition tree corresponding to a module contains a list of possible shapes, i.e., (width, height) pair in terms of tiles. For all instances, the corresponding partition trees are traversed simultaneously bottom-up, level by level, generating a set of irredundant sub-floorplans by combining the available shapes of its left and right children with vertical or horizontal cut. To generate irredundant shapes at the parent node of static super modules, a particular shape of static super modules may be thrown out in some of the instances when combined with its neighboring dynamic modules by a cut. We discard such shapes of static super modules from all the instances when a particular shape is eliminated from any of the instances to maintain same list of shapes of static super modules in all the instances. If at any level of simultaneous processing of slicing tree generation for all instances, we end up with an empty list of shapes for any of the static super modules, then we can directly report that no floorplan is possible for the set of instances on the given target board for the linear arrangement of modules/super modules obtained in the first phase. We might have to iterate the process with a different linear arrangement of modules. At the end of this phase a set of slicing trees for each instances is generated with static super modules at two opposite corners of the floorplan. The final shape (width, height) of the floorplan may not fit the target FPGA chip when the shapes are considered in terms of tiles. All shapes that are either wider or longer than the target chip may not result in a feasible floorplan satisfying all its resource requirements[2]. Thus we select only those slicing trees with final shape of width between 3 and 6 tiles for a target chip of 4×26 tiles, as there is a high possibility of obtaining a feasible floorplan on this chip in third phase.

6. Realization of Slicing Trees on the chip

For the selected slicing trees of every instance, coordinate positions are assigned to each module and static super modules respecting the cut lines and satisfying the exact CLB and RAM/MUL requirements. Thus a set of feasible floorplans for each instance is generated.

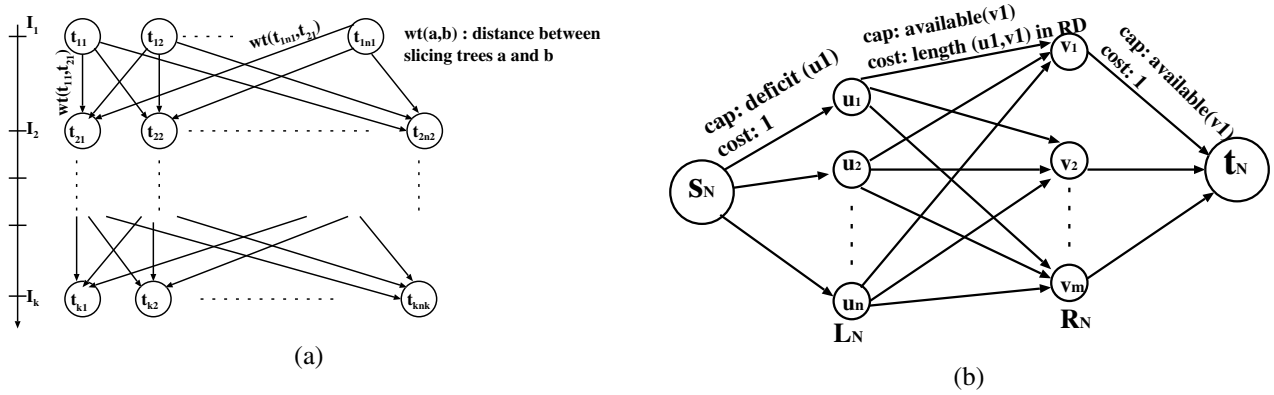


Figure 3. (a)Grouping of slicing trees by finding shortest pathin the associated digraph; (b) postprocessing by min-cost network flow graph for satisfying CLB requirement

Allocation of rectangular region to a module: Each selected slicing tree of every instance is traversed top-down and a rectangular region is assigned to every node using the cut direction at its parent and the number of CLBs required at that node. The root node of the slicing tree corresponds to the target board with 80 CLB columns and 104 CLB rows in it. Let the region $({x_{min}}^p, {y_{min}}^p, {x_{max}}^p, {y_{max}}^p)$ allocated to a parent node p contains r_{clb} rows and c_{clb} columns of CLBs. Let the CLB requirements at node p, its left child l and its right child r be p_{clb},l_{clb} and r_{clb} respectively. Let the number of CLB columns (rows) allocated to a node p is p_{col} (p_{row}), then l_{col} (l_{row}) and the rectangular region $({x_{min}}^l, {y_{min}}^l, {x_{max}}^l, {y_{max}})$assigned to the left child l of parent p is computed as follows. For a horizontal cut at p, $l_{col} = l_{clb}/p_{row}$; ${x_{min}}^l = {x_{min}}^p$; ${y_{min}}^l = {y_{min}}^p$; ${x_{max}}^l = {x_{min}}^p + l_{col} - 1$; ${y_{max}}^l = {y_{max}}^p$. For a vertical cut at p, $l_{row} = l_{clb}/p_{col}$; ${x_{min}}^l = {x_{min}}^p$; ${y_{min}}^l = {y_{min}}^p$; ${x_{max}}^l = {x_{max}}^p$; ${y_{max}}^l = {y_{min}}^p + l_{row} - 1$. For right child r, $r_{col} = r_{clb}/p_{row}$ for horizontal cut and $r_{row} = r_{clb}/p_{col}$ for vertical cut. The coordinate position for the right child r is computed similar to the left child l. As a convention, the vertical cut line is positioned by counting the CLB columns from left to right for the left child and right to left for the right child. Similarly for horizontal cut, it is positioned by counting the rows from bottom to top for the left child and top to bottom for the right child. Positioning of cut lines in this fashion generates two types of regions; (i) two non overlapping regions corresponding to two modules at opposite sides within the rectangle assigned to the parent node (ii) an overlapping or free region at the middle of the parent region. The overlapped region is generated when a column or row has to be shared by both modules and a free rectangular region is generated when resource requirement of both modules are much lesser than the avilable resources in the parent region. We allocate the CLBs required by a module to the non overlapping region of the rectangles assigned to the corresponding module. The remaining CLB requirement of each module, called *deficit*, has to be satisfied either within the overlapping rectangle or in the neighboring rectangles assign to other modules. The deficit of any module is satisfied during post processing described later.

At the end of this step, a set of floorplans are generated corresponding to each selected slicing tree of each instance by allocation of rectangular regions to each module / super module satisfying the CLB requirements either completely or partially. The process of allocation generates three types of rectangular region; (i) non overlapping part of parent rectangle assigned to a module either with no free CLBs within it or with some free CLBs in it (ii) overlapping rectangle in the middle, where conflicts for CLB requirements of more than one module needs to be resolved, (iii) free rectangular region in the middle due to the convention followed to assign rectangles and lesser resource requirement of a module than the available.

Pruning the set of slicing trees: While allocating the rectangular regions to modules in different instances, their RAM/MUL requirements are not considered. To check whether the RAM/MUL requirement of each module and super modules are satisfied within the rectangular region allocated, we define the following.

Definition 3 *Major Violation : If a module has RAM/MUL requirement and has been assigned the rectangular region such that no RAM/MUL column passes through it, then the module is said to have the major violation.*

We discard all the floorplans from each instance if there is atleast a single module with major violation. These floorplans are discarded because a module with major violation has to borrow the RAM/MUL resources from its neighbouring regions allocated to different modules. This might make a module non-contiguous and the shape of the module can be severely affected.

Grouping slicing trees across instances: The set of pruned floorplans in each instance have static modules

placed in the same location but might not have the exact shape after the rectangular region allocation. Thus the question of selecting a single floorplan from each instances arises where not only the position but the shapes of each static modules matches. Thus we find a set of groups, where each *group* consists of a slicing tree for each instance and the floorplans in each group are similar with respect to their cut lines or aspect ratios of static modules.

We calculate the aspect ratios of static modules in each floorplan for each instance. We group the floorplans from each instance on the basis of nearly equal aspect ratios of the static module such that a group contains at least one floorplan from each instance. If there is more than one candidate floorplan for an instance in the group, we need to select a single floorplan for that instance. We choose that particular floorplan from each instance which are similar with respect to their cut lines

Definition 4 *Distance between two slicing trees: Let a and b be the strings representing level order traversal of nodes from root till one level above the leaves of the two slicing trees respectively, with horizontal (vertical) cut represented as 0 (1). Let $l = min\{$length(a),length(b)$\}$ and length of the longer string be truncated till l from right. Then the distance between these trees is the number of ones in $a \oplus b$.*

This measures the closeness among two slicing trees in terms of slicing topology. In the context of partial reconfiguration, a schedule implies the ordering of the instances on the time line. To have same shapes of static modules from one instance to the consecutive one, the change in slicing tree must be minimum. Let, $t_1 < t_2 < \cdots t_k$ be the k sets of slicing trees for the k instances in a given schedule, where t_1 and t_k are the trees for the first and the last instances in the schedule. An associated distance digraph $G = (V, E)$ is defined with $v \in V$ corresponding to a slicing tree for some instance as shown in Fig. 3(a). There is a weighted edge $e \in E$ from u to v, if u and v correspond to the slicing trees in consecutive instances. The weight is the distance between u and v as in Definition 4. If there are k instances, we find a minimum weighted k-length path starting from the nodes corresponding to t_1 to those for t_k. The floorplans corresponding to the trees in the minimum weighted path are selected as the final floorplans for a group. There may be more than one minmum weighted k length path. Each of them correspond to a group of floorplans that can be considered as global floorplan.

Postprocessing for satisfying resource requirements: The slicing trees selected from each instance in a group have static modules with nearly equal aspect ratios but not exactly the same shape. We consider all pair of shapes, taking one from the list of S_L and the other from S_R. For all instances, we impose the respective shape in a shape pair to static modules at bottom-left and top-right corner of the floorplan. This requires reallocation of CLBs of some of the dynamic modules which are neighbours of static modules in each floorplan due to new overlaps generated by imposition of the exact shape of static modules. Now we reallocate CLBs of such modules along with the deficits generated earlier using the free regions available on the chip. We formulate a minimum cost maximum flow (MCMF) problem for each floorplan corresponding to a slicing tree in the group to resolve the deficit of CLBs in all instances. A network flow graph $N = (V_N, E_N)$ for each floorplan corresponding to a slicing tree of the group is defined, where N is a bipartite graph having a source node s_N and a sink node t_N. Let $V_N = L_N \cup R_N$. Each $v \in L_N$ corresponds to a module that is deficient of CLBs. Each $v \in R_N$ corresponds to the rectangular region if they have any free CLBs. Let $E_N = E_s \cup E_{uv} \cup E_t$. For each $u \in L_N$ there exists an edge $e \in E_s, e = (s_N, u)$ with capcity as the remaining number of CLBs of a module to be reallocated, i.e., the deficit, corresponding to u and cost as 1. For each $v \in R_N$ there exists an edge $e \in E_t, e = (v, t_N)$ with capacity as the number of free / unallocated CLBs in the rectangle corresponding to v and cost as 1. For each floorplan, a *rectangular dual graph* RD [11] is generated from the adjacency relationship of rectangles. For each $u \in L_N$, and for each $v \in R_N$, there exists an edge $e \in E_{uv}$ with capacity equal to the number of free CLBs in rectangle corresponding to v. The cost is the length of the shortest path in RD from the vertex in RD corresponding to u to the vertex in RD corresponding to v. Figure 3(b) shows one such network flow graph. By solving MCMF, if the amount of flow is equal to the total deficit of CLBs, then these deficit CLBs corresponding to each $u \in L_N$ is satisfied by its neighbouring rectangles. For each edge $e = (u, v) \in E_{uv}$ having a positive flow f and cost c implies that module corresponding to u borrows f CLBs from the rectangle corresponding to v following the c length path in RD, from the vertex in RD corresponding to u to the vertex in RD corresponding to v. This results in rectilinear shape of a module in a floorplan. If MCMF does not give a solution for any one of the floorplan in a group, this group is rejected as a candidate solution for the partial reconfiguration problem.

Finally, the RAM/MULs of each module are allocated by minimum weighted bipartite matching formulation as described in [2] for that group of floorplans, where each floorplan is feasible in terms of CLBs. This gives the final floorplans for each instance in partial reconfiguration problem. Since there may be more than one group with feasible solution, we choose a group with feasible floorplans of all instances with minimum sum of HPWL over all instances. The proposed method is illustrated with an example in [7].

If k is the number of instances, and h, the maximum number of signal nets in any instance, the time complexity of first phase is $O(kh)$ [9]. If q is the maximum num-

Table 1. Floorplans of individual instance vs. global floorplan; SM: Static modules

Benchmark details				HPWL			CPU time(s)		Overlap of SM (%)	
ckt	# inst.	# SM	max # modules, nets	Global	Indiv	Avg. Incr.(%)	Global	Indiv.	Global	Indiv.
b1	5	4	31,660	268184	212094	26	6.5	5.3	100	2
b2	5	2	31,527	186247	141391	31	6.6	5.0	100	0
b3	6	3	33,510	285659	232910	22	7.9	6.5	100	9
b4	6	4	29,486	223642	180048	24	7.6	5.8	100	8
b5	6	2	31,450	264308	200501	31	7.9	5.9	100	1
b6	7	3	30,510	308010	247481	24	8.9	6.9	100	7
b7	8	3	34,500	330114	237856	38	9.8	8.3	100	7
b8	9	5	30,420	379354	287604	31	11.4	9.2	100	1
b9	10	3	29,544	326026	257314	26	12.2	8.6	100	17

ber of shapes generated for a module and n, the maximum number of modules in any instance, the time complexity of second phase is $O(kqn^2)$ [2]. The time complexity of the third phase is $O(kn^3)$. Thus, total time complexity of the proposed method is $O(k(h + n^3))$.

7. Experimental Results

We implemented the proposed method in C on Unix using hMetis[6] and LEDA [10] library on 1.2 GHz Sun-Blade 2000 workstation and obtained results for 9 synthetic benchmarks. The first four columns of Table 1 shows the number of instances, number of static modules, maximum number of modules and signal nets in each benchmark.

The total wirelength obtained by summing up the HPWL over all instances is compared with the total wirelength obtained if each individual instance is floorplanned optimally. Column 7 shows the average increase in wirelength. Over nine benchmark circuits, the average increase in wirelength is 28%, while the time taken to generate the global floorplans for all instances is $1.27\times$ of the total time taken for floorplanning each instance individually. These CPU times are shown in columns 8 and 9 for global and individual floorplan generation respectively. Our global floorplan generation method places the static modules of same shape at same location thereby yielding the overlap of static modules of consecutive instances to 100%. Whereas, in case of floorplanning individual instances consecutively, the overlap for static modules is only about 5.8% on the average. This shows that, with little increase in wirelength and with little extra time, it is possible to generate a set of floorplans for a given schedule satisfying all its resource requirement and yet causing least partial reconfiguration overhead. This shows the suitability of our fast deterministic floorplanning method for partial reconfiguration.

8. Conclusion

In this paper, we proposed a fast deterministic floorplanning method in the context of partial reconfiguration for FPGAs with heterogeneous resources. To reduce the reconfiguration overhead the static modules are placed on the board at a fixed location with same shapes at each instance of a given schedule, while remaining contiguous space is used for placing the dynamic modules. Experiments on a set of benchmark shows that being a deterministic method it is fast and it generates feasible floorplans for each of the instances of each benchmark with a small increase in wirelength compared to the optimal floorplan of individual instances.

References

[1] A. Ahmadinia, C. Bobda, S. P. Fekete, J. Teich, and J. C. van der Veen. Optimal free-space management and routing-conscious dynamic placement for reconfigurable devices. *IEEE Trans. Comput.*, 56(5):673–680, 2007.

[2] P. Banerjee, S. Sur-Kolay, and A. Bishnu. Floorplanning in modern FPGAs. In *Proc. of the* 20^{th} *Intl. Conf. on VLSI Design and* 6^{th} *Intl. Conf. on Embedded Systems Design*, pages 893–898. IEEE Computer Society, 2007.

[3] K. Bazargan, R. Kastner, and M. Sarrafzadeh. Fast template placement for reconfigurable computing systems. *IEEE Des. Test*, 17(1):68–83, 2000.

[4] L. Cheng and M. D. F. Wong. Floorplan design for multimillion gate FPGAs. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(12):2795–2805, 2006.

[5] Y. Feng and D. P. Mehta. Heterogeneous floorplanning for FPGAs. In *Proc. of the* 19^{th} *Intl. Conf. on VLSI Design and* 5^{th} *Intl. Conf. on Embedded Systems Design*, pages 257–262. IEEE Computer Society, 2006.

[6] http://www-users.cs.umn.edu/ karypis/metis/hmetis.

[7] http://www.isical.ac.in/˜pritha_r/BSS_partialfloorplan.pdf.

[8] http://wwww.xilinx.com.

[9] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Trans. on VLSI Systems*, 7(1):69–79, 1999.

[10] http://www.algorithmic-solutions.com/.

[11] M. Sarrafzadeh and C. Wong. *An Introduction to VLSI Physical Design*. Mcgraw Hill, 1996.

[12] L. Singhal and E. Bozorgzadeh. Multi-layer floorplanning on a sequence of reconfigurable designs. In *Field Programmable Logic and Applications*, pages 1–8, 2006.

2009 22nd International Conference on VLSI Design

Efficient Synthesis of a Uniformly Spread Layout Aware Pareto Surface for Analog Circuits

Almitra Pradhan, Ranga Vemuri
Dept. of ECE, University of Cincinnati, Cincinnati, OH 45219
{pradhaa,ranga}@ececs.uc.edu

***Abstract*— Accurate and fast optimization of analog circuits is an important requirement of current synthesis methods. Obtaining the entire pareto optimal surface for conflicting performance objectives is essential for design space exploration as well as circuit sizing. Layout parasitics prevent the circuit from realizing the estimated optimal performance values but are not considered in most existing pareto-front generation methods. We develop a layout-aware circuit matrix modeling method along with an efficient multi-objective optimizer to synthesize the parasitic inclusive pareto-optimal performance surface. The algorithm achieves a pareto surface with points spread uniformly in all regions. The sensitivity of critical performance to candidate design points is used to select the best sizing solution during synthesis. Experiments on benchmark circuits show the effectiveness of the proposed method in obtaining a speedup of an order of 10^3 with negligible loss of accuracy as compared to SPICE.**

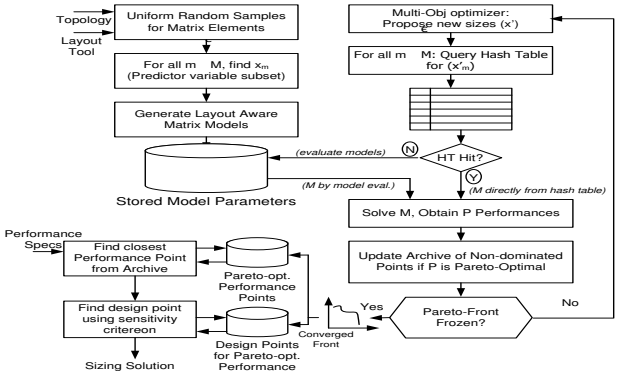


Fig. 1. Overview of the Proposed Approach

I. Introduction

Automated design and optimization of analog circuits is an important requirement during the synthesis process. Many tools developed for analog cell level design use a single objective optimization algorithm such as Simulated Annealing (SA), Genetic Algorithms (GA), Convex Programming (CP) at its core [1], [2]. An analog circuit is required to meet several performance specifications which are often conflicting in nature. Deriving the entire set of performance curves, called pareto surfaces, for a given circuit topology helps in analyzing performance tradeoffs at various design points.

Pareto surface generation is essentially a problem of multi-objective optimization and both stochastic and deterministic methods have been proposed for its solution. Watson [3], a tradeoff analysis tool for analog circuits, uses a GA and simulation to find the pareto-optimal performances. Tiwary *et al.* [4] use epochs of GA-SA along with simulation. A large number of iterations are required for a converged pareto-optimal performance set which makes simulation based techniques expensive. Yu *et al.* [5] avoid simulation by using a Kriging based performance model. However, their converged pareto-front is obtained by a local search in the vicinity of an initial front generated by a random sampling. It is possible to miss several performance vectors using a *local* search. Deterministic methods such as Normal Boundary Intersection [6] have been applied, however they suffer from high computational complexity.

Generating pareto-optimal tradeoff surfaces of analog circuits finds application in performance space exploration as well as in a bottom up design approach. Performance space exploration attempts to find the boundaries of achievable performance values for a given circuit topology. During a top-down design procedure, constraints are propagated to lower level blocks. The appropriate topology (e.g. cascode or miller topology for an amplifier) can be selected based on the limits of attainable performance which is characterized by its pareto surface. Gielen *et al.* [7] use a pareto surface for circuit sizing.

Layout parasitics adversely affect the circuit performance and need to be accounted early during the synthesis flow. The pareto surface generated should be inclusive of layout effects so that the performance boundary predicted by it is synthesizable post layout. Generating the pareto surface requires the order of 10^6 iterations for convergence. Synthesizing the layout at each iteration is expensive and most of the existing pareto surface generation methods *do not* consider the effects of layout parasitics during curve generation.

The focus of this paper is the development of a procedure for the *layout aware pareto front* of analog circuits. We use an SA based multi-objective optimization algorithm due to its lower memory and cpu requirements. SA methods avoid sorting and ranking of the solution population as required by EAs and have been experimentally shown to find a larger set of non-dominated solutions [8], [9]. The pareto front generation algorithm has been modified to make use of sub-solution hashing to improve its efficiency. Layout aware circuit matrix models developed predict circuit performance accurately. This method is 3-4 orders of magnitude faster than layout and simulation which makes it practical. The improvement in speed is obtained with only about 1% loss in accuracy on average. Moreover, the proposed method generates a pareto

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.67

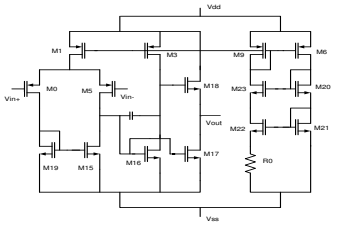


Fig. 2. An Operational Amplifier Circuit (SEO)

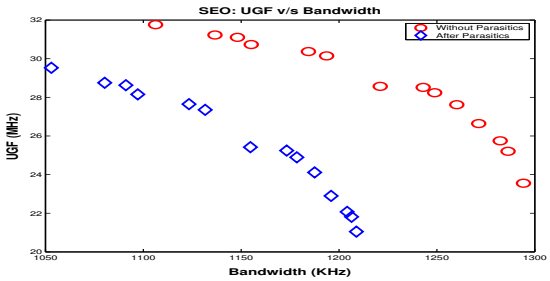


Fig. 3. Effect of layout parasitics on Pareto-optimal performances

surface with points distributed in all its regions which is essential for sizing the circuit using the generated front. Fig. 1 shows an overview of the proposed approach.

This paper is organized as follows. Section II outlines the parasitic aware matrix model development while section III describes the pareto surface generation procedure. section IV presents experiments and results and section V concludes the paper.

II. Circuit Matrix Model Generation

In [10], the authors have proposed matrix models for accurately predicting the performance of analog circuits in the design space. In this section we will briefly review the concept of circuit matrix models and also describe the method to extend these models to predict circuit performance considering layout parasitics.

$$\begin{aligned}(G + s * C)x &= B \\ y &= L^T * x\end{aligned} \tag{1}$$

Any linear(ized) circuit can be represented mathematically by its modified nodal analysis (MNA) formulation (eq. 1). The elements of conductance matrix G are a linear combination of small signal values of active devices such as g_m, g_{ds} and passive resistances. The matrix C elements are a combination of Mosfet capacitances and other capacitive elements in the circuit. For each unique matrix element g $\in$ G and c $\in$ C, data is gathered by spice simulation using a uniform random distribution in the circuit design space. Design variables affecting each matrix element are identified. Higher order polynomial response surface models are generated to predict the relation between a matrix element and design variables. For further details about model generation the reader is referred to [10]. Predicting performance with circuit matrix models is about an order faster than simulation.

The circuit matrix values as well as performance predicted by this method is very accurate and comparable to a Spice model. For the operational amplifier circuit shown in fig. 2, all the circuit matrix elements could be modeled accurately with a mean error of about 1%. The results for worst case errors over all circuit matrix element models (M) as well as the estimated performance parameters are reported in Table I. The accuracy results reported are on an independent 4000 point validation dataset.

To study the effect of parasitics on circuit performance we plot the bandwidth versus unity gain frequency for the amplifier as shown in Fig. 3. The red points (o)are on the front obtained when layout parasitics are ignored. Post-layout values for bandwidth and ugf corresponding to the points on the pareto-surface are shown by the blue points($\diamond$). Thus, the achievable performance predicted for this circuit cannot be attained in practice due to the presence of layout parasitics.

Layout-inclusive circuit matrix models are obtained as follows. Sample layouts are generated for a uniform random distribution of design points in the circuit parameter range. We have used a procedural layout generator (PLG) using the MSL package [11] as the layout tool. The PLG generates layouts using rules for device placement, spacing given by the user. Circuit parasitics are divided into two types (i) device parasitics (ii) non-device parasitics and models are developed for them separately.

(i) ***Device parasitics*** include the bulk capacitance such as c_{sb}, c_{db} of analog devices. These are dependent on the operating point as well as device size. They can be estimated using polynomial models with the bias, diffusion area A_s,A_d and perimeter P_s,P_d as predictor variables. Since diffusion area and perimeter are highly correlated, they are centered and normalized before model fitting. For analog modules, functions for the diffusion area and perimeter are written using the PLG properties. A library of such functions is pre-developed for and used in the estimation of parasitics without actual layout generation.

TABLE I
Validating Model Accuracy

Relative Error (%)	**Max**	**Mean**	**Std. Dev.**
Worst Case Errors ($\forall M$)	1.4	0.27	0.43
Worst Errors ($\forall Parasitics$)	3.0	1.8	1.9
Using Circuit Level Models			
Gain	0.14	0.0015	0.0052
Bandwidth	0.57	0.013	0.04
UGF	0.33	0.0057	0.02
Phase Margin	1.76	0.013	0.05
Using Layout Aware Models			
Gain	0.6	0.18	0.1
Bandwidth	1.2	0.46	0.3
UGF	2.4	0.56	0.5
Phase Margin	3.4	0.96	0.9

(ii) ***Non-device parasitics*** include various area and coupling capacitances extracted from the layout. These are dependent on the device width and also the final routing achieved. These

parasitics cannot be predicted exactly like device parasitics, however linear models generated using device widths and placement rules give acceptable accuracy.

Parasitic models for device and interconnect parasitics are developed as described above. Gate level matrix elements are modeled as given in [10]. A unified parasitic matrix element model is generated based on MNA equations. The circuit matrix model thus obtained now predicts the parasitic-aware performance. Table I shows the accuracy of the layout aware performance prediction for the op-amp circuit using models developed by the proposed approach. Average error is less than 1% while the worst case error is about 3.4%.

III. Extracting the Pareto-Optimal Performance Curves

The notion of pareto-optimality exists in multi-objective optimization problems. Here several objectives need to be max(min)imized simultaneously. However, these objectives may be contradictory and it is not possible to obtain a single best solution. Instead we may obtain several *pareto-optimal* solutions, each of which is best with respect to a certain objective.

A. Problem Formulation

For an analog circuit, measures such as gain, bandwidth, phase margin are the performance parameters of interest. These performance parameters are often conflicting and it is not possible to improve a single performance objective without deteriorating some others. The general multi-objective optimization problem with N objectives is formulated as given by eq.(2). Here, $P(x)$ is the objective vector of N performance parameters that have to be maximized and x is a k-dimensional vector representing the design variables within a parameter space R.

$$\begin{aligned} Maximize \quad & P(x) = [P_1(x), P_2(x), ..., P_N(x)] \\ & x = [x_1, x_2, ..., x_k] \in R \end{aligned} \tag{2}$$

Pareto-dominance and pareto-optimality are used to compare performance vectors and find the optimal performance values. A performance vector P_a = [P_{a1}(x), P_{a2}(x), ... , P_{aN}(x)] is said to dominate the performance vector P_b = [P_{b1}(x), P_{b2}(x), ..., P_{bN}(x)], i.e. $P_a \succ P_b$ in a maximization context, if and only if:

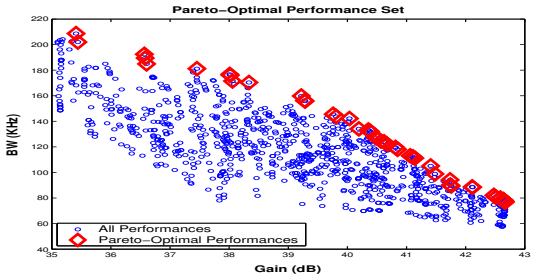


Fig. 4. Pareto-Optimal performance sets

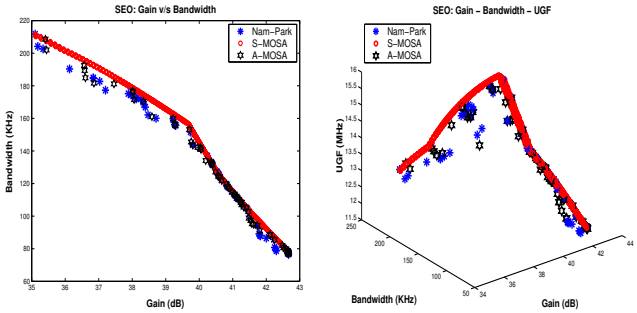


Fig. 5. Pareto front generated by the three MOSA methods

$$\begin{aligned} & \forall i \in \{1, .., N\}, P_{ai} \geq P_{ai}, \quad and \\ & \exists j \in \{1, .., N\}, P_{aj} > P_{aj} \end{aligned} \tag{3}$$

A performance vector is pareto-optimal if there does not exist any other set of performance values that dominate it. The pareto-surface typically does not have a closed form and is described by the set of points that represent the multi-objective tradeoffs. Fig. 4 shows the set of pareto-optimal points among the achievable performance for the operational amplifier circuit.

B. Front Generation Methods

We use a Multi-Objective SA (MOSA) for pareto surface generation. We tested three MOSA based methods, Nam-Park [12], S-Mosa [13] and A-Mosa [9] to find their suitability for the analog circuits domain. Table II shows the number of performance points and fig. 5 shows the actual pareto-optimal surfaces generated by each method for the amplifier.

The front coverage index (*fci*) metric is developed to measure uniformity of pareto point distribution along the surface. Each dimension of the pareto surface is divided into small and equally spaced intervals. The fci is a fraction of the number of populated intervals to the total number of intervals. An fci of 1.0 indicates a uniform spread of points. For analog circuit optimization, the S-Mosa gave better results both in terms of the number and spread of points found.

$$\delta E(x, x') = \frac{\mid F_x \mid - \mid F_{x'} \mid}{\mid F \mid} \tag{4}$$

S-Mosa uses eq.(4) as the energy function where F is the front and F_x is the dominated set. Candidate solutions (x′) that are dominated by fewer points have a lower energy level and are more likely to be accepted into the surface (archive). This favors discovery of pareto-optimal points belonging to less explored regions of the front [13]. We require a good spread of points along the entire front to be able to use it during the synthesis process.

C. Improving Efficiency of Pareto Curve Generation

We have improved the existing S-Mosa algorithm to incorporate hashing of circuit matrix element values during pareto front generation. An optimizer like S-Mosa visits various

TABLE II

COMPARISON OF THREE MOSA ALGORITHMS

Number of Pareto-Optimal Points Found			
	Nam-Park	S-MOSA	A-MOSA
Gain-Bw	72	3081	72
Gain-Bw-UGF	72	4692	78
Front Coverage Index *(fci)*			
	Nam-Park	S-MOSA	A-MOSA
Gain-Bw	0.42	0.98	0.43
Gain-Bw-UGF	0.45	0.99	0.42

points in the design space and evaluates the circuit matrix and performance at each point. Although a good optimizer does not visit the same design point repeatedly, hashing is possible in our models due to two important properties of the matrix elements:

- Each element is dependent on only a subset of design variables
- Multiple elements depend on a given variable subset

In any SA, candidate solutions are proposed by perturbing one of the design variables of the current solution. During an SA run, although the proposed solutions are unlikely to be repeated, *common subsets* of variable values often occur among the proposed solutions. As matrix elements are functions of only a *subset* of design variables, their values once calculated can be reused whenever such variable subsets are repeated during optimization.

Hash tables are used for the storage and reuse for circuit matrix element values. Hash tables are implemented efficiently by building a single table for all matrix elements that are functions of a common variable subset. Such a hashing scheme can avoid many recomputations and make the SA faster. Without hashing all matrix elements have to be evaluated in each iteration, whereas with our method only the hash table misses have to be evaluated.

Obtaining the pareto front requires a large number of optimizer iterations within the circuit's design space. A hashed-SA is expected to give increasing returns, as greater exploration of the design space results in a large number of common sub-solutions (for which matrix element values are obtained directly from the hash table). Fig. 6 shows the reduction in hash table misses with increasing iterations of the optimizer which supports our expectation.

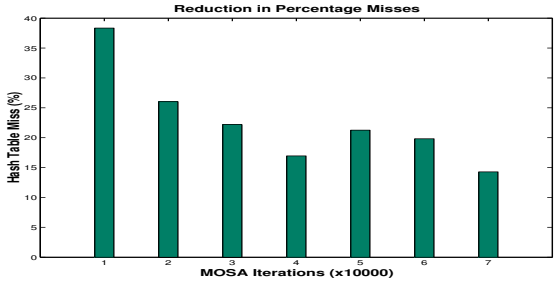


Fig. 6. Reduction in Hash Table Misses for MOSA run on SEO

Along with the pareto front as stored by S-Mosa we also save (i) the design points that achieve the pareto-optimal

Algorithm 1 Hashed-MOSA

Input: Circuit Topology(SPICE), Design Variables x, Performance of Interest {P}
Output: Pareto-optimal vectors for {P}

Model Generation

{U} ← Uniform random values for design variables
∀ u ∈ {U} :
  Sample layout
  Get layout-aware matrix element values
∀ m ∈ {M}:
  Identify predictor variable subset
  Fit polynomial regression models
  Initiate hash tables H_M

Pareto Surface Generation

while (Pareto-set ! frozen) **do**
  while (T != Equilibrium) **do**
    x′ = perturb(x)
    for $m \in$ {M} **do**
      x'_m = predictor_subset(x′)
      if ($x'_m \in H_M$) **then**
        m(x′) ← H_M
      else
        m(x′) ← Model Evaluation
        Add computed m(x′) value to H_M
      end if
    end for
    P(x′) = [P_1(x′) ··· P_N(x′)] ← Solve M(x′)
    F(x′) ← Archive pts dominated by x′
    F(x) ← Archive pts dominated by x
    Δ E = | F(x′) - F(x) | / | F |
    if (random ≤ - Δ E / T) **then**
      x′ = x
    end if
    Update Archive
  end while
  T ← α * T
end while

performance (ii) performance sensitivity value. The sensitivity of a performance parameter at a design point is given by the percentage change caused by small perturbations in the circuit variables at that point. In our application we choose the least sensitive design point for implementing the final sizing solution.

D. Algorithm

This section summarizes the layout-aware pareto front generation of analog circuits using a hashed-MOSA approach. First, parasitic-aware matrix element models are generated from layout samples. Optimization uses these models instead of evaluating performance by simulation and layout synthesis. When a new candidate (x′) is proposed, solution subsets corresponding to predictor variables of matrix elements are formed. If the matrix element value at this subset was computed earlier it is simply fetched from the hash table else it has to be obtained by model computation. The candidate solution is accepted if it dominates more points in the archive. The archive is updated so that it only contains non-dominating performance vectors. The procedure converges when no new pareto-optimal points can be detected (frozen state).

E. Sizing procedure using a generated Pareto Front

Using the above algorithm we obtain an archive of pareto optimal performance vectors and corresponding design points. During synthesis, block level specifications are derived by propagating top-level specifications. Once these are obtained, the appropriate cell topology is chosen by referring to its performance space boundary. The performance vector from the archive that is closest to the cell specification is selected using a fast nearest neighbor approach.

Several design points can meet the performance requirements and may be saved as candidates for a particular performance objective. We use the sensitivity of critical performance parameter to the design point to choose the correct design point. For an application where realizing a high gain is critical a design point to which gain is least sensitive is chosen. The user indicates critical parameters based on the application by a weight assignment.

IV. Experiments and Results

In this section we present results for accuracy, speed and parasitic aware pareto front generation for three benchmark analog circuits. An improved archive based optimization algorithm that uses dominance set cardinality as the acceptance measure is selected for the pareto front generation. Layout aware circuit matrix models are used for fast and accurate performance calculation whereas hashing improves the optimizer efficiency. A C++ based tool that incorporates these features has been developed. Experiments are run on a 2048 Mb, 2 $\times$ 750 MHz Solaris Sunblade 1000 workstation.

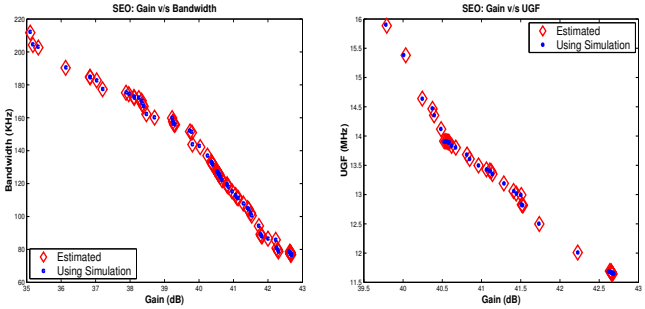


Fig. 7. Accuracy comparison - Model v/s Simulation based Front

Expt I - Single Ended Op-amp: The first benchmark circuit used is the Single Ended Op-amp (SEO) as shown in fig. 2. This circuit has 15 transistors. Six design variables are identified for the circuit after applying matching requirements. The design variables include widths of various transistors and have values ranging from 25u-300u. The results of pareto front generation using the proposed method is compared to a competing simulation based method.

Fig. 7 shows the pareto front generated using the developed circuit matrix models (red - $\diamond$) and also by simulation (blue - o). The estimated and actual performance points are almost overlapping with a negligible relative error. Thus, the pareto front generated is of a comparable accuracy as that obtained using simulation. Performance is calculated from the estimated circuit matrix values and numerical simulation is avoided, the proposed approach is also faster. The total speedup achieved by our approach is about 12000 $\times$ compared to simulation as shown in table III.

TABLE III
SPEEDUP USING PROPOSED METHOD (COMPARED TO SIMULATION)

Circuit	Speedup (model)	Speedup (hashing)	Total Speedup
SEO	4000x	3x	12000x
DA	2400x	2.6x	6240x
BPF	2800x	2.5x	7000x

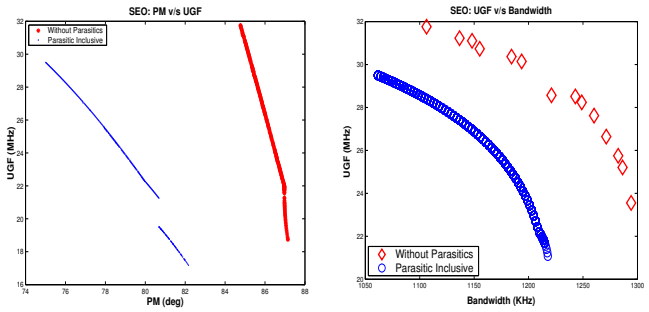


Fig. 8. SEO: (i) PM (deg) vs UGF (KHz) (ii) BW (KHz) vs UGF (KHz)

Fig. 8 shows the pareto front obtained by using the layout-level models for the circuit matrix elements. The figure also shows the pareto front generated using the original transistor-level circuit models of [10]. The pareto fronts (i, ii) have 326 and 2000 points respectively and are generated in 11.7 minutes. The layout-aware front varies from the one obtained when parasitics were ignored. In the Phase Margin versus Bandwidth plot some performance values are not realizable due to the effect of parasitics. This behavior cannot be predicted if only the transistor models are used. For the SEO, front (i) had an front coverage index (fci) of 93% and front (ii) had 65% coverage.

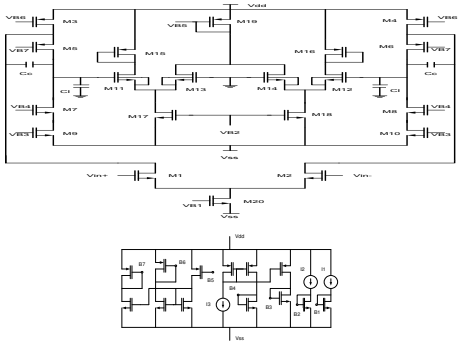


Fig. 9. Differential Amplifier Schematic

Expt. II - Differential Op-amp: The second benchmark circuit is a differential amplifier (DA) as shown in Fig. 9. This circuit has 33 transistors and five variables. They include four device lengths varying from 40u-200u and a coupling capacitor having range 10pF-50pF. The pareto optimal performance

points obtained for this circuit are shown in fig. 10. The fronts (i, ii) have 367, 152 points respectively and required 10.5 minutes for synthesis. Generating the pareto front using models is about 6240 $\times$ faster than obtaining it using simulation. For the DA, front (i) is generated with 57% coverage and front (ii) with 52% coverage.

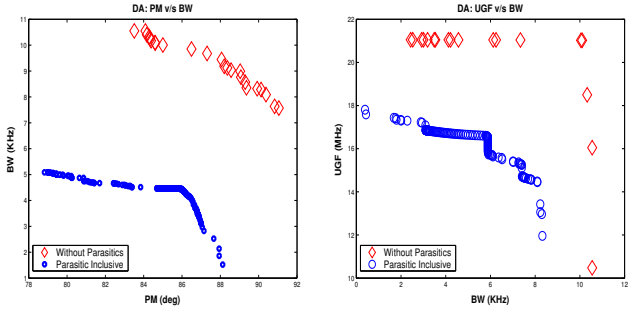


Fig. 10. DA: (i) BW (kHz) vs PM (deg) (ii) UGF (KHz) vs BW (KHz)

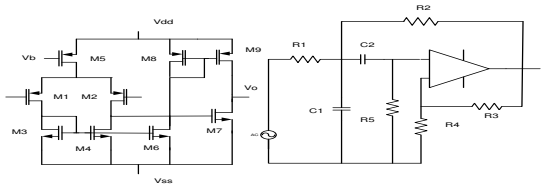


Fig. 11. Bandpass Filter Schematic

Expt. III - Bandpass Filter: The third circuit is a second order Sallen-Key bandpass filter (BPF) as shown in Fig. 11. This circuit has 9 transistors and 4 variables. The layout-inclusive pareto optimal performance points obtained for this circuit are shown in fig. 12. The pareto fronts (i, ii) are generated with 440, 370 points respectively in 16 minutes. As the Q factor is being optimized for minimization it has been included with a negative sign. For the BPF, the proposed method was about 7000 $\times$ faster than SPICE and layout generation. For the BPF, front (i) is generated with 53% uniform coverage and front (ii) with 50% uniform coverage.

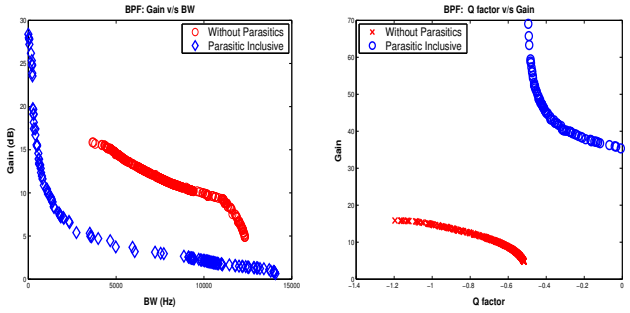


Fig. 12. BPF:(i) Gain (dB) vs BW (Hz) (ii) Gain (dB) vs Q

Expt. IV - Sizing from generated Pareto front: The pareto optimal performance curve generated is then used for sizing the circuit from its specifications. The closest pareto optimal point to the target specification is chosen through a closest neighbor searching method. Table IV shows the specification

TABLE IV
CIRCUIT SIZING FROM PARETO CURVE (SEO)

Specification	Estimated	Verified (Hspice)
Gain $\geq$ 40 dB	40.5	40.55
UGF $\geq$ 14 MHz	14.1	14.06
Bandwidth $\geq$ 120 KHz	124.53	124.85
Design point: 300u, 250u, 300u, 145u, 131u, 80u		

provided, performance achieved and verified for the SEO. In the design point archive, 11 points were able to meet this performance. The final design point chosen for synthesis was the one for which bandwidth (chosen critical performance) was least sensitive.

V. CONCLUSION

This paper developed a pareto-surface generation algorithm based on the multi-objective simulated annealing and parasitic aware circuit matrix models. The method is successful in generating pareto-optimal performance curves with a uniform spread of points. The entire pareto optimal surfaces were generated about 6000x to 12000x faster than using a simulation based method. Using layout aware pareto curves during topology selection or sizing leads to designs with parasitic closure.

REFERENCES

[1] G. Wolfe and R. Vemuri, "Extraction and use of neural network models in automated synthesis of operational amplifiers," *IEEE Trans. CADICS*, vol. 22, no. 2, pp. 198–212, 2003.

[2] F. D. Bernardinis, M. I. Jordan, and A. S. Vincentelli, "Support vector machines for analog circuit performance representation," in *Proc. DAC '03*, 2003, pp. 964–969.

[3] B. De Smedt and G. Gielen, "Watson: design space boundary exploration and model generation for analog and rfic design," *IEEE Trans. CADICS*, vol. 22, no. 2, pp. 213–224, Feb. 2003.

[4] S. K. Tiwary, P. K. Tiwary, and R. A. Rutenbar, "Generation of yield-aware pareto surfaces for hierarchical circuit design space exploration," in *Proc. DAC*, 2006, pp. 31–36.

[5] G. Yu and P. Li, "Yield-aware analog integrated circuit optimization using geostatistics motivated performance modeling," in *Proc. of ICCAD*, 2007, pp. 464–469.

[6] G. Stehr, H. Graeb, and K. Antreich, "Performance trade-off analysis of analog circuits by normal-boundary intersection," in *Proc. of DAC*, 2003, pp. 958–963.

[7] G. Gielen, T. McConaghy, and T. Eeckelaert, "Performance space modeling for hierarchical synthesis of analog integrated circuits," in *Proc. of DAC*, 2005, pp. 881–6.

[8] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," *Jnl of the Operational Rsrch Soc.*, vol. 57, pp. 1143–60, 2006.

[9] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," *IEEE Trans on Evol. Comp.*, vol. 12, no. 3, pp. 269–283, 2008.

[10] A. Pradhan and R. Vemuri, "Regression based circuit matrix models for accurate performance estimation of analog circuits," in *Proc. of IFIP VLSI-SOC*, 2007, pp. 48–53.

[11] H. Sampath and R. Vemuri, "MSL a high level language for parameterized analog and mixed signal layout generators," in *Proc of the 12th IFIP VLSI Conf.*, 2003, pp. 416–421.

[12] D. Nam and C. Park, "Multiobjective simulated annealing: a comparative study to evolutionary algorithms," *Int. J. Fuzzy Systems*, vol. 2, no. 2, pp. 87–97, 2000.

[13] K. Smith, R. Everson, and J. Fieldsend, "Dominance measures for multi-objective simulated annealing," *Evol. Comp., Cong. on*, vol. 1, pp. 23–30, 2004.

Efficient Analog/RF Layout Closure with Compaction Based Legalization

Subramanian Rajagopalan, Sambuddha Bhattacharya, Shabbir H. Batterywala
ATG, Synopsys (India) Pvt. Ltd., Bangalore, 560016, India
{rsubbu, sbb, battery}@synopsys.com

Abstract—
Advancements in process technology have resulted in tremendous increase in the number of design rules. This has greatly complicated the task of building design rule clean layouts. While EDA tools aid in layout creation for standard cell based ASICs, the problem remains unsolved for custom, analog and RF circuits. For such circuits, layout designers spend lot of time converting functionally correct schematic circuits into acceptable design rule clean layouts. While techniques have been proposed to remove Design Rule Violations (DRVs) with minimum perturbation to hand crafted layouts, designers still spend lot of time to get to layout closure. In the proposed methodology, designers can quickly draw *sparse* and possibly *design rule unclean* layouts and then use a compaction based layout legalization to clean up the DRVs and reduce area. This increases the productivity of layout designers and reduces the turnaround time for layout closure. The proposed technique achieves close to best possible area for a given sparse layout, keeps hard macros unaltered, respects relative positions, and removes all violations of modeled design rules. Reported experimental results suggest that this method can be used to automate layout creation process.

I. Introduction

Advancements in fabrication processes have facilitated IC manufacturing in two key ways. One, the scaling in device geometry. And other, the capability to put heterogeneous devices on the same die. While the former is achieved with sub-wavelength lithography the later is facilitated by process recipes which allow devices of different threshold voltages (V_{th}), supply voltages (V_{dd}) etc. The result of these advancements is that the number of design rules have increased significantly. Simple width and spacing rules have changed into complex width based spacing rules, length based width rules, forbidden spacing rules, etc. These rules are primarily there due to the inability to print various patterns with sub-wavelength lithography. In order to facilitate low threshold, high threshold, and ultra high threshold NMOS/PMOS transistors, foundries have introduced several implant layers. Different width specifications for polysilicon gates are specified to support devices of different supply voltages. Consequently, the number of rules that layout designers must comply with has increased tremendously.

Increased number of design rules, particularly the context dependent rules, pose a major challenge to correct layout construction. EDA tools have constantly tried to cope up with this challenge. Digital ASIC designers have access to lot of commercially available tools to build layouts and fix DRVs to get to *layout closure*. Tools [1][2] are available for creating design rule compliant standard cell layouts, which are used as building blocks in ASICs. Placement and routing tools [3][4] are constantly extended to support new design rules which come out of fabrication houses.

Unlike digital ASICs, the task of creating design rule clean layouts is especially difficult for analog/RF and custom digital designs as they are usually drawn manually using layout editors. Furthermore, there are several iterations between circuit sizing and layout construction. Thus, at each stage a designer needs to ensure a design rule clean layout through tedious manual edits. Clearly there is a need for automation in layout creation. To this end, most of the layout solutions are available within Layout Editors. These editors allow reuse through foundry supplied layout components as Foundry Tool Kit / Process Design Kit (FTK/PDK)[5], design rule aware editing, sophisticated editing features to create arrayed instances and align layout shapes, etc. Commonly used Pcell [6] based methodology also aids by providing automatic creation and modification of layout components.

Despite these techniques, designers still spend a lot of time building minimum area design rule clean layouts. Updating these area optimized layouts is also time consuming. For example, to fix a spacing violation in a tight neighborhood would require moving lots of adjacent geometry to create extra space. An automation tool which allows designers to connect the components without worrying about area and complex design rules would be useful. Such a tool should however be able to remove most of the DRVs and minimize area. An illustration of this feature is provided in Figures 1 and 2. In Figure 1, a sparsely drawn layout is presented that connects four layout instances or macros. Figure 2 shows the same layout after running the compaction based layout legalization proposed in this paper. While the original layout had three DRVs in the interconnects, the final output had none.

Layout legalization is defined as the automatic correction of design rule violations. It is usually modeled as modified compaction problem [7]. The idea is to perform line sweep on layout geometry to capture applicable design rules as constraints, and solve these constraints to remove DRVs. These constraints are linear and can be modeled through either constraint graphs or linear programs (LP) [8]. If the objective is to minimize only the area this results into standard compaction. Reference [9] surveys' techniques for area minimization of layouts while obeying simple design rules.

In contrast to traditional compaction, [7] proposed a layout legalization technique that minimizes perturbation of layout edges. It is particularly effective in the presence of contextual design rules that are often approximated with a collection of simpler rules. Minimizing perturbations avoids inadvertent introduction of context dependent DRVs. The work in [10] proposed 'geometric closeness' objective that removes DRVs by doing minimum perturbations of widths of layout tiles and white spaces. The techniques in [11] and [12] focus on compaction under restrictive design rules such as coarse grids for transistor

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.61

gates. In general, reported methods either remove white spaces by moving layout geometry or remove DRVs by minimum perturbation, but do not do both.

In this paper, a systematic framework for simultaneously performing compaction and legalization is presented. A novel objective function, spCompact (structure preserving compactor), is proposed that minimizes the bounding box area and minimizes perturbation on width, extension, overlap, and connectivity constraints. Thus it retains layout structure and avoids introduction of context dependent DRVs. White space is reduced indirectly to achieve compaction since a direct minimization of spacing between layout rectangles can destroy the structure of layout. Any user specified layout component (or *macro*) is kept unaltered during legalization. Specifically, the spCompact objective function tries to achieve simultaneous compaction and legalization by using a subset of the generated constraints to build the objective function.

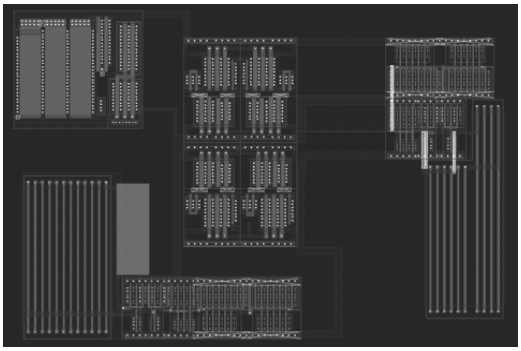


Fig. 1. Sparsely drawn layout

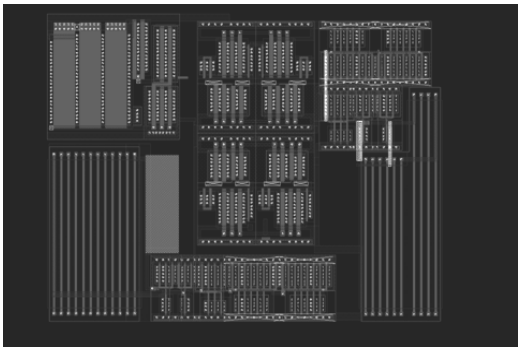


Fig. 2. Post compaction of sparsely drawn layout in Figure 1

The paper is organized as follows. Section II gives a background on compaction, including constraint generation and prior work on objective functions. Section III describes the proposed objective function, spCompact. Section IV provides some experimental results and comparison with other objective functions. Finally, section V concludes the paper.

II. Background

This section provides a brief overview of layout compaction. Layouts consist of a set of mask layers each with a collection of rectangles. A Compactor perturbs the set of rectangles in the layout to achieve the desired objective while obeying technology design rules. Each rectangle edge is considered as a positional variable. Constraints are imposed on the rectangle edges according to the design rules and layout connectivity. The constraints are of the form $x_i - x_j \geq val$, where x_i and x_j are the positional variables corresponding to two rectangle edges, and val corresponds to the particular rule value. If all rectangle edges, horizontal as well as vertical, are considered together this results into a two dimensional (2D) compaction. Typically, for reducing problem sizes and having acceptable run-times, compaction is done in one dimension at a time.

One dimensional (1D) layout compaction decouples constraints between horizontal and vertical edges of rectangles. The usual approach for 1D compaction is to generate constraints in horizontal direction, solve them, update the layout, repeat the same in vertical direction, and iterate till convergence is achieved. Minimizing the layout area is the common objective and that is achieved by minimizing the height and width of the layout. The problem can be modeled as a standard Linear Program (LP) formulation given in equations 1 and 2 where, **x** is a column vector consisting of the variables corresponding to rectangle edges, the vector **b** represents the rule values, and the vector **c** represents the objective function to be minimized.

$$minimize \quad \mathbf{c^T x} \tag{1}$$

$$subject\ to: \quad \mathbf{Ax} >= \mathbf{b} \tag{2}$$

The matrix **A** in equation 2 is the coefficient matrix for the constraints, where each row A_i represents a single constraint. If all constraints are difference type and objective function is also a difference of two variables then graph based longest path solvers can be used for quick solution [9]. However, the objective function is typically a general linear function, in which case an LP solver (either general simplex, or network simplex) is used. Moreover, since layouts typically are drawn on integer grid, it is required to have an integral solution. Method reported in [12] suggests techniques to achieve this without using slow integer linear program solvers. Furthermore, often layouts are assembled bottom-up (either from FTK/PDK components or pre-done macros) which necessitates the instances to remain unmodified during compaction. The technique proposed in [13] addresses this problem by replacing edge variables inside these macros by constants. This not only ensures difference type constraints but also reduces number of variables.

Layout legalization consists of three major steps: (a) constraint generation (b) objective function selection, and (c) constraint solution. Constraint generation is generally achieved by sweeping an imaginary line across the input layout. Algorithmic aspects of constraint generation are described in [14]. The work in [15] presents methods for modeling conditional design rules. Techniques for edge based modeling of some context dependent rules were reported in [16].

A key component of constraint solution is the objective function used in the LP. The objective function provides better control over the final layout. Even though layout legalization techniques are derived from a compaction formulation the objective need not be restricted to minimizing widths/heights of layouts.

This is specifically true for manually drawn analog layouts. Often the legalized layout is required to inherit all the structural properties of input layout. These properties include, symmetrically drawn components, aligned transistors and interconnect lines, geometry that is wider than minimum width, pre-done macros, etc. A legalized layout with pure compaction as objective could end up disturbing symmetry, resizing all geometry close to minimum width, modifying macro geometry etc., which is unacceptable. The general technique then is to add constraints to capture the *designer's intent* and then use a suitable objective function in LP optimization. In the past several objective functions have been proposed. These are briefly mentioned here.

- *Minimum-Area* objective function [9] is one of the earliest proposed objectives. When the constraints are all of difference type, longest path on the constraint graph built with the constraints as edges gives the minimum width of layout in 1D. While this solution is very fast, it does not offer a control over the structure of the layout and the longest path approach is not applicable directly in the presence of non-difference linear constraints like symmetry constraints.
- *Minimum-Perturbation* objective function [7] is used in legalization to remove DRVs in a layout by moving edges minimally. This approach retains the output layout very similar to the input layout. However this is not very effective in reducing the area of the layout.
- *Geometric Closeness* objective function [10] was proposed as a strategy for migration of layouts from one technology to another technology while retaining the structure of the layout. It penalizes any change to a layout tile, metal and space tiles alike, as opposed to a layout edge in minimum perturbation approach.
- *Wire Length Minimization* objective function [9] minimizes the sum of the lengths all the wires in the layout. While this objective is good for parasitics and often for area, the output layout typically is structurally very different from the input layout as the shape of polygons are minimized.

While the objective functions proposed in the past are good at minimizing one particular objective, they often sacrifice on other fronts. In this work, spCompact, an objective function that minimizes the area while preserving the structure of the layout is proposed. The key difference between this approach and the other approaches in the past are as follows.

- The proposed objective minimizes area while preserving the structure of the layout.
- All layout constraints generated by line sweep are used to guide construction of the objective function. This is unlike previous techniques, where only rectangle widths and spacings were used in objective function.

III. spCompact Objective Function

The aim of this work is to build an objective function for LP that makes the output layout design rule clean, reduce the area of the layout, and keep the output layout structurally similar to the input layout. This section describes how such an objective function, spCompact, can be built using the layout constraints for 1D compaction. Section III-A gives the basis for setting up the objective, section III-B describes the proposed spCompact objective function, and finally, section III-C describes how to estimate the minimum achievable layout area.

A. Formulation

The input layout is constituted of macro instances, interconnects and other layout components. Macros are kept unaltered during compaction and are referred as fixed-instances. So the input layout is constituted of a set of fixed-instances and *flexible* rectangles. Each fixed-instance in-turn is a set of *fixed* rectangles. A *flexible* rectangle is free to be modified, whereas a *fixed* rectangle can only be translated as determined by the translation of the fixed-instance to which it belongs. Let the input layout, L, be constituted of a set of *flexible* rectangles, $R = R_1, R_2, \ldots, R_n, n \in N$, and a set of fixed-instances $F = f_1, f_2, \ldots, f_m, m \in N$. Let R^f be the set of all *fixed* rectangles in the layout. Let each fixed-instance, $f_i \in F$ be a set of *fixed* rectangles R_i^f, such that, $R_i^f \subset R^f$, $R_i^f \cap R_j^f$ is empty $\forall i \neq j$, and $\forall i, R_i^f \cap R$ is empty.

Each *flexible* rectangle edge is assigned an id, $x_i, 1 \leq i \leq 2n$, and each fixed-instance is assigned an id, $x_j, (2n+1) \leq j \leq (2n+m)$, that corresponds to the position. For a fixed-instance, the variable can be assumed to correspond to the location of the origin of the design that has been instantiated. Let x_i^0 represent the original layout position of x_i. The position of a *fixed* rectangle edge can then be represented as $x_j + c_j, (2n+1) \leq j \leq (2n+m)$ where c_j is an integer constant decided by the position of the *fixed* rectangle in its corresponding fixed-instance. Figure 3(a) shows a design with a single rectangle which is then instantiated in Figure 3(b). In Figure 3(b), the two edges of the *flexible* rectangle are assigned variables x_1 and x_2 and the position of the origin of fixed-instance of design in Figure 3(a) is assigned x_3. Hence the position of left edge of the *fixed* rectangle can be computed as $x_3 + 100$.

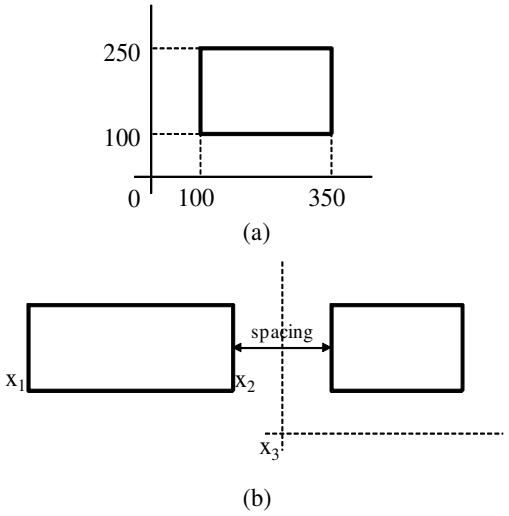


Fig. 3. Fixed-instance formulation example: (a) Design to be instantiated (b) Design with a *flexible rectangle* and a *fixed* rectangle from a fixed-instance

Consequently, the position of any rectangle edge can be written in the form $x_k + c_k$, where c_k is 0 for $1 \leq k \leq 2n$. As stated in Section II, each constraint that is generated is applied between

two rectangle edges. Hence any constraint can be written as a difference constraint even in the presence of fixed-instances as shown in equation 3. For example, if the spacing rule between the two rectangles in Figure 3 is 200, then the actual spacing rule can be computed as shown in equation 4.

$$x_i - x_j \geq C, i \neq j \tag{3}$$

$$(x_3 + 100) - x_2 \geq 200$$
$$x_3 - x_2 \geq 100 \tag{4}$$

Each difference constraint of the type shown in equation 3 arising from the line-sweep constitutes a row in equation 2.

B. spCompact

In order to minimize area and also preserve the structure of the layout, three things are important. First, it is important to keep clusters of rectangles together, i.e., clusters of packed rectangle edges should try to move as a unit and not independently. Second, the length of long connecting lines need to be reduced. Third and finally, the area of the layout must be reduced. Consequently, spCompact must penalize disturbance of relative positions of edges within a cluster, not penalize reduction of spacings of rectangles, and minimize the bounding box.

It can be observed that if a sufficiently large number of pairs of clustered edges are kept relatively unperturbed at the end of compaction, then all the clustered edges together would remain relatively unperturbed. Using this observation as the basis, the set of constraints generated by the constraint generator is used as the guiding heuristic to build spCompact. This is done by observing the different types of constraints generated by the constraint generator and identifying the subset that would help preserve the layout structure. The generated constraints broadly fall into six types of constraints, namely, width, spacing, clearance, overlap, extension, and connectivity constraints. It is often the case in compaction that a cluster of rectangle edges generates more constraints than sparsely spaced edges, predominantly due to the extension, overlap and connectivity constraints that have to be enforced. Consequently, if the pairs of edges constrained by overlap, extension, and connectivity constraints can be kept relatively unperturbed through penalty functions, this would achieve the objective to keep clusters of rectangles together. This is achieved in spCompact using the *Geometric Closeness* metric proposed in [10].

From equation 3, any overlap, extension or connectivity constraint can be represented as a difference constraint. *Geometric Closeness* metric keeps two edges relatively unperturbed by penalizing any change to the difference in positions, but allowing them to be translated without any penalty. Since a change in the difference between the two variables represented by x_i and x_j has to be penalized, equation 5 has to be added to spCompact.

$$|(x_i - x_j) - (x_i^0 - x_j^0)| \tag{5}$$

This is linearized similar to [10] by introducing two new variables R_k and L_k as shown in equation 6.

$$R_k \geq x_i - x_i^0$$
$$R_k \geq x_j - x_j^0$$
$$L_k \leq x_i - x_i^0$$
$$L_k \leq x_j - x_j^0 \tag{6}$$

The set of equations 6 is added to the set of constraints and minimize $(R_k - L_k)$ is added to the objective function. The similarity with *Geometric Closeness* is only in the metric used in the objective function. Constraints are used to construct the objective function in this work whereas the layout metal and space tiles are used to build the objective function in [10]. The linearization constraints in equation 6 are all difference type, since x_i^0 and x_j^0 are constants, unlike those in [10] which are linear.

Since reducing the layout area is the desired objective and not reducing the layout white space, the spacing and clearance constraints are not used in the penalty functions. It is important to note that the objective of minimizing area does not translate directly to minimizing the spacing between rectangles. For example, if a layout has achieved its minimum possible width during x compaction, minimizing white space can make the layout rectangles larger to reduce white space, thereby destroying layout structure. Hence, in order to minimize the area, spCompact minimizes the bounding box.

The bounding box is minimized as follows. Two variables x_l and x_r are introduced to compute the bounding box of the layout. x_l, the left boundary variable, is constrained to be $\leq$ the left edge of all layout rectangles. Similarly, x_r, the right boundary variable, is constrained to be $\geq$ the right edge of all layout rectangles. Equation 7 is added to the objective function to be minimized where $C_b \in N$ is a cost parameter.

$$C_b \cdot (x_r - x_l) \tag{7}$$

If C_{cnst} is the set of constraints used to build the objective function and R_k and L_k are linearization variables introduced for each constraint as shown in equation 6, then spCompact can be written as shown in equation 8.

$$minimize : C_b \cdot (x_r - x_l) + \sum_{C_{cnst}} (R_k - L_k) \tag{8}$$

The two components of spCompact actually oppose each other. While the first part tries to reduce the layout size, the second tries to retain the layout. So, higher the cost, C_b, more will be the weight to reduce area.

C. Estimating minimum area

In order to find out the effectiveness of spCompact in reducing area, it is desirable to get a lower bound on the bounding box of the layout. This section describes how to compute a lower bound on the bounding box for the layout.

From Equation 3, it can be seen that any constraint can be represented as a difference constraint in fixed-instance compaction. This suggests using a compaction technique using longest paths [9]. A directed constraint graph G is built with $2n + m + 2$ vertexes, corresponding to the $2n + m$ variables $x_i, 1 \leq i \leq 2n + m$ and the two boundary variables x_l and x_r. All the layout constraints that were generated are added as weighted edges. For example, the equation 3 would result in

a directed edge, $e(v_j, v_i)$, with a weight C from vertex v_j to v_i corresponding to the variables x_j and x_i respectively. Hence the longest path distance between v_l and v_r in G, corresponding to the boundary variables x_l and x_r respectively, gives the lower bound on the width of the bounding box.

The bound on the bounding box area is computed as follows assuming compaction is done along x dimension first, followed by compaction in y. Prior to compacting along x, the bounding box width W_x is estimated. Compaction is performed along x and the layout is updated. Similarly, prior to compacting along y, the bounding box height H_y is estimated. The lower bound on area is then computed as $W_x \cdot H_y$. While a true 2D technique can produce a lower bound on the area, in practice, since two successive 1D runs are performed, this gives a good estimate of the lower bound on the layout bounding box area.

IV. Experimental Results

A layout legalization engine has been written in C++. The engine takes in an input layout with DRVs and corrects them automatically. The legalizer can be configured towards different objectives such as wire length minimization (WLM)[9], minimum perturbation (minPert) [7] and the new objective spCompact presented in this paper. This section presents the results of experiments carried out with the different objective functions during legalization.

Designs from an industrial analog library are used in the experiments. These include comparator, voltage reference, voltage-controlled oscillator etc. in a 90nm technology. The actual design names are omitted here due to confidentiality reasons. Many of the designs have DRVs. The legalizer is run on each design with three different objectives. The first objective is traditional compaction with wire length minimization. The second objective is to legalize the layout while minimizing the perturbation in the layout. The third objective is the spCompact objective proposed in this paper. The experiments are run on a 2.2GHz Linux machine with 8GB RAM.

Table I presents a comparison of legalization with the three objectives. Column 2 presents the number of DRVs in the input layouts. For example, the input layout for Design6 has 70 design rule violations while Design9 has 0 violations. Both WLM and spCompact aim to legalize the layout while also attempting to minimize the output area. The minPert objective attempts to legalize without minimizing area. Columns 3 – 5 presents the ratio of input area to output area. Area ratio greater than 1 indicates a smaller layout after legalization. Column 3 reports the area ratio for WLM, column 4 for minPert and column 5 for spCompact.

Both WLM and spCompact result in smaller layout area after legalization. The minPert objective produces layouts that are of equal or slightly larger area after legalization. The best achievable area was computed for each one of these designs using the technique presented in Section III-C. Both WLM and spCompact consistently achieved the minimum area, whereas minPert did not. Since the area ratio for minPert is close to 1.0, it is as far away from the minimum as the area ratio of WLM and spCompact. For example, spCompact achieves 1.6x reduction in area for Design5, and 1.15x reduction in area on average. As can be observed from Column 5, spCompact can be used to recover significant area even if there are no input DRVs. For Design2, the layout had to be expanded to remove the DRVs. For Design10, the fixed instances determine the minimum area and hence it is the same for all objective functions.

Columns 6 – 8 in Table I present the number of DRVs in the output layout. Column 6 reports the number of DRVs with the WLM objective, column 7 reports DRVs for minPert and column 8 presents the number of DRVs for spCompact. Interestingly, WLM consistently reports more DRVs in the output compared to both minPert and spCompact. For example in Design2, WLM reduces DRVs from 16 to 2, while both minPert and spCompact clean up all DRVs. For Design7 WLM actually introduces 12 new DRVs while minPert and spCompact clean up all violations. This is because both minPert and spCompact attempt to retain layout structure thereby avoiding introduction of context-dependent DRVs that are hard to model accurately with linear constraints. An extreme case is presented in Design10 where WLM introduces 111 context dependent violations for a completely design rule clean input layout. Hence minimizing area alone is not sufficient, it is important to preserve the layout structure for effective DRV removal.

The results in Table I substantiates an important point. The spCompact objective presented in this paper combines the good properties of both WLM and minPert. Thus, it reduces area like WLM while also removing DRVs like minPert. Therefore, spCompact can be used to recover area for even DRV clean layouts. This is illustrated for Design3 and Design9.

Table II presents the runtime and problem size data for the three objectives. Column 2 reports the number of rectangles in the layouts. Columns 3 – 5 presents the number of variables normalized with respect to WLM. Column 4 reports that minPert involves more than 2x of the number of variables compared to WLM. Column 5 shows that spCompact introduces close to 22x the number of variables used by WLM on the average. This is because spCompact uses lot of constraints to guide the objective function. Each constraint linearization introduces new variables. Columns 6 – 8 reports the total runtime in seconds for WLM, minPert and spCompact respectively. As the results indicate, spCompact takes longer to legalize/compact layouts compared to WLM and minPert. It should be noted that the time taken to achieve closure for a layout design is significantly larger than the run-time of spCompact. As illustrated in Table I spCompact generally has much better quality of results in terms of both minimizing area (1.15x) and removing DRVs. Hence the run-time of spCompact, though larger than WLM and minPert, is still acceptable. Particularly, from Table II, it can be observed that for large designs, Designs 9 and 10, the runtime for spCompact is less than 2x of WLM.

V. Conclusions

In this paper, a compacting legalizer, spCompact, was presented that preserves layout structure while removing design rule violations in the layout and minimizing the layout area. The layout area is minimized through a bounding box approach as opposed to direct minimization of spacing and clearance constraints which can otherwise destroy the structure of the layout. It can consistently match the estimate of the minimum achievable area similar to *wire length minimization*, effectively remove

TABLE I

COMPARISON OF AREA AND DRVS FOR DIFFERENT OBJECTIVE FUNCTIONS

Design	*Input DRVs*	*Ratio of Input-Area to Output-Area*			*Output DRVs*		
		WLM	*minPert*	*spCompact*	*WLM*	*minPert*	*spCompact*
Design1	3	1.24	1.00	1.24	1	0	0
Design2	16	0.92	0.91	0.92	2	0	0
Design3	0	1.06	1.00	1.06	4	0	0
Design4	3	1.46	1.00	1.46	0	0	0
Design5	0	1.46	1.00	1.59	0	0	0
Design6	70	1.02	1.00	1.02	8	2	1
Design7	1	1.02	1.00	1.02	12	0	0
Design8	37	1.02	0.99	1.02	3	2	1
Design9	0	1.17	0.99	1.17	0	0	0
Design10	0	1.00	1.00	1.00	111	0	0

TABLE II

COMPARISON OF PROBLEM SIZE AND RUNTIME WITH DIFFERENT OBJECTIVES

Design	*Rectangles*	*Variables normalized wrt WLM*			*Runtime (s)*		
		WLM	*minPert*	*spCompact*	*WLM*	*minPert*	*spCompact*
Design1	1378	1	2.78	15.32	8	9	36
Design2	1538	1	2.70	13.41	6	10	146
Design3	1548	1	2.91	15.66	14	16	52
Design4	2189	1	2.76	15.83	8	7	10
Design5	4447	1	2.70	18.16	13	14	26
Design6	5143	1	2.78	23.36	72	79	354
Design7	5464	1	2.77	17.57	95	89	497
Design8	5846	1	2.87	34.34	114	130	1468
Design9	9662	1	2.79	23.59	89	84	156
Design10	35007	1	2.83	59.43	1108	1082	1739

design rule violations like *minimum perturbation*, and not introduce any new violations. In the future work, pruning the constraints used to construct the objective function by using the constraint graph is being explored. The reduction in area, 1.15x on average, and removal of design rule violations while preserving the layout structure is a significant aid to achieve design closure.

REFERENCES

[1] *Cadabra User Guide*, Synopsys Inc., 2007.

[2] *ProGenesis User Guide*, Prolific Inc., 2006.

[3] *IC Compiler User Guide*, Synopsys Inc., 2007.

[4] *Soc Encounter User Guide*, Cadence Design Systems, 2007.

[5] MOSIS, *TSMC Design Kits*, World Wide Web, http://www.mosis.com/products/fab/vendors/tsmc/tsmc-kits.html.

[6] *OpenAccess Release 2.2 Standard*, Silicon Integration Initiative Inc., 2005.

[7] F-L. Heng, Z. Chen, and G. Tellez, "A VLSI artwork legalization technique based on a new criterion of minimum layout perturbation," *Proc. Int. Symp. Physical Design*, pp. 116–121, Apr. 1997.

[8] L. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, USA, second edition, 1984.

[9] D. Boyer, "Symbolic layout compaction review," *Proc. IEEE/ACM Design Automation Conf.*, pp. 383–389, Jun. 1988.

[10] J. Zhu, F. Fang, and Q. Tang, "Calligrapher: A new layout-migration engine for hard intellectual property libraries," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 9, pp. 1347–1361, Sep. 2005.

[11] X. Yuan, K. McCullen, F-L. Heng, R. Walker, J. Hibbeler, R. Allen, and R. Narayan, "Technology migration technique for designs with strong RET-driven layout restrictions," *Proc. Int. Symp. Physical Design*, pp. 175–182, Apr. 2005.

[12] X. Tang and X. Yuan, "Technology migration techniques for simplified layouts with restrictive design rules," *Proc. Int. Conf. Computer Aided Design*, pp. 655–660, Nov. 2006.

[13] S. H. Batterywala, S. Bhattacharya, S. Rajagopalan, H. K. T. Ma, and N. V. Shenoy, "Cell swapping based migration methodology for analog and custom layouts," *Proc. Int. Symp. Quality Electronic Design*, pp. 450–455, Mar. 2008.

[14] J. Doenhardt and T. Lengauer, "Algorithmic aspects of one-dimensional layout compaction," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-6, no. 5, pp. 863–878, Sep. 1987.

[15] J-F. Lee, "A new framework of design rules for compaction of VLSI layouts," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 11, pp. 1195–1204, Nov. 1988.

[16] M. Riepe and K. Sakallah, "The edge-based design rule model revisited," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 3, no. 3, pp. 463–486, Jul. 1998.

Session 2C

Network on Chip

2009 22nd International Conference on VLSI Design

Improving Scalability and Per-core Performance in Multi-cores through Resource Sharing and Reconfiguration

Tameesh Suri and Aneesh Aggarwal
Department of Electrical and Computer Engineering
State University of New York at Binghamton
Binghamton, NY 13902
{tameesh, aneesh}@binghamton.edu

Abstract

Increasing the number of cores in a multi-core processor reduces per-core performance. On the other hand, providing more resources to each core limits the number of cores on a chip. In this paper, we propose a mechanism to improve the per-core performance while maintaining the scalability. In particular, we integrate a Reconfigurable Hardware Unit (RHU) in the resource-constrained cores to improve their performance. The RHU executes the frequently encountered instructions to increase the core's overall execution bandwidth, thus improving its performance. The RHU has low area overhead, and hence has minimal impact on scalability of the number of cores. To further limit the area overhead of this performance improving mechanism, generation of the reconfiguration bits for the RHUs of multiple cores is delegated to a single core. Our experiments show that the proposed architecture improves the per-core performance by an average of about 23% across a wide range of applications, while incurring a small per-core area overhead.

1. Introduction

In any technology generation, increasing the number of cores in multi-core processors requires reducing resources in each core, which is further exacerbated by the increasing die area requirement for peripheral hardware such as interconnects, snoopy logic, etc [20]. Fewer per-core resources degrades the performance of each thread of execution [12]. In this paper, we propose a mechanism that improves the per-core performance while maintaining the scalability of the number of cores. We build on prior work [30], where a core's performance is improved by integrating an off-the-critical path reconfigurable hardware unit (RHU) in its datapath. Speed-up is obtained by executing frequently executed instructions on the RHU. These instructions do not consume the core's resources, effectively increasing the per-core performance.

In this paper, we use the approach for a multi-core processor. We also extend the approach by including memory instructions, as it was a limiting factor in [30]. The reconfiguration bits for the RHU are generated at run-time and each core consists of a hardware/software co-design methodology to generate reconfiguration bits along with the RHU. Furthermore, we propose a innovative methodology of delegating the reconfiguration bits generation for multiple cores to a single core. The reconfiguration bits are arranged as reconfiguration instructions. We term the hardware used for reconfiguration instruction generation as RIG-hardware and the cores with RIG-hardware as RIG-cores. Separating the RHU- and RIG-cores limits the per-core area overhead, maintains the scalability, and reduces the opportunity cost of integrating other resources. The proposed architecture also better utilizes the RIG-hardware because if the RIG-hardware is included in each core, it will be idle for the majority of cycles; traces are formed once and executed many times. With this approach, there will be no performance impact of trace generation if the number of threads is smaller than the number of cores because the RHU-cores do not incur any overhead for generating the reconfiguration instructions. Providing the RIG-hardware in each core may forfeit this benefit. If the number of threads concurrently executing is more than the number of cores, then only the thread scheduled on the RIG-core may have some performance impact.

Our experiments show that the RHU-core performance improves by about 23% across a wide variety of applications. The performance of the RIG-core lowers by an average of only about 0.4% to achieve the performance gains in the RHU-cores. Our studies show that our approach incurs a small per-core area overhead.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.58

2 Proposed Multi-core Architecture

There are two main themes of the proposed architecture – reconfiguration to improve performance and division of cores into RIG-cores and RHU-cores to maintain scalability. The RHU is reconfigured through dynamically generated reconfiguration instructions, consisting of a 6-bit specialized operation code (opcode).

A RIG-core generates the reconfiguration instructions for RHUs of multiple RHU-cores. When a RHU-core detects a frequently executed trace of instructions, it requests a RIG-core for reconfiguration instructions. We do not provide the RIG-cores with a RHU, to somewhat equalize the per-core transistor budget. The number of RHU-cores served by a single RIG-core is a design choice. For instance, in a 16-core processor, each four-core cluster may include one RIG-core serving the remaining three RHU-cores.

2.1 RHU-Core and trace execution

The RHU-core organization is mostly the same as that in [30]; we do not discuss it in detail here for want of space. Furthermore, the execution of a trace is also the same as explained in [30]. The differences are the following: Trace buffer – which is used to generate reconfiguration instructions in [30] is now a part of the RIG-core datapath, as shown in figure 1. Also, once a trace is detected by the RHU-core, reconfiguration instruction generation request is sent to a RIG-core. When the reconfiguration instructions are received back from the RIG-core, they are stored at the end of the RHU-core thread's code section.

In our implementation, a maximum of one memory instruction can be placed in a specific column in each row. An address bus is extracted from that column in each row to output the memory address. For a store instruction in the RHU, the value to be stored is also extracted as a live-out from the same row in which the store instruction is present. The memory addresses output from the RHU are multiplexed into the existing load/store buffer (LSB). For this, LSB entries, equal to the number of RHU rows, are allocated for the trace memory instructions. The relative program order of stores and that of loads and stores is maintained, *i.e.* a store preceding other stores and loads is placed in a row prior to them, to maintain the correct order in the LSB. However, the relative program order between loads may not be maintained.

2.2 RIG-Core

Exclusive hardware mechanisms to generate reconfiguration instructions may have a large area overhead. In this paper, we propose a hardware/software co-design for reconfiguration instructions generation, where the reconfiguration bits are generated in hardware and then converted into reconfiguration instructions using an embedded software. Figure 1 shows the schematic diagram of the RIG-core datapath. The RIG-core fetches instructions for the

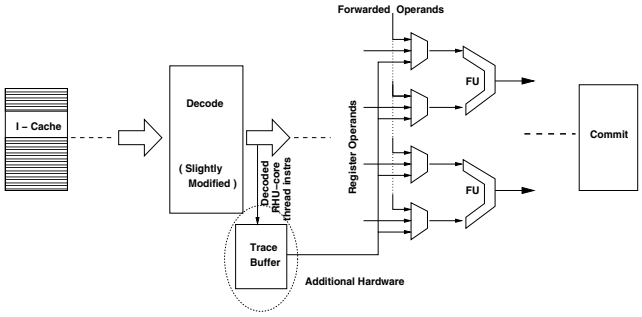


Figure 1. Schematic diagram of the Reconfiguration Instructions Generation core datapath

RHU-core thread using the existing fetch datapath. While fetching the RHU-core thread instructions, RIG-core stalls the fetch of the current thread running on it. The RIG-core thread is not context-switched out of the core, and its instructions already in the pipeline continue to execute. The RHU-core thread instructions are decoded and forwarded to the trace buffer [30], as shown in Figure 1.

Trace buffer hardware and operation is identical as used in [30] However, to optimize the hardware for reconfiguration bits generation, only one RHU-core thread is fetched and analyzed at a time.

In trace formation, we use an innovative method of forwarding instructions (FIs) to forward values across rows, in order to obtain larger traces for the RHUs. Trace sizes were limited in [30] due to unavailability of column or row in the RHU. FIs provide a cost-effective method to obtain larger traces than having more live-ins, live-outs or ALUs. As more live-ins are available in the top row and more live-outs are provided in the bottom-most row, FIs can be included to forward the values, and allow more flexibility in placing an instruction. FIs are also used to forward values across rows, for instance, if value produced in row one is required in row three, then a FI is inserted in row two to forward the value. A FI is treated as any regular RI.

The RHU reconfiguration bits are generated in two phases, as detailed in [30]. However, phase 1 and 2 are modified to incorporate FIs. We briefly go over the operation of reconfiguration bit generation. In phase 1, the dependencies are resolved by comparing the operands and destinations of instructions. In this phase, the rows and columns are also allocated to instructions depending on the availability of operands. FIs are inserted in phase 1 if operands of an instruction are available in different rows. If an instruction cannot be assigned a row, its entry is invalidated and phase 1 is halted. Phase 1 for each instruction requires 3 cycles.

Phase 2 starts after phase 1 and operates only on the instructions in the instruction buffer. Hence, the RIG-core thread instructions can be fetched and executed in parallel to phase 2. Phase 2 has a forward pass and a reverse pass through the instruction buffer. In the forward pass, the live-

outs are determined. If a live-out port is not available, then FIs are inserted, and if even that fails, then the forward pass is halted. The forward pass of phase 2 requires three cycles per instruction.

The reverse pass is used to remove any live-out violations in the forward pass and remains the same as in [30]. The reverse pass requires four cycles per instruction.

We also experimented with simpler trace formation techniques, but the experiments showed that this technique forms the largest traces, at the expense of higher trace formation latency.

Once phase 2 completes, the RIG-core thread fetch is stalled and embedded software [30](part of the operating system) instructions are fetched and executed to form the reconfiguration instructions. The embedded software reads each instruction buffer entry and generates the reconfiguration instructions. The embedded software instructions are fetched when the all in-flight RIG-core thread instructions have committed, and executed in-order. These instructions use only the speculative register file; they do not update the architectural register files. These instructions do not access the memory as well. Hence, the context of the RIG-core thread is intact in the core. The operation of the embedded software is detailed in [30], and remains the same. The reconfiguration instructions are forwarded to the RHU-core as they are formed.

3 Experimental Results

3.1 Experimental Setup

We experiment with a quad-core processor with three RHU-cores and one RIG-core. In this paper, we experiment with non-data-sharing threads scheduled on the cores. The hardware features and default parameters of each core are given in Table 1. The per-core resources are constrained to depict a core in a multi-core processor with large number of cores, and are similar to those in the current multi-core implementations. For instance, Hydra [24] has four 2-issue cores.

For benchmarks, we use a collection of Spec2K and MiBench [11] benchmarks. The statistics are collected for 200M instructions after skipping 1B instructions for Spec2K benchmarks and 50M instructions for the rest. For better legibility, we present the individual results of a representative set of nine benchmarks (`art`, `equake`, `mesa`, `mgrid`, `vpr`, `sha`, `susan`, `CRC32`, and `FFT`). We evaluate the performance of each benchmark on an RHU-core after averaging its performance in all its runs on an RHU-core. Similar approach is used to evaluate the performance of a benchmark on an RIG-core.

3.2 Area Results

We integrated a 36-ALU RHU (in particular a 4x9 RHU) with one SUN T1 OpenSource core [25] of an eight-core processor, to estimate the area overhead of RHU- and RIG-hardware. The design for the RHU and RIG-hardware was synthesized using Synopsys Design Compiler using a TSMC 90nm Standard cell library [31], and was placed and routed using Cadence SoC Encounter. After integrating the RHU, the core area increased by about 2.5%. The RIG-hardware adds about 3% to the core area. Figure 2 shows the die image of the RHU and the RIG cores. If the RHU and the RIG-hardware are included in every core, the per core area overhead would have increased by about 5.5%, instead of 3% and 2.5%. Furthermore, our experiments show that integrating the RHU and RIG-hardware in each core does not give any noticeable performance benefits over our approach.

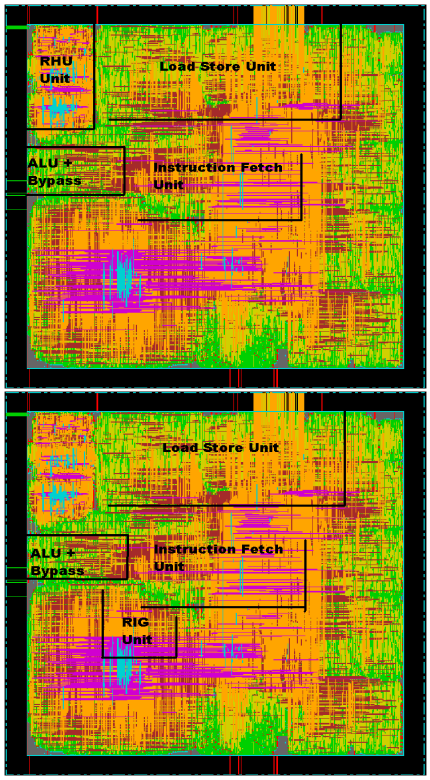


Figure 2. Die image of RHU Core and RIG Core

The per core resources of the SUN T1 Opensource core may not exactly match the per core parameters in Table 1. However, integration of the additional hardware into the SUN T1 core gives a reasonably accurate measure of the per-core area overhead of our approach in an eight-core processor. Previous studies [22, 24] suggest that a slight increase in the width of a core will easily increase the core area much more than the RHU. Hence, issue-width of a core is constrained while scaling the number of cores in a CMP. Figure 2 shows that the RHU is placed close to the functional and the load/store units as the RHU interacts with them, whereas the RIG-hardware is placed close to the fetch/decode and the functional units.

Parameter	Value	Parameter	Value
Fetch/Commit Width	4 instructions	*Instr. Window Size*	8 Int/8 Mem/16 FP instructions
ROB Size	96 instructions	*Issue Width*	1 Int/1 Mem/2 FP
Speculative Register File	48 Int/48 FP	*Int. Functional units*	1 ALU, 1 Mul/Div, 1 AGU
Load/store buffer	40 entries	*FP Functional Units*	2 ALU, 1 Mul/Div
Branch Predictor	gshare 4K entries	*L2 - cache* (shared by 4-cores)	unified 2M, 8-way assoc., 20 cycles
L1 - I-cache	16K, direct-mapped, 1 cycle latency	*L1 - D-cache*	16K, 4-way assoc. 64 bytes block, 1 cycle latency

Table 1. Experimental parameters for each core

3.3 Trace Results

We experiment with 36-ALUs, investigating 6x6, 5x7 and 4x9 RHUs. The 4x9 RHU performed the best with an overall average trace size of about 15 instructions. We observed that trace terminations due to column and row unavailability are almost balanced for the 4x9 RHU. We further observed that most of the original instructions are concentrated in the top two rows, whereas most of the FIs are concentrated in the bottom two rows. Overall, our experiments suggested that more columns and live-out ports are required in the top two rows.

To further increase the trace sizes, we also investigate an asymmetrical RHU structure – *AsymmRHU* – for the 36 ALUs. AsymmRHU is provided 11 columns in the first and second rows, six columns in the third row, five columns in the fourth row, and three columns in a fifth row. A fifth row is added to reduce the trace terminations due to row unavailability. However, the live-ins and live-outs per intermediate row are kept at two. All the ALUs in rows four and five are provided with live-outs. In AsymmRHU, each ALU output is still forwarded to four ALU-inputs in the next row.

Figure 3 compares the trace sizes, excluding the FIs, of AsymmRHU with the 4x9 RHU. Figure 3 shows that the trace sizes increase with AsymmRHU, with the overall average reaching almost 17 instructions. Our experiments also showed that the RIs formed a considerable fraction of the overall instructions executed in the applications, about 33% for 4x9 RHU and 37% for AsymmRHU.

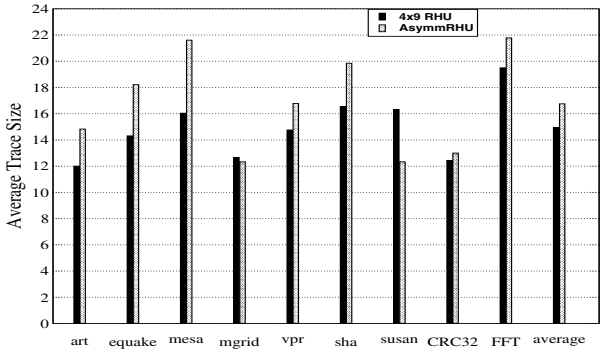


Figure 3. Average trace sizes for 4x9 RHU and AsymmRHU

3.4 RIG-core performance impact

In our experiments, we observed that the hardware takes an average of about 180 cycles, across the benchmarks, for generating the RHU reconfiguration bits. The embedded software takes an average of about 428 cycles for converting the reconfiguration bits into reconfiguration instructions. Our experiments showed that the average performance overhead in the RIG-core is less than 0.1%.

The low performance impact in the RIG-core is because of infrequent trace generations. Figure 4 presents the average number of cycles between successive trace generation requests and between successive RHU reconfigurations. On an average, only about 300 traces are generated, in the process of committing 500 million instructions, per benchmark. Hence, the average number of cycles between successive requests is high, about 2 million cycles as shown in Figure 4. The RIG-core, thus, receives only a small number of requests and spends minimal time in generating the reconfiguration instructions for the RHU-cores.

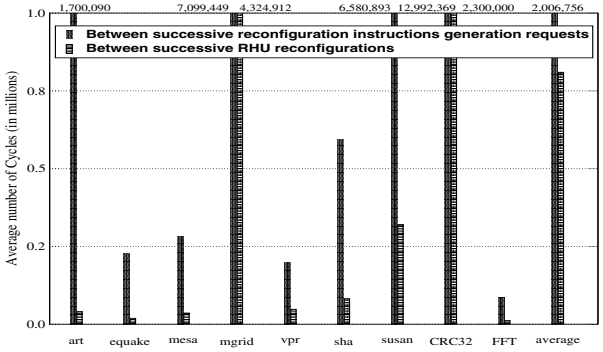


Figure 4. Average number of Trace generation requests and RHU reconfigurations

The low frequency of requests also results in negligible impact on the interconnect pressure from the trace-generation-related communication between RHU- and RIG-cores for reconfiguration instructions generation. We observed that an average total of only about 49 words are communicated per trace between the RIG-core and the RHU-cores.

3.5 RHU-core Performance Results

Figure 4 shows that an average of about 800,000 cycles elapse between successive RHU reconfigurations. Hence,

the overhead of executing the reconfiguration instructions to reconfigure the RHU for the first time is also negligible.

Next, we present the performance (IPC) improvement of RHU-cores, with 6x6 RHU, 4x9 RHU, and AsymmRHU, over the base core, in Figure 5. Figure 5 also shows the IPC speedup of cores with double-sized memory scheduler and with double-sized integer and memory scheduler. A double-sized scheduler doubles the issue queue size and issue width of the base case shown in Table 1. The number of functional units are accordingly increased. The maximum average IPC speedup of about 15% is obtained with AsymmRHU. The 4x9 RHU achieves about 12% IPC speedup.

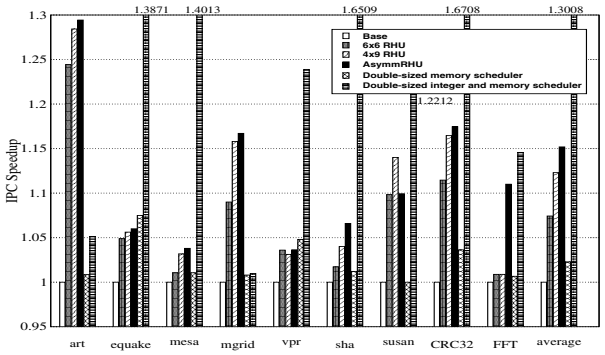


Figure 5. IPC speedup of RHU-cores compared to the base core configuration and that of cores with double-sized schedulers

Interestingly, our approach performs significantly better than all the double-sized scheduler configurations for `art` and `mgrid`. This is because the double-sized scheduler is still limited by other resources such as the fetch width, registers, etc., the pressure on which is somewhat relieved by the RHU. Additionally, when instructions are executing on the RHU, the effective issue queue size and issue width may more than double during that time. Our approach almost always performs better than the double-sized memory scheduler configuration. However, the average performance of AsymmRHU is about 14% lower than the double-sized integer and memory schedulers. This is because the RHU only speeds up a part of the loop. The rest of the application runs with the narrow width. *It is important to note that doubling the schedulers for better performance may have much higher impact on the scalability than our approach because of the increased scheduler size and additional functional units, forwarding paths and register file ports.* Our experiments showed that the average number of instructions issued per cycle in the integer and memory schedulers increased from about 0.48 in the base configuration to about 0.65 in AsymmRHU, but fell short of 0.76 observed in the double-sized integer and memory scheduler configuration.

3.6 Two RHUs per-core

To further improve the RHU-core performance, we experimented with two RHUs in each RHU-core. Two traces are formed from each innermost loop. The RIG-core datapath is not modified. We observed that the RIG-core performance impact increases to about 0.4% due to forming more traces. The first trace starts from the first instruction of the innermost loops. The second trace starts from an instruction that is approximately at the middle of the loop, provided that that instruction is not included in the first trace. This approach maximizes the distance between the traces, thus exploiting local ILP within each RHU and distant ILP in the two RHUs.

Figure 6 compares the speedup of two AsymmRHUs per core with that of a single AsymmRHU per core. Figure 6 shows that speedup increases for two AsymmRHU from about 15% to about 23%. We do not observe double performance improvement with two RHUs because two traces could not be formed in some loops, and there was not enough distant ILP to be exploited in some other cases. The area overhead of two RHUs is increases to about 5% in Figure 2.

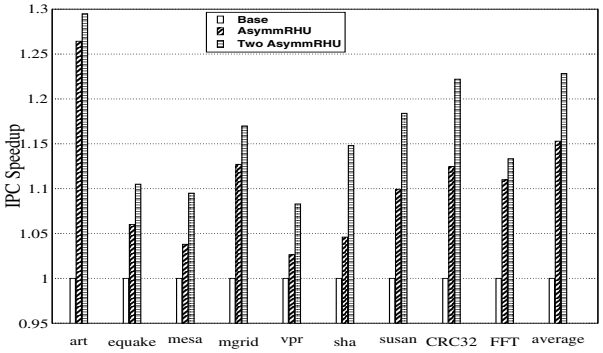


Figure 6. IPC speedup of one AsymmRHU per core and two AsymmRHUs per core

4 Related Work

Previous studies integrate FPGA modules with a processor to improve performance. PRISM [1], Spyder [16], Piperench [5], and Garp [4] use a loosely coupled FPGA as a co-processor. Similar co-processor based proposals [18] [23] [32] [27] [29] target application specific architectures.

Chimaera [13], PRISC [26], and OneChip [33] integrate the FPGA as a functional unit (RFU) in the processor datapath with direct access to the processor register file. The compiler statically generates the RFU instructions and FPGA reconfiguration bit-streams, which are used to dynamically reconfigure the FPGA. FPGAs have high area overhead, are considerably slower, and have higher energy consumption as compared to the ICs. Furthermore, FPGAs incur extensive overhead in generating and communicating the huge bit-streams required for reconfiguring them.

Other approaches execute aggregated instructions on custom functional units, for instance, [14] [15] [17] [19] fuse x86 micro-op pairs. These approaches target pairs of ALU instructions. Dynamic strands [28] extend beyond

pair-wise aggregation still targeting Integer ALU instructions. The authors in [2] [3] [9] fuse a dependence chain to form a special instruction, which is then executed on non-reconfigurable custom functional unit.

Clark et al. [8] propose a restrictive reconfigurable custom compute accelerator (CCA) that has a maximum of four inputs and two outputs, executing subgraphs of a small number of instructions terminating at branch and memory instructions. The authors acknowledge the performance limitations of terminating at branch and memory instructions in [6], a restriction not present in our approach. Hence, in [6], they also propose execution of more arbitrary acyclic sub graphs that cross branch boundaries and include memory instructions. This approach requires store-load collapsing within the sub-graph, and is targeted for single-issue in-order embedded processors.

Authors in [30] execute memory instructions as PIs to simply memory disambiguation. Trace generation hardware and RHU are included in each core, resulting in a large per-core area overhead.

Commit time trace formation has also been proposed to improve the fetch bandwidth and perform dynamic optimizations in superscalar processors [10]. However, the reconfiguration instruction generation in our approach is significantly different from the trace formations for superscalar processors.

5 Conclusion

In a multi-core processor, scalability of the number of cores and per-core performance conflict one another. The design choice is between having more cores with poor per-core performance and having good per-core performance but with fewer cores. In this paper, we explore a multi-core architecture that improves per-core performance, while maintaining the ability to scale the number of cores. In the architecture, the cores are divided into two categories – RHU-cores and RIG-cores. RHU-cores have integrated reconfigurable hardware unit (RHU) to improve their performance. The reconfiguration instructions for multiple RHU-cores are generated by a single RIG-core, thus reducing RIG-hardware overhead and improving its utilization. We propose innovative mechanisms to integrate the RHU in the core's datapath, to generate reconfiguration instructions using a hardware/software co-design, and to reconfigure the RHU. These mechanisms keep the area of the additional hardware requirement to a minimum, and have a small impact on the scalability of the number of cores. The proposed architecture improves the average per-core performance of RHU-cores by about 23%. The approach has a 0.4% impact on the RIG-core performance to achieve the performance gains in the RHU-cores.

References

[1] P. Athanas et al., "Processor reconfiguration through instruction-set metamorphosis," *IEEE Computer,26(3)*, 1995.

[2] A. Bracy et al., "Dataflow Mini-Graphs: Amplifying Superscalar Capacity and Bandwidth," *Proc. MICRO*, 2004.

[3] A. Bracy et al., "Serialization-Aware Mini-Graphs: Performance with Fewer Resources," *Proc. MICRO*, 2006.

[4] T. Callahan et al., "The garp architecture and c compiler," *IEEE Computer,33(4):62-69*, April 2000.

[5] Y. Chou et al., "Piperench implementation of the instruction path coprocessor" *Proc. MICRO*, 2000

[6] N. Clark et al. "An architecture framework for transparent instruction set customization in embedded processors," *Proc. ISCA*, 2005.

[7] N. Clark et al., "Processor acceleration through automated instruction-set customization" *Proc. MICRO*, 2003

[8] N. Clark et al. "Application Specific Processing on a General Purpose Core via Transparent Instruction Set Customization"*Proc. MICRO*, 2004

[9] M. L. Corliss et al., "DISE: A Programmable Macro Engine for Customizing Applications", *Proc. ISCA*, 2003

[10] B. Fahs et al., "Performance characterization of a hardware mechanism for dynamic optimization" *Proc. MICRO*, 2001

[11] M. R. Guthaus et al."MiBench: A free, commercially representative embedded benchmark suite", *Work. Workload Characterization*, 2001

[12] L. Hammond et al., "A Single-Chip Multiprocessor," *IEEE Computer*, Volume 30, No. 9. Sept. 1997.

[13] S. Hauck et al., "The chimaera reconfigurable functional unit," *Proc. FCCM*, 1997.

[14] S. Hu et al. "An Approach for Implementing Efficient Superscalar CISC Processors," *Proc. HPCA*, 2006.

[15] S. Hu and J. Smith, "Using Dynamic Binary Translation to Fuse Dependent Instructions," *Int. Symp. on CGO*, 2004.

[16] C. Iseli and E. Sanchez, "Spyder: a sure (superscalar and reconfigurable) processor," *Journal of Supercomputing, 9(3):231-252*, 1995.

[17] Intel Corporation, "Mobile Intel Pentium 4 M-Processor Datasheet," Jun. 2003. http://www.intel.com/design/mobile/datashts/250686.htm.

[18] J. A. Jacob and P. Chow, "Memory interfacing an instruction specification for reconfigurable processors," *Symp. FPGAs*, 1999.

[19] I. Kim and M. Lipasti, "Macro-op Scheduling: Relaxing Scheduling Loop Constraints," *Proc. MICRO*, 2003.

[20] R. Kumar et al., "Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads and Scaling," *Proc. ISCA*, 2005.

[21] C. Lee et al., "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems", *Proc. MICRO*, 1997

[22] J. Lotz et al., "A Quad-Issue Out-of-Order RISC CPU," *Proc. Int'l Solid-State Circuits Conf.*, 1996.

[23] T. Miyamori and K. Olukotun, "Remarc: Reconfigurable multimedia array co-processor," *IEICE Trans. on information and systems, E82-D(2):389-397*, 1999.

[24] K. Olukotun et al., "The Case for a Single-Chip Multiprocessor," *ASPLOS*, 1996.

[25] Sun Microsystems, Inc. "OpenSPARC T1 Micro Architecture Specification," *Sun Microsystems, Inc.*, 2006.

[26] R. Razdan and M. Smith, "A high-performance microarchitecture with hardware-programmable functional units," *Proc. MICRO*, 1994.

[27] C.R. Rupp et al., "The napa adaptive processing architecture," *Proc. FPGAs for computing machines*, 1998.

[28] P. Sassone and D. Wills, "Dynamic Strands: Collapsing Speculative Dependence Chains for Reducing Pipeline Communication," *Proc. MICRO*, 2004.

[29] H. Singh et al.,"Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE Trans. on Computers, 49(5): 465-481*, 2000.

[30] T. Suri and A. Aggarwal,"Scalable Multi-cores with Improved Per-core Performance using Off-the-critical Path Reconfigurable Hardware," *Proc. HiPC*, 2008.

[31] "TSMC 90nm Core Library - TCBN90GHP", *App. Note - Revision 1.2*, 2006

[32] S. Vassiliadis et al. "The molen polymorphic processor," *IEEE Trans. on Computers*, Vol. 53, Issue 11, 2004.

[33] R. Wittig and P. Chow, "Onechip: An fpga processor with reconfigurable logic," *Proc. FCCM*, 1996.

Forecasting-based Dynamic Virtual Channels Allocation for Power Optimization of Network-on-Chips

Amir-Mohammad Rahmani, Masoud Daneshtalab, Ali Afzali-Kusha, Saeed Safari, Masoud Pedram†

Nanoelectronics Center of Excellence
School of Electrical and Computer Engineering
University of Tehran
{am.rahmani, m.daneshtalab}@ece.ut.ac.ir,
{afzali, saeed}@ut.ac.ir

†Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089
pedram@usc.edu

Abstract

In this paper, we present a dynamic power management technique for optimizing the use of virtual channels in network on chips. The technique which is called dynamic virtual channels allocation (DVCA) makes use of the traffic conditions and past buffer utilization to dynamically forecast the number of virtual channels that should be active. In this technique, for low (high) traffic loads, a small (large) number of VCs are allocated to the corresponding input channel. This provides us with the ability to reduce the power consumption of the router while maintaining the data communication rate. To assess the efficacy of the proposed method, the network on chip has been simulated using several traffic profiles. The simulation results show that up to 35% reduction in the buffer power consumption and up to 20% savings in the overall router power consumption may be achieved. Finally, the area and power overheads of the technique are negligible.

1. Introduction

Reducing feature sizes into the nanoscale regime and the trend towards integrating more functionality onto a single chip led to the rise of the System-on-Chip (SoC) paradigm which could have area, power, and delay problems [1][2][3][4]. The architecture used for the data communication in these systems is one of the components strongly affecting the area, power, and delay as three critical design parameters. Networks on Chip (NoCs) were proposed as a solution for the SoC interconnect power and delay problem. Among different components of routers, buffers consume a large amount of dynamic power which increases rapidly as the packet flow throughput increases [5][6]. Increasing the buffer size improves the performance of the interconnection network significantly at the price of a higher power consumption and, hence, the buffer size should be optimized [6]. One of ways to reduce the buffer size is to use the wormhole routing [7]. The latency of data communication in NoCs is one of the key design parameters which should be minimized. One of the ways to minimize the latency is to use virtual channels (VCs) which provide virtual communication path between routers as the main elements for the data communication in NoCs. Virtual Channel (VC) [8] decouples buffer resources from transmission resources. This decoupling allows active message to pass blocked messages using the available network bandwidth that would otherwise be left idle. In addition to avoiding deadlock situations [9], virtual channels increase network throughput by up to 40% over a wormhole router without VCs and reduce the dependence of throughput on the depth of the network [8]. The use of VCs, which increases the communication throughput, increases the power consumption of NoCs. The power consumption is also a crucial parameter in NoCs which should be minimized. To optimize the power, one ought to employ power efficient designs for routers.

In this paper, a dynamic power management technique for reducing the power consumption of the NoCs with virtual channels is proposed. The technique optimizes the number of active VCs for the router input channel based on the traffic condition and past link utilization. The rest of the paper is organized as follows. Section 2 presents the switch structure in NoCs while Section 3 describes the proposed forecasting-based dynamic virtual channels allocation architecture while the simulation results are discussed in Section 4. Finally, Section 5 concludes the paper.

2. Switch Structure

In this work, we make use of a switch whose main structure is based on *RASoC* switch [10]. We have made some modifications to its buffering, routing and flow control parts and added support for Virtual Channels (VCs) based on [8] was needed. The switch contains two generic modules, namely, input channel and output channel parts. In this section, we describe the details of the switch.

2.1 Communication Model

The switch utilizes a handshake mechanism for its communication model. Switch communicates with its neighbor switches or cores by sending and receiving request and response messages. Each link includes two

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.87

unidirectional channels in opposite direction to each other used to transmit data, framing and flow control signals. In addition to n bits for the data, there are two bits used for the packet framing which are bop and eop. The bop (begin-of-packet) is set only at the packet header, and eop (end-of-packet) is set just in the last payload word, which is also the packet trailer. Therefore variable packet length was supported.

2.2 Switching

The switch uses the wormhole packet switching approach [11] where messages are sent by means of packets composed of flits. A flit (*fl*ow control un*it*) which is equal to the physical channel word (or phit – *ph*ysical un*it*) has *n*+2 bits. It is the smallest unit over which the flow control is performed.

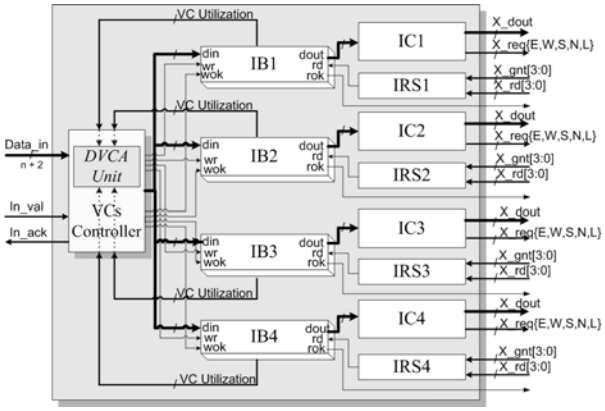


Fig. 1. Input channel architecture.

2.3 Routing and arbitration

The proposed switch supports different deterministic or adaptive routing algorithms such as XY [10], DyXY [12], and Odd-Even [13] used in the 2-D mesh topology. In addition, exhaustive round-robin [14] and priority-based [15] arbitration schemes are supported by the switch. Note that the switch supports locking mechanism required for the wormhole packet switching.

2.4 Flow control and VCs management

Since the handshaking mechanism is used for the communication, when a sender puts a data on the link, it activates the *val* (valid) signal. When the receiver receives the data, it activates the *ack* (acknowledge) signal. In our VC management approach, after the reception of each packet, *VC Controller* unit allocates one of the free VCs to this packet and locks this VC until the packet leave the VC.

2.5 Input channel and Output channel modules

The input channel module shown in Fig. 1 consists of four important units which are *VC Controller, IB* (Input Buffer), *IC* (Input Controller), and *IRS* (Input Read Switch). In this figure, four VCs are used for each input channel. The VC Controller exploits handshaking protocols for the flow control, allocation/deallocation of each VC to the input flow, and DVCA (Dynamic Virtual Channels Allocation) mechanism which will be in Section 3. The IB block is a p × (n+2)-bit FIFO buffer which is responsible to store the flits of the incoming packets while they can not be forwarded to an output channel. The number of the VCs is p. The IC block performs the routing function while IRS is responsible to deliver the read signal form the output channel to its connected IB.

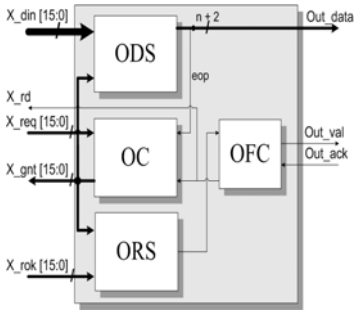


Fig. 2. Output channel architecture.

The output channel architecture of the proposed switch which is similar to RASoC switch [10] is depicted Fig. 2. It is composed of four blocks which are *OC* (Output Controller), *ODS* (Output Data Switch), *ORS* (Output Read Switch), and *OFC* (Output Flow Controller). The OC block runs the arbitration algorithm to select one of the requests sent by the VCs. Then, it activates the grant line of the selected request which induces proper switching of the ODS and ORS blocks. They connect the x_din and x_rok signals of the selected input channel to the external output channel interface. ODS and ORS blocks respectively connect the x_din and x_rok signals of the selected input channel to the external output channel interface However, before being connected to out_val, the x_rok signal pass through the OFC block.

3. Forecasting-based Dynamic Virtual Channels Allocation (DVCA) Architecture

Buffers are the single largest power consumer for a typical switch in an on-chip network [17] such as the Alpha 21364 router [17] the input buffers contribute 46-61% of the total power in a switch. This provides the motivation for us to analyze and optimize the power consumption of VCs without degradation in performance in the context of input channel switches for an on-chip network. In this work, we propose to use a dynamic power management (DPM) technique to dynamically determine the number of active VCs. The DPM technique has the objective of minimizing the power consumption with a minimal impact on the throughput. It also provides us with flexibility of adjusting the trade-off between the power and performance. The technique is based on a

distributed forecasting-based policy, where each router input port predicts its future communication workload and required virtual channels based on the analysis of the prior traffic.

3.1 Communication Traffic Characterization

The characteristics of the communication traffic in an input channel may be captured using several potential network parameters. Different traffic parameters such as link utilization, input buffer utilization, and input buffer age have been proposed for simple input channels (without VCs) [18]. None of these parameters (indicators) alone may correctly represent the communication traffic in VC-based input channels. Thus, we need a combination of these parameters to explore suitable indicators that are useful for predicting the network load in VC-based input channels. In this work, we use the link utilization and the virtual channel utilization as explained here.

This *Link Utilization* parameter is defined as

$$LU = \frac{\sum_{t=1}^{H} A(t)}{N}, \quad 0 \le LU \le 1 \qquad (1)$$

where $A(t)$ is equal to 1, if the traffic passes through the link i in cycle t and 0 otherwise, and N is the number of clock cycles, which is sampled within a history window with the size of H defined in clock cycles. The link utilization is a direct measure of the traffic workload.

Regarding this parameter, it should be noted that when the network is lightly loaded or highly congested, the link utilization is low [18]. At low traffic loads, the link utilization is low due to the fact that the flit arrival rate is low. When the network traffic increases, the flit arrival rate between the adjacent routers and the link utilization of each link increases. When the network traffic approaches the congestion point, the number of free buffer spaces in the upstream router will become limited. This causes the link utilization to start to diminish. This observation reveals that the link utilization alone will not be sufficient for assessing the network traffic. The forecasting-based DVCA policy, therefore, requires more information for making a right decision. In this work, we use the utilization of each virtual channel to complement the link utilization indicator for the proposed forecasting policy.

As mentioned earlier, after receiving each packet the VC Controller allocates one available VC to the corresponding received packet and locks the VC (L = 1). When the packet completely leaves the router, the controller releases the VC (L = 0). The virtual channel utilization tracks how many locks of VC in the router input channel occurs.

Let us denote the number of cycles that each packet uses a VC if it is sent without interrupt by s. $L(s)$ is set when the corresponding VC is occupied during s cycles, n is the number of VC per input channel and H is the window size. We denote the virtual channel utilization by VCU and define it as the lock rate of each VC through H cycles obtained from

$$VCU = \frac{\sum_{s=1}^{H} L(s)}{H}, \quad 0 \le VCU \le 1 \qquad (2)$$

Also, the overall VCU, denoted by *OVCU*, is defined as the sum of VCUs of each input channel obtained from

$$OVCU = \frac{\sum_{i=1}^{n} VCU}{n}, \quad 0 \le OVCU \le 1 \qquad (3)$$

Table 1 shows a simple example of calculating the virtual channel utilization. In this example, H is equal to 5s, the number of virtual channels per input channel is four (n = 4), the numbers in the *VCx* columns are the packet number that uses a given VC at cycle i, and the numbers in *Lx* columns show the locking status of *VCx* at cycle i. The VCU of each VC in the window is given in the last shaded row. For this input channel, the OVCU is equal to 15/20 or 3/4 and LU is equal to 7/20.

Table 1. An example of calculating VCU

cycle	*VC1*	*VC2*	*VC3*	*VC4*	*L1*	*L2*	*L3*	*L4*
1	1	2	-	-	1	1	0	0
2	1	3	4	-	1	1	1	0
3	1	3	5	-	1	1	1	0
4	6	3	5	7	1	1	1	1
5	-	3	5	7	0	1	1	1
Total	2	2	2	1	**4**	**5**	**4**	**2**

We use the link utilization as the primary traffic indicator, while the virtual channel utilization is used as a litmus test for detecting the network congestion. Next, we will show the usage of these indicators for DVCA policy.

3.2 Forecasting-based DVCA Policy

In the proposed technique, the DVCA unit uses the LU and OVCU parameters for measuring the past communication traffic. Based on this, the communication traffic of the next period, the number of required active VCs is adjusted. To reduce the area and power overheads of the proposed unit, we should simplify the forecasting equation. For this, we combine the two measures using a simple weighted equation given by

$$CT = LU + W \times (OVCU - OVCU_{\min}),$$
$$0 \le W \le 1, 0 \le CT \le 1 \qquad (4)$$

where W is forecasting weight, CT is the communication traffic parameter, $OVCU_{\min}$ is the sum of all $VCU_{\min}$ for each input channel in each history interval, and $VCU_{\min}$ is the smallest possible lock rate for each VC. $VCU_{\min}$ occurs when there are no stalls for the packets to leave the corresponding VC and, hence, $OVCU_{\min}$ is equal to LU. In this equation, we set W to 0.5 which simplify Eq. (4) to a straightforward average equation (because $OVCU_{min} = LU$). As this equation implies, the communication traffic is a function of LU and the network load. The network load is proportional to the

extra locks of VCs multiplied by the coefficient of *W*. The latter will be added to the link utilization when the congestion happens in VCs.

To make the forecasting formula reliable, we use an exponential smoothing function. This is simple and popular forecasting formula which is used in programming and inventories control science and defined as [19]:

$$CT_{predict} = CT_{past} + \alpha \times (CT_{actual} - CT_{past}) \quad \text{or}$$
$$CT_{predict} = \alpha \times CT_{actual} + (1 - \alpha) \times CT_{past} \quad (5)$$

where α is the forecasting weight $CT_{predict}$ is the predicted communication traffic, CT_{actual} is the actual communication traffic in the current history period, and CT_{past} is the predicted communication traffic in the previous history period (H). The accuracy of the prediction is a strong function of α and hence its value should be selected such that the error may be minimized.

The network traffic profile has short-term and long-term fluctuations. The proposed technique in this work filters out the short-term traffic fluctuations adapting the number of active VCs based on long-term traffic transitions. Based on the prediction, the controller decides to increase, decrease, or keep the same the number of active VCs. The pseudo-code of our proposed Forecasting-based DVCA policy for the case of four virtual channels per input channel is shown in Algorithm 1.

3.3 Hardware Implementation

Figure 3 shows the hardware realization of the proposed forecasting-based DVCA policy which relies on the local link and buffer information. Since the communication overhead of relying on global information is avoided, a simple hardware implementation may be used. To measure the link utilization (LU), a counter at each input port gathers the total number of packets that are passed from the link in each history interval. Similarly, there is a counter for each VC calculating the number of occurred locks (VCU). For computing the OVCU, an adder block is used to sum up the VCUs in each input channel. The *CT Calculator* carries out CT using Eq. (4). The *Forecasting Unit* uses the calculated CT and previous predicted CT from the previous interval (CT_{past}) to predict the new CT ($CT_{predict}$) for the next H period intervals. A register stores the $CT_{predict}$ to be used as the CT_{past} in the next interval.

To reduce the hardware overhead, we set α to 12/16, which is very close to the optimal values for this parameter for the traffic profiles used in this work. We implemented the division and multiplication operations by right and left shifts, respectively.

The Decision Logic unit determines the number of required active virtual channels (Required_VCs) using $CT_{predict}$, CT_{past} and required number of the virtual channels for the previous *H* period. Based on the Required_VCs value, the number of virtual channels in each input channel may be changed. The change in the number of active VCs is performed via clock gating technique. Finally, note that we simplify the division and multiplication operations by setting *H* and *H*×*n* values as power-of-two. For example, we set both *n* and *H* to 4.

Algorithm 1 Forecasting-based DVCA

$CT_{actual} = LU + (W \times (OVCU - LU))$
$CT_{predict} = CT_{past} + \alpha \times (CT_{actual} - CT_{past})$
if ($CT_{predict} > CT_{past}$) **then**
  if (*required_VCs* = 1 **and** $CT_{predict} > (\frac{H \times (1) - 1}{H \times n})$) **then**
    $required\_VCs = required\_VCs + 1$
  else if (*required_VCs* = 2 **and** $CT_{predict} > (\frac{H \times (2) - 1}{H \times n})$) **then**
    $required\_VCs = required\_VCs + 1$
    • • •
    • • •
  else if (*required_VCs* = *n* -1 **and** $CT_{predict} > \frac{H \times (n-1) - 1}{H \times n}$)**then**
    $required\_VCs = required\_VCs + 1$
  end if
else if ($CT_{predict} < CT_{past}$) **then**
  if (*required_VCs* = *n* **and** $CT_{predict} < \frac{n-1}{n}$) **then**
    $required\_VCs = required\_VCs - 1$
  else if (*required_VCs* = *n-1* **and** $CT_{predict} < \frac{n-2}{n}$) **then**
    $required\_VCs = required\_VCs - 1$
    • • •
    • • •
  else if (*required_VCs* = *2* **and** $CT_{predict} > \frac{1}{n}$) **then**
    $required\_VCs = required\_VCs - 1$
  end if
end if
$CT_{past} = CT_{predict}$

4. Results and Discussion

To assess the efficiency of the proposed technique to, we have compared NoCs with the forecasting-based DVCA and conventional virtual channel controllers. The comparison is performed in terms of power and latency for different traffic profiles. We used VHDL to develop six switches based the XY routing algorithm with 2, 4 and 8 virtual channels based on the conventional (non-DVCA) and DVCA and. They are labeled as XY-2VCs, XY-2VCs-DVCA, XY-4VCs, XY-4VCs-DVCA, XY-8VCs and XY-8VCs-DVCA, respectively. The simulations were carried out for a 5×5 mesh NoC using these six switch models. Also, we set *W*, α, and *H* to 1/2, 12/16 (75%), and 4, respectively. The performance of the network is evaluated using latency curves as a function of the packet injection rate (i.e., the number of packets injected to the network per cycle). The packet latency is defined as the duration from the time when the first flit is created at the source core to the time when the last flit is

delivered to the destination core. For each simulation, the packet latencies are averaged over 250,000 packets. Latencies are not collected for the first 30,000 cycles to allow the network to stabilize. It is assumed that the packets have a fixed length of five flits, the buffer size of each virtual channel is five flits, and the data width is set to 32 bits. To perform simulations, we used *uniform* and *transpose* traffic patterns [20]. In the uniform traffic pattern, a core sends a packet to any other cores with an equal probability while in the transpose traffic pattern, a core at the position (i, j) only send packets to the core at the position $(5 - j, 5 - i)$.

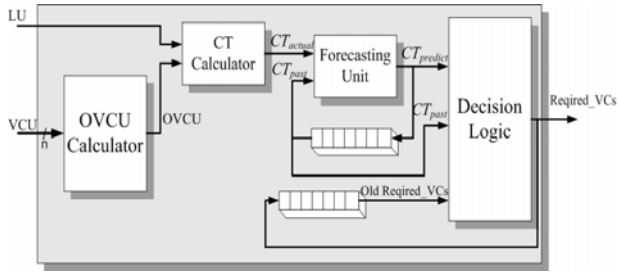


Fig.3. The hardware implementation of the forecasting-based DVCA policy.

The NoC performances for the two virtual channel management schemes under uniform and transpose traffic are given in Fig. 4 and Fig. 5. As observed from the figure, each pair of DVCA and non-DVCA with 2, 4, and 8 VCs have almost the same performance at low traffic loads. As the traffic load increases, the packet latency rises dramatically due to the network congestion. The results show that the conventional VC controller performs slightly better than the DVCA VC controller. This originates mainly from the prediction error. The power consumptions of the routers which are computed by Synopsis Power Compiler for a 0.13µm standard CMOS technology are presented in Fig. 6 and Fig. 7. As seen from the figure, the average power consumptions of the switches with the DVCA VC of the switches with the DVCA VC controller are considerably lower than the corresponding conventional switches at low traffic loads where some of the VCs may be clock-gated for saving the power. As the traffic load increases the difference between the average power consumptions of the switches with the same number of the VCs decreases till they eventually become almost the same at the congestion injection rate. The power saving is achieved at the price of slightly higher latency for the NoC with the DVCA VC controller. As we summed up the power consumption at each injection rate the power dissipation is reduced up to 35% in the buffer power consumption and up to 20% savings in the overall router power consumption. Finally, to determine the area and power overheads of the proposed technique, we synthesized the DVCA and conventional switches using Synopsys Design Compiler. Note that the controller is not on the critical path of the router and, hence, its delay overhead does not need to be considered. The synthesis results which are obtained for a 0.13µm standard CMOS technology are given in Table 2 and Table 3. The figures given in these tables reveal, the area and power overheads of the proposed forecast-based DVCA VC controller are negligible. Note that in estimation of power overhead, both dynamic and leakage power was considered in high traffic loads (worst-case).

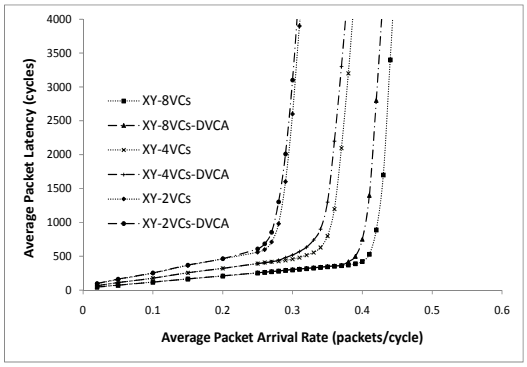


Fig. 4. Average latency under transpose traffic.

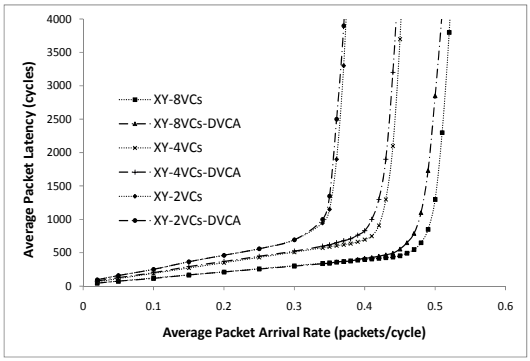


Fig. 5. Average latency under uniform traffic.

5. Conclusion

In this paper, we introduced a forecasting-based dynamic power management technique for controlling the number of active virtual channels (VCs). The approach makes use of the link and VC utilizations in predicting the communication traffic. Based on the predicted traffic, the number of the active virtual channels may be increased, decreased, or remain the same. The clock-gating power management technique is used to activate/deactivate the VCs. To determine the efficacy of the proposed technique NoCs with conventional and DVCA VC controller for 2, 4, and 8 VCs were simulated. The simulation results which performed for the uniform and transpose traffic profiles showed considerable power savings with a minimum impact on the latency for the proposed technique. The technique was implemented using a simple hardware to make the power and hardware overheads very small.

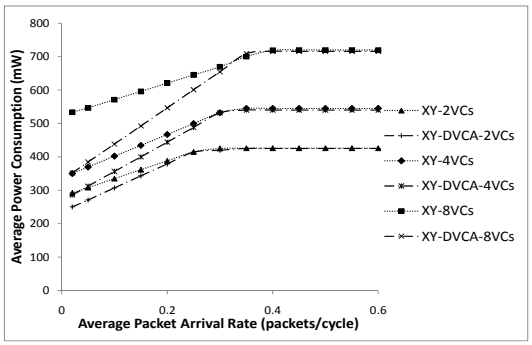


Fig. 6. Average power consumption under transpose traffic.

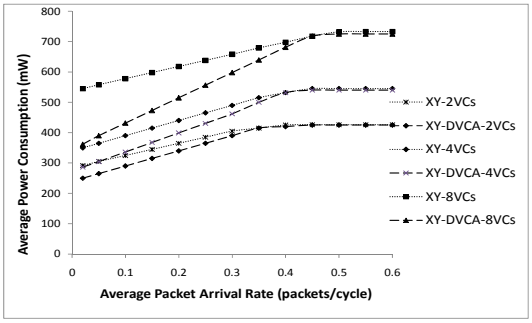


Fig. 7. Average power consumption under uniform traffic.

Table 2. Area overhead of the DVCA unit

Component	Area (μm^2)	Overhead (%)
DVCA Unit for 2 VCs	9296.81	2.78
DVCA Unit for 4 VCs	11918.68	1.99
DVCA Unit for 8 VCs	19863.11	1.09

Table 3. Power overhead of the DVCA unit

Component	Power (mW)	Overhead (%)
DVCA Unit for 2 VCs	4.86	1.7
DVCA Unit for 4 VCs	5.27	1.23
DVCA Unit for 8 VCs	8.55	0.89

Acknowledgement

The authors wish to acknowledge the financial support by Iran Telecommunication Research Center (ITRC) during the course of this project.

References

[1] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, pp. 70–78, January, 2002.

[2] W.J. Dally et al., "Route Packets, Not Wires: On-Chip Interconnection Networks," *in Proc. of the DAC Conference*, pp.684-689, 2001.

[3] S. Heo and K. Asanovic, "Replacing global wires with an onchip network: a power analysis," *in Proc. of the ISLPED Conference*, pp. 369-374, 2005.

[4] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in multi-core architectures: understanding mechanisms, overheads and scaling," *in Proc. of ISCA Conference*, pp. 408-419, 2005.

[5] W. Hangsheng, L. S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," *in Proc. of the MICRO Conference*, pp. 105-116, 2003.

[6] T. T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," *in Proc. of DAC*, pp. 524-529, 2002.

[7] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, 26:62{76, Feb. 1993.

[8] W. J. Dally, "Virtual-channel flow control," *in Proc. of the ISCA*, pp. 60-68, 1990.

[9] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, pp. 547-553, 1987.

[10] C.A. Zeferino, M.E. Kreutz, A.A. Susin, "RASoC: A Router Soft-Core *for* Networks-on-Chip," *DATE*, Feb. 2004, pp. 198- 203.

[11] W. J. Dally and C. L. Seitz, "The torus routing chip," *Journal of Distributed Computing*, vol. 1(3), pp. 187-196, 1986.

[12] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY – a proximity congestion-aware deadlock-free dynamic routing method for network on chip," *in Proc. of the DAC Conference*, July 2006, pp. 849–852.

[13] G. M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Trans. on Parallel and Distributed Systems*, 11:729 – 738, July 2000.

[14] E.S .Shin, V.J. Mooney, G.F. Riley, "Round-robin Arbiter Design and Generation," *in Proc. of International Symposium on System Synthesis*, 2002, pp. 243-248.

[15] A. Bystrov, D. J. Kinniment, and A. Yakovlev, "Priority Arbiters," *in Proc. of the ASYNC Conference,* 2000, pp. 128-137.

[16] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36(5), pp. 547-553, 1987.

[17] S. Banerjee, and N. Dutt, "FIFO Power Optimization for OnChip Networks," *in Proc. of the 14th GLSVLSI Conference,* 2004, pp. 187-191.

[18] L. Shang, L. S. Peh, and Jha. N.K., "Dynamic Voltage Scaling with Links for Power Optimization of interconnection Networks," *in Proc. of the 19th HPCA Conference*, Feb. 2003, pp. 91-102.

[19] R .J. Tersine, *Principles of Inventory & Material Management*, Fourth Edition, Prentice Hall PTR, August 1994.

[20] M. Rezazad, H. Sarbazi-azad, "The Effect of Virtual Channel Organization on the Performance of Interconnection Networks," *in Proc. of the 19th IPDPS Conference*, April 2005, pp. 264-271.

Negative Exponential Distribution Traffic Pattern for Power/Performance Analysis of Network on Chips

Amir-Mohammad Rahmani, Iman Kamali, Pejman Lotfi-Kamran,
Ali Afzali-Kusha and Saeed Safari

Nanoelectronics Center of Excellence
School of Electrical and Computer Engineering, University of Tehran
am.rahmani@ece.ut.ac.ir, Iman_kam@aut.ac.ir, plotfi@computer.org, {afzali, saeed}@ut.ac.ir

ABSTRACT

In this paper, we propose a synthetic traffic model based on Negative Exponential Distribution (NED). This synthetic traffic profile is more similar to some statistical behavior of realistic traces obtained by running different applications on Network-on-chips that those of conventional synthetic traffic profiles. To assess usefulness of this traffic model, the average packet hops for the proposed traffic profile is compared with those of some synthetic and realistic traffic patterns obtained from running applications on NoCs. The results show that the NED traffic profile has more similarity with the realistic traffic profiles than those of conventional synthetic ones.

1. Introduction

To increase the computing power of single chip systems, several processors may be used. As the computing power increases, the communication speed should also increase to satisfy the data exchange requirements of the system. In these systems, conventional bus based communication architectures may not work and may be replaced by Network on Chip (NoC) ones [1][2]. Many research groups have devoted their efforts on different aspects of NoC's, including topology, routing algorithms and architectures, and core mapping (see, e.g., [1][2][3]). To assess the performance of the NoC options which exist in its vast design space, simulations with different traffic profiles should be used. These simulations are used to determine power and latency characteristics of a given NoC architecture. Traffic profiles may be classified as either synthetic or realistic. Synthetic traffics are abstract models of message passing in NoCs while realistic traffics are traces of real applications running on NoCs. In contrast to realistic traffics which are representative of a more specific class of application, synthetic traffics should cover a broad class of applications running on NoCs [4][5].

Designers frequently rely on synthetic traffic patterns such as Uniform random and Hotspot to evaluate their network design [5][6]. Some of other synthetic traffic profiles include Transpose, Bit-Complement, Bit-Reversal, and Self-Similar. Recently, several synthetic traffic profiles have been proposed. In [7], a traffic model for on-chip networks is proposed. This synthetic traffic is a good model for the multimedia applications running on NoC, but it may not be suitable for other applications.
In this paper, we propose a synthetic traffic profile based on negative exponential distribution. This traffic pattern can be used effectively to model the bursty traffic behavior at chip-level.

The rest of paper is organized as follows. Section 2 briefly introduces related works in this area and presents the motivation for presenting a new traffic profile for the NoC analysis. Section 3 presents the NED traffic model and its properties while the comparison results of this traffic profile with those of others are discussed in Section 4. Finally, the conclusion is given in Section 5.

2. Related Works

As discussed in the previous section, traffic profiles for the design and the analysis of NoCs can be categorized into realistic and synthetic groups. Next, we briefly discuss widely used traffic profiles.

2.1. Realistic Traffics

In some works, to evaluate power and delay, NoCs have been analyzed using under realistic traffic loads. For example in [8], the performance of the proposed technique has been evaluated using a GSM voice CODEC traffic profile. Other realistic traffic profiles used by researchers include SPLASH-2 [9], MediaBench [10], and SPEC [11] traces. It, however, should be noted that the traffic patterns generated by different modules in a NoC strongly depends on the application for which the

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.86

NoC is designed. Since the performance of the NoC is a function of the traffic profile, the most accurate way to assess the characteristics of the NoC would be to invoke the traffic profiles corresponding to the application. In many cases, the system is designed for multiple applications. In these cases, the traffic profiles corresponding to all the applications should be used during the NoC design and analysis. This can be time consuming even if all the applications are known beforehand. As another option, synthetic traffic profiles which can represent a class of application may be used. This suggests that the use of both realistic and synthetic traffic profiles forms a complete set for the evaluation of the techniques proposed for NoC systems. Next, synthetic traffic models are introduced and their features including the application that they are representative for are mentioned.

2.2. Synthetic Traffics

Different synthetic traffic patterns have been used for evaluating interconnection networks. Uniform, Transpose, Bit-Complement, Bit-Reversal, Hotspot [4], and Self-similar [7] are the most widely used traffic models for the analysis of power and delay in interconnection networks.

To describe the synthetic patterns, let each node (x, y) also be labeled with a number resulting from the concatenation of x and y. The binary representation of *xy* is $n_1 n_2 n_3 \ldots n_{m-2}\, n_{m-1} n_m$. Also, let $\bar{0} = 1$ and $\bar{1} = 0$.

- Uniform Traffic: Each node sends messages to other nodes with an equal probability (*i.e.*, destination nodes are chosen randomly using a uniform distribution).
- Hot-spot Traffic: Each node sends messages to other nodes with an equal probability except for a specific node (Hotspot) which receives messages with a greater probability. The percentage of messages that a Hotspot node receives beyond the usual nodes is indicated after the Hotspot name (e.g., Hotspot 10%).
- Transpose Traffic: Each node sends only to the destination given by $(n_{m/2} n_{(m/2)+1} \ldots n_m n_1 n_2 \ldots n_{(m/2)-1})$.
- Complement Traffic – Each node sends only to the destination given by $(\bar{n}_1\, \bar{n}_2\, \bar{n}_3 \ldots \bar{n}_{m-2} \bar{n}_{m-1} \bar{n}_m)$.
- Bit reversal Traffic: Each node sends only to the destination given by $(n_m\, n_{m-1}\, n_{m-2} \ldots n_3\, n_2\, n_1)$.

The uniform traffic model is a standard benchmark used in network routing studies. This model can be considered as the representative of well-balanced shared memory computations. In the Hotspot traffic pattern, one or more nodes are designated as the hot spot nodes, which receive Hotspot traffic in addition to the regular traffic. Therefore, the Hotspot node represents the very busy nodes. For example, in multiprocessors, Hotspot nodes could be the representative of computations in which there are critical sections or shared/replicated data. For the transpose traffic, two types of patterns are proposed. With the first transpose traffic pattern, a node (i, j) only sends messages to node $(n - j, n - i)$ where n is the network dimension (e.g., $n \times n$ in the mesh topology). This traffic pattern is very similar to the matrix-transpose [7] In the second transpose traffic pattern, a node (i, j) only sends messages to node (j, i). The bit-complement, reversal, and transpose traffics can model traces of applications related to numerical computations [4].

New synthetic traffic patterns may be inspired by analyzing the traffic patterns in a class of applications. As an example, in [7], a self-similarity concept is utilized to propose a new traffic pattern. The objective of the work was to introduce self-similarity as a fundamental property of the bursty traffic patterns flowing between the modules in typical MPEG-2 video applications. This property was inferred by examining the extracted traces when common video statistical tests were performed on the chip.

The above discussion shows that each synthetic traffic model is useful for certain classes of applications. Next, we discuss the motivation for another synthetic traffic profile which is representative for a broader class of NoC applications.

2.3. Motivation for NED

One of the stages in design of NoC systems is to map an application onto the cores existing on the chip. The mapping is an optimization problem with the objective of minimizing the total power consumption and delay of the communication on the chip. Several research works have been focused to the application mapping onto NoCs (see, e.g.,[12][13][14]). The power consumption as well as the delay for each data communication operation is minimized by lowering the number of hops and shortening the total physical distance between the source and destination cores. As a result of using these mapping algorithms, an application should be mapped among different cores such that the cores with a higher communication volume should be mapped as close to each other as possible. For these networks, the closer the nodes are, the more packets they send to each other. An example of this situation is shown in Fig. 1 which shows the number of packets sent from Node $X_{3,3}$ to other nodes. For this case a total number of 1,000,000 packets has been considered. Therefore, for a more accurate evaluation of these networks, a synthetic traffic pattern with this property should be used. Most of the above synthetic traffic profiles do not have this property, and hence, a new traffic profile

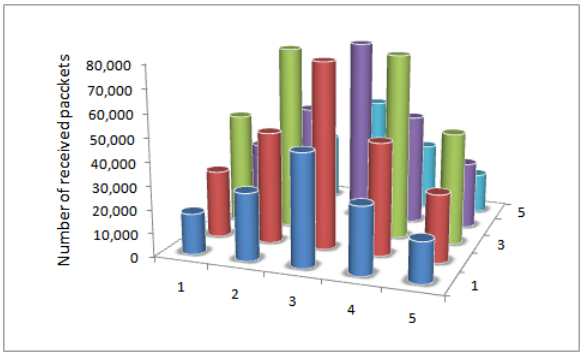


Fig. 1. Number of messages sent by Node (3, 3) to other nodes in a 5×5 mesh

3. NED Traffic Model

In a network, a source node S that is located in position (a, b) is referred to as $S = X^*_{a,b}$. In addition, other nodes that are placed in position (i, j) are referred to as $X_{i,j}$. Assuming a mesh topology for now, the distance matrix, **R**, for the source node is defined as

$$R_{n*n} = [r_{i,j}]$$

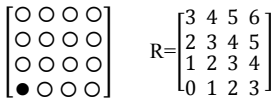
where $r_{i,j}$ is the distance (number of hops) between S and $X_{i,j}$ and is given by

$$r_{i,j} = | i + j - (a + b) | \qquad (1)$$

where is. Using $\mathbf{R}_{n*n}$, Number of Distance (NoD) matrix, $\mathbf{D}_{1,k}(n)$ is defined as

$$d_{1*k}(n) = \sum_{j=1}^{n} \sum_{i=1}^{n} \frac{r_{i,j=K}}{k} \quad , \quad k = | \text{Max } r_{i,j} | \qquad (2)$$

The j^{th} column of this matrix indicates the number of nodes in the network that have a distance of j from S. Fig. 2 shows R, and $D_{1,6}(4)$ for a 4×4 network with the source node of $X^*_{1,1}$.



$$R = \begin{bmatrix} 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 \\ 1 & 2 & 3 & 4 \\ 0 & 1 & 2 & 3 \end{bmatrix}$$

$$D_{1*6}(4) = [\,2\ 3\ 4\ 3\ 2\ 1\,] \ , \ k=6$$

Fig. 2. Distance matrix(R), Number of Distance matrix (D).

We are looking for a Probability Distribution Function (PDF) that computes the probability of sending from the source node to other nodes. The PDF should have the property that the probability decreases as the distance between the source and destination nodes increases. In addition, for the mesh topology, the PDF should be dependent on the source position. The reason is that the longest distance between a source and other nodes are dependent on the source position. Also, the number of nodes that have a specific distance from a source is different for different source positions in the mesh topology.

Fig. 3 shows the distance matrices for $S = X^*_{1,1}$ and $S = X^*_{3,3}$ in a 5×5 network. As shown, in Fig. 3(a), four nodes have a distance of one from the source node and the longest distance to source is 4. On the other hand, in Fig. 3(b), two nodes have a distance of one from the source node while the longest distance to the source is 8. It should have the general properties of probability distribution functions as well. If the P_r is the probability of sending a packet to a destination with the distance r from the source, and D is the set of all the distances from the source, then then $0<P_r<1$, $\sum_D P_{r=1}$.

$$\begin{bmatrix} 4 & 3 & 2 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 2 & 3 \\ 4 & 3 & 2 & 3 & 4 \end{bmatrix} \qquad \begin{bmatrix} 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 5 & 6 & 7 \\ 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

(a) (b)

Fig. 3. (a) Distance matrix for $S = X^*_{1,1}$,(b) Distance matrix for $S = X^*_{3,3}$

Note that a distribution function with the above specifications may be used for other topologies such as Torus, Hypercube, and 3D Torus. The reason for this is that this distribution function is only dependent on the distance between the source and destination.

In this work, we are looking for a probability distribution function in which the value of P_r decreases exponentially with increasing r. Denoting P_1 by P, we propose that the following probability function:

$$P_r = P^{m(r-1)} \times P \qquad (3a)$$

or

$$P_r = P^{(m(r-1))+1} \qquad (3b)$$

where m is a parameter between 0 and 1.

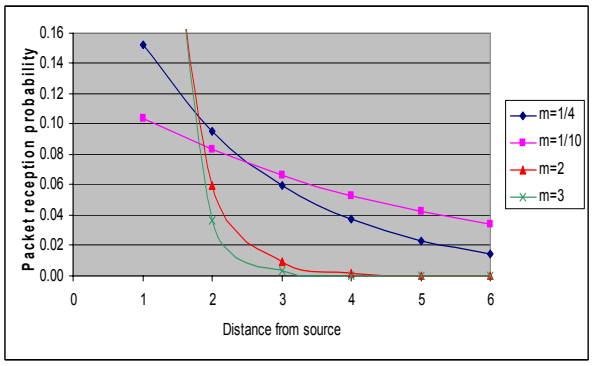


Fig. 4. Sending probability for m = 3, m = 2, m = 0.1, and m = 1/4.

Based on the central limit theorem [15], the sum of a large number (practically 30 or more) of independent and identically distributed random variables will have approximately a normal distribution. For a normal distribution function, the area under the function in the range μ±3δ is 97.65% of the total area. Based on this, we suggest that a suitable value for the parameter m be the one that results in a traffic pattern in which the length of the longest path is equal to μ+3δ. Thus, the value of parameter m is chosen to be $\mathrm{m} = \frac{1}{\mathrm{n}}$.

As an example, in Fig.4, the probability for sending a packet from $S = X^*_{1,1}$ to nodes with different distances in a 4×4 network for m = 3, 2 , 0.1 and 1/4 is shown. Using (4), we can write

$$\sum_{\mathrm{L}=1}^{\mathrm{k}}(\mathrm{r}_{1.\mathrm{L}}(\mathrm{n}) \times \mathrm{P}_{\mathrm{L}}) = 1 \qquad (4)$$

For the example shown in Fig. 2, we have

$$2P_1 + 3P_2 + 4P_3 + 3P_4 + 2P_5 + P_6 = 1 \qquad (5)$$

Putting $P_r = P \times P^{(r-1)m}$ in (5) leads to

$$2P + 3P^{m+1} + 4P^{2m+1} + 3P^{3m+1} + 2P^{4m+1} + P^{5m+1} = 1 \quad (6)$$

Let us assume m = ¼, and $P' = P^{1/4}$ or $P = P'^4$, then

$$2P'^4 + 3P'^5 + 4\,P'^6 + 3P'^7 + 2P'^8 + P'^9 = 1 \qquad (7)$$

The value of P' may be found by solving the above equation numerically. Notice that there is only one solution between 0 and 1 for the class of above equations. Therefore, the solution is unique. This is proved using the following theorem:

Theorem: For the equation given by

$$\mathrm{Y} = \mathrm{aP'^m} + \mathrm{bP'^{m+1}} + \mathrm{cP'^{m+2}} + \ldots + \mathrm{zP'^{m+n}} - 1 \qquad (8)$$

where the coefficients a to z are integer greater than or equal to 0 zero, there is a unique solution between 0 and 1.

Proof: The derivative of Y with respect to P' is given by

$$\mathrm{Y'} = \mathrm{maP'^{m-1}} + (\mathrm{m}+1)\mathrm{bP'^{m}} + (\mathrm{m}+2)\mathrm{cP'^{m+1}} + \ldots + (\mathrm{m}+\mathrm{n})\mathrm{zP'^{m+n-1}} \qquad (9)$$

where $\mathrm{Y'}$ is a continuous function on the set of real numbers. On the other hand, if $\mathrm{Y} = \mathrm{aP'^m} + \mathrm{bP'^{m+1}} + \mathrm{cP'^{m+2}} + \ldots + \mathrm{zP'^{m+n}} - 1 > 0$, $\mathrm{Y'}$ is also greater than zero. Therefore Y is strictly increasing in the range of 0 and 1. Since the value of Y for $\mathrm{Y} = \mathrm{aP'^m} + \mathrm{bP'^{m+1}} + \mathrm{cP'^{m+2}} + \ldots + \mathrm{zP'^{m+n}} - 1 = 0$ and 1 are –1 and $a + b + c + d + \ldots + z - 1 > 0$, respectively, and Y is continuous and strictly increasing, therefore, there is exactly one point in the range of 0 and 1 at which $Y = 0$.

Solving Equation (7) leads to P' = 0.6247. The probability distribution diagram for $S = X^*_{1,4}$ is shown in Fig. 5.

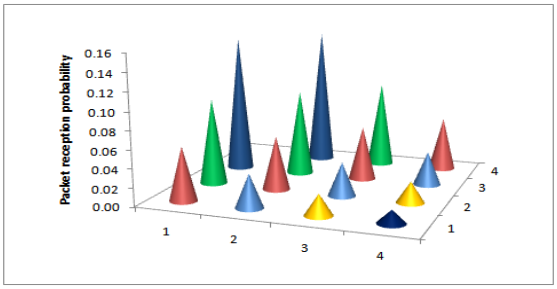


Fig. 5. Probability distribution diagram for $S = X^*_{1,4}$ for a 4×4 mesh network.

Note that the computation for a network with specific dimension is done only once. In addition, since due to the symmetry, there are nodes with the same distances, the computation should only be done for about one fourth of the nodes. This is shown in Fig. 6 for 4×4 and 5×5 meshes.

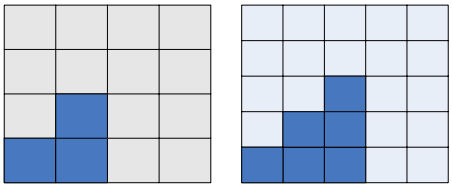


Fig. 6. The nodes that require specific computation for (a) 4×4 and (b) 5×5 meshes.

It should be noted that for ring, 2D, and 3D torus topologies this traffic has less computational complexity than that of 2D mesh. Fig. 7 shows ring, 2D torus, and 3D torus topologies. In these networks, the computations are done just for one node in a network with any dimension. To illustrate the point, let us define $n(d)$ for the source node S as the number of nodes with the Manhattan distance of d hops from S. Table 1 shows mathematical expressions of $n(d)$ for these network topologies. These expressions reveal that, regardless of the source position, the model of adjacent nodes is similar and is a function of d.

Table 1. $n(d)$ for major network topologies.

Topology	$n(d)$
Rings (1-dimensional)	2
2-dimensional torus	$4d$
3-dimensional torus	$4d + 2 + 8\sum_{a=1}^{d}(d-a)$

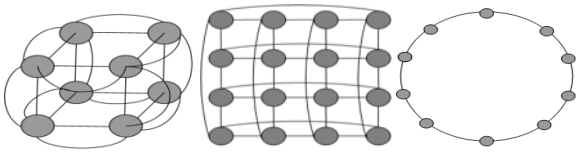


Fig. 7. (a) 3-dimensional torus (b) 2-dimensional torus (c) Rings (1-dimensional) network-on-chip topologies.

As an example, matrices of Fig. 3 for a mesh topology has been repeated for the same nodes in a 5×5 torus topology and shown in Fig. 8. Distance matrices of these two figures are equal regardless of the fact that the position of the source is changed.

$$\begin{bmatrix} 4 & 3 & 2 & 3 & 4 \\ 3 & 2 & 1 & 2 & 3 \\ 2 & 1 & 0 & 1 & 2 \\ 3 & 2 & 1 & 2 & 3 \\ 4 & 3 & 2 & 3 & 4 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 & 3 & 3 & 2 \\ 2 & 3 & 4 & 4 & 3 \\ 2 & 3 & 4 & 4 & 3 \\ 1 & 2 & 3 & 3 & 2 \\ 0 & 1 & 2 & 2 & 1 \end{bmatrix}$$

$(a) \qquad (b)$

$$D_{1*4}(5) = [4\ 8\ 8\ 4],\ k=4$$

Fig. 8. (a) Distance matrix for S = $X^*_{1,1}$, (b) Distance matrix for S = $X^*_{3,3}$ for torus topology

4. Results and Discussion

To evaluate the efficacy of using NED traffic profile in evaluating NoCs, we compared the average hop counts of all packets transported based on different traffic profiles. In addition, to NED, Transpose, Uniform, Hotspot 5%, Hotspot 10%, Hotspot 20%, and Bit complement, and traffic profiles generated based on some realistic applications mapping on an $n \times n$ mesh were used. For the Hotspot synthetic traffic profile, the hotspot point was chosen to be the node ($\lceil n/2 \rceil$, $\lceil n/2 \rceil$). For mapping realistic applications on n×n meshes, we used Adaptcell [16]. Adaptcell is a mapping tool that maps DSP or similar applications into distributed-control multiprocessor system on chip (MPSoC). This tool is capable of run-time task decomposition and scheduling capability. Each computational cell in this platform is a special processor which can be configured to 8, 16 or 32 bit mode. These cells are placed in a 2D-mesh topology and uses NoC scheme for communication. The size of each dimension of mesh can be configured easily.

Table 2 shows the average hop counts of synthetic and realistic traffic profiles on networks with different mesh sizes. For all the switches, the data width was set to 32-bits. Each input virtual channel had a buffer (FIFO) with the size of six flits. In all the simulations, the latency was measured by averaging the latency of the packets when each local core generated 30,000 packets. The router used the minimally fully adaptive reserved VC deadlock avoidance technique discussed in[17]. As this table shows, the average hop count of NED is more similar to those of realistic traffics than those of other synthetic traffic models. Other than NED, Uniform and Hotspot (which is a specific kind of uniform traffic) are among the synthetic traffic model with good hop count match to those of realistic benchmarks. Even for these traffic profiles, the differences between their average hop counts and the realistic ones are larger than that of NED. In the case of the semi realistic benchmark of "GSM + Uniform" [7], the difference is less. In this benchmark, just a few cores generate packets based on the GSM voice codec and the rest of the cores are sent packet based on the Uniform traffic profile. For other realistic benchmarks, NED traffic has closer average packet hops to those of these benchmarks. In addition, the difference of NED and other synthetic traffic profiles increases as the dimension of the network increases. Note that an empty slot means that the mesh size is too large for the application.

Fig. 9 also shows the average hop counts of different synthetic traffic models for different mesh sizes. The rate of increase in the average hop count in NED with the network size is lower than those of other synthetic traffic model. This behavior makes NED resembling more realistic traffics.

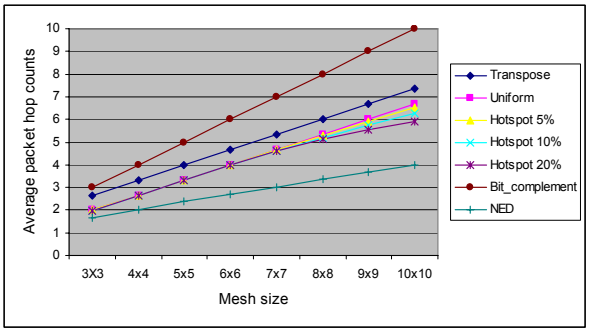


Fig. 9. Average hop counts of different synthetic traffics for different mesh sizes.

5. Conclusion

In this work, a synthetic traffic profile based on Negative Exponential Distribution (NED) for network on chips was proposed. In this traffic profile, the probability of sending a packet from a source to a destination decreases as the distance between them increases. This property made NED more similar to traffic profiles of real applications where the cores with higher packet interchange were mapped closer to each other to minimize the communication and delay. To show this property for NED, the average packet hops for some synthetic and realistic traffic profiles were compared. Experimental results showed similarity of NED traces with those of realistic applications running on NoCs with different sizes.

TABLE 2. Average packet hops for some synthetic and realistic traffic patterns

Traffic Type	Traffic Pattern	3×3	4×4	5×5	6×6	7×7	8×8	9×9	10×10
Synthetic	Transpose	2.6667	3.3333	4	4.6667	5.333	6	6.6667	7.333
	Uniform	2	2.6667	3.3333	4	4.6667	5.333	6	6.6667
	Hotspot 5%	1.9965	2.6632	3.332	3.9911	4.6398	5.2816	5.9167	6.518
	Hotspot 10%	1.9944	2.6632	3.3306	3.9834	4.6188	5.2011	5.761	6.2507
	Hotspot 20%	1.9884	2.66	3.3248	3.9718	4.598	5.1071	5.5311	5.8861
	Bit complement	3	4	5	6	7	8	9	10
	NED (m = 1/n)	**1.6519**	**2.0341**	**2.3995**	**2.6919**	**3.0255**	**3.3594**	**3.6933**	**4.0145**
Realistic	GSM+Uniform	-	2.1280	2.765	3.177	3.647	-	-	-
	Order 16 FIR	1.455	1.93	2.4544	-	-	-	-	-
	Order 24 FIR	-	2.121	2.478	3.5855	-	-	-	-
	5×5 Matrix Multiplication	1.2351	1.915	2.7747	3.0323	-	-	-	-
	7×7 Matrix Multiplication	1.575	2.156	2.5578	2.8952	3.2887	-	-	-
	8-points DCT	1.5961	1.8954	2.0425	-	-	-	-	-
	16-points DCT	-	2.252	2.45	2.8221	-	-	-	-

Acknowledgement

The authors wish to acknowledge the financial support by Iran Telecommunication Research Center (ITRC) during the course of this project.

References

[1] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, vol. 35, pp. 70–78, January, 2002.

[2] A. Jantsch and H. Tenhunen (Eds.), *Networks on Chip*, Kluwer, 2003.

[3] T. Bjerregaard AND S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," *ACM Computing Surveys,* Vol. 38, No. 1, 2006.

[4] M. L. Fulgham and L. Snyder, "Performance of Chaos and Oblivious Routers under Non-Uniform Traffic," *Technical Report UW-CSE-93-06-01*, Univ. of Washington, July 1993.

[5] K. Lahiri et al. Evaluation of the traffic-performance characteristics of system-on-chip communication architectures. *In Proc. of the 14th International Conference on VLSI Design*, pp. 29–35, Oct. 2000.

[6] W. J. Dally and B. P. Towles. Principles and practices of interconnection networks. *Morgan Kaufmann Publishers*, ISBN: 0122007514, San Francisco, CA, 2004.

[7] G. V. Varatkar and R. Marculescu, "On-chip traffic modeling and synthesis for MPEG-2 video applications," *IEEE Transactions of Very Large Scale Integration (VLSI) Systems*, Vol. 12, No. 1, Jan. 2004.

[8] D. Wu, B. M. Al-Hashimi, M. T. Schmitz, "Improving Routing Efficiency for Network-on-Chip through Contention-Aware Input Selection," in *Proc. of Asia and South Pacific Conference on Design Automation (ASPDAC)*, Jan. 2006, pp. 36-41.

[9] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The Splash-2 Programs: Characterization and Methodological Considerations," *In Proc. of ISCA-22*, June 1995.

[10] C. Lee et al. Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. *In Proc. of the 30th International Symposium on Microarchitecture (MICRO-30)*, pp. 330-335, Nov. 1997.

[11] The Standard Performance Evaluation Corporation. *Available [online]:*http://www.spec.org/.

[12] M. Nickray, M. Dehyadgari, and A. Afzali-Kusha, "Power and Delay Optimization for Network on Chip," *In Proc. of European Conference on Circuit Theory and Design (ECCTD'05),* pp. III-277–III-281, 2005.

[13] A. Mehran, A. Khademzadeh, A. Afzali-Kusha, and B. Shirpour, "A Heuristic Energy Aware Application Mapping Algorithm for Network on Chip," *In Proc. of IP Based SoC Design Conference & Exhibition, Grenoble*, France, pp. 289-294, Dec. 6-7, 2006.

[14] S. Murali and G. D. Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures," *In Proc. of DATE'04*, pp. 896-901, Feb. 2004.

[15] Robert V. Hogg, Allen Craig, Joseph W. McKean, "Introduction to Mathematical Statistics," *Prentice Hall*, 2005.

[16] Sh.Vakili, S. M. Fakhraie and S. Mohammadi, "Adaptcell: a NoC-based multiprocessor system with run-time task decomposition and scheduling capability," *will be appear in IET Computers & Digital Techniques*, 2009.

[17] L. M. Ni, and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *In IEEE computer*, pp. 62–76, 1993.

Latency, Power and Performance Trade-offs in Network-on-Chips by Link Microarchitecture Exploration

Basavaraj Talwar
Electrical Communication Engineering
Indian Institute of Science, Bangalore
bt@ece.iisc.ernet.in

Shailesh Kulkarni
ESAT, Katholieke Universiteit Leuven
3001 Heverlee, Belgium
shailesh.kulkarni@esat.kuleuven.be

Bharadwaj Amrutur
Electrical Communication Engineering
Indian Institute of Science, Bangalore
amrutur@ece.iisc.ernet.in

Abstract

This paper presents a power, latency and throughput trade-off study on NoCs by varying microarchitectural (e.g. pipelining) and circuit level (e.g. frequency and voltage) parameters. We change pipelining depth, operating frequency and supply voltage for 3 example NoCs - 16 node 2D Torus, Tree network and Reduced 2D Torus. We use an in-house NoC exploration framework capable of topology generation and comparison using parameterized models of Routers and links developed in SystemC. The framework utilizes interconnect power and delay models from a low-level modelling tool called Intacte[1][1]. We find that increased pipelining can actually reduce latency. We also find that there exists an optimal degree of pipelining which is the most energy efficient in terms of minimizing energy-delay product.

1. Introduction

Network-on-Chip design parameters such as topology generation and link pipelining have varying impacts on throughput of the network, latency of flits and power dissipation of the NoC in an SoC. The paper presents results on power-performance tradeoff studies on three NoC topologies (2D Torus, a Reduced 2D Torus and a Tree based network) by varying pipelining in links and frequency and voltage scaling. Using frequency scaling experiments we show that switching to a higher degree of link pipelining to achieve higher frequency instead of adding larger buffers is advantageous from a power perspective. A comparison of the three topologies based on throughput is presented. A SystemC based simulation framework containing parameterizable Routers, Links, Traffic generators and Sink nodes is used for NoC exploration. The framework uses Intacte[1] to estimate delay and power based on micro-architecture parameters such as wire length, wire width, activity for a given technology and voltage.

Rest of the paper is organized as follows. Some of the recent related works have been listed in Section 2. NoC exploration framework used in the tradeoff studies is described in Section 3. Latency, power, performance tradeoffs, Frequency scaling and Voltage scaling results are presented in Section 4. Paper concludes in Section 5.

2. Related Work

Current research in architectural level exploration of NoC in SoCs concentrates on understanding the impacts of varying topologies, link and router parameters on the overall throughput, area and power consumption of the system (SoCs and Multicore chips) using suitable traffic models[2]. Work in [3] emphasizes need for co-design of interconnects, processing elements and memory blocks to understand the effects on overall system characteristics. Simulation tools have been developed to aid designers in ICN space exploration[4][5]. Tools model ICN elements at system level and help in power/performance trade-off studies[6]. Another area of active research is design of Router architectures[11][12] and ICN topologies[10] with varying area/performance trade-offs for general purpose SoCs or to cater to specific applications.

Kogel et. al.[4] present a modular exploration framework to capture performance of point-to-point, shared bus and crossbar topologies. Orion[5] is a power-performance interconnection network simulator that is capable of providing power and performance statistics. Orion model estimates power consumed by Router elements (crossbars, FIFOs and arbiters) by calculating switching capacitances of individual circuit elements.

Previous works largely concentrate on Router power and do not take into account various link microarchitectural parameters for power and performance trade-off calculations. This paper presents results for NoC power by considering

[1]We thank developers of INTACTE for making the tool available for our research.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.55

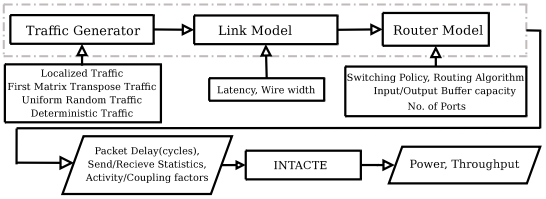
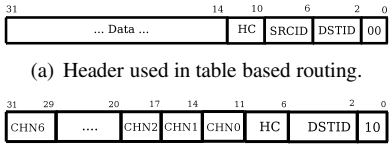


Figure 1. Flow of the ICN exploration framework.



(a) Header used in table based routing.

(b) Header used in source routing.

Figure 2. Example flit header formats considered in this experiment. (DST/SRCID: Destination/Source ID, HC:Hop Count, CHNx:Direction at hop x).

Table 1. ICN exploration framework parameters.

Parameter	*Description*
NoC Parameters	
Routing Algorithms	Source Routing and Table based routing
Switching Policy	Packet, Circuit, Wormhole, VC switching
Traffic Generation Scheme	Deterministic, Self-Similar
Traffic Distribution Scheme	Deterministic, Uniformly random HotSpot, Localized, First Matrix Transpose
Router Microarchitecture	
No. of Input/Output Ports	2-8 (based on topology to be generated)
Input/Output buffer sizes	Flit-level buffers
Crossbar Switching capacity	In terms of flits (default=1)
Link Microarchitecture	
Length of interconnect	Longest link in mm
Bit width of the interconnect	
Circuit Parameters	
Frequency, Supply Voltage	

effects of various pipelining configurations, frequency and voltage scaling values. Various traffic generation and distribution models have been used to mimic realistic traffic patterns and activity in NoCs.

3. NoC Exploration Framework

The NoC exploration framework (Figure 1) has been built upon Open Core Protocol-IP models[8] using OSCI SystemC 2.0.1[9] on Linux (2.6.8-24.25-default). The framework contains Router, Traffic generator and consumer modules, Latency modules (to model link latency). A single run outputs data files from which latency and power statistics are extracted. Table 1 presents various parameters that can be varied in the framework. Most options a designer might encounter during NoC design process have been added into the framework.

3.1. NoC Elements

Router Model. The router model is a parameterized, scalable module of a generic router[2]. *Router microarchitecture* parameters include number of Input/Output ports, sizes of input/output buffers, switching capacity of the crossbar (no. of bits that can be transfered from input to output buffers in a cycle) etc. Example *Routing algorithms* are source and table based routing. *Switching policies* such as circuit switching, packet switching, wormhole switching have been implemented. Flow control implemented through sideband signals ([8]) prevents traffic generators from spewing phits into the network after input buffer fills to a threshold. Router model has been carefully designed to be easily adapted for use in various topologies (with varying flit header formats as shown in Figure 2) with minimal changes.

Traffic Generator. Traffic models implemented in the Traffic Generator module are Deterministic, Uniformly Random, Localized, Hotspot and First Matrix Transpose traffic. Each of the models vary in how many and how often do destination nodes receive flits from a given generator. In the current implementation flit header formats have been varied based on the type of routing scheme used. The flit header formats for Source routing and Table based routing are shown in Figure 2. Source routing header (2(a)) is larger as it contains 'directions' per hop the flit has to traverse whereas the Table based routing header (2(b)) contains the final destination address only. The examples are shown for a 4x4 2D mesh topology. The framework also contains sink nodes to receive flits and a top module to instantiate the framework.

3.2. Power Model

We use the Intacte[1] for interconnect area, delay and power estimates. Design variables the tool considers for interconnect optimization are wire width, wire spacing, repeater size and spacing, degree of pipelining, supply (V_{dd}) and threshold voltage (V_{th}). Intacte considers activity and coupling factors to calculate dynamic power and includes short circuit and leakage power statistics to calculate total power dissipation in a bus made of wires with known length, spacing, repeater sizes and repeater spacing. The SystemC framework(Figure 1) generates activity and coupling factors averaged over the total simulation run for each of the links to be input into Intacte. Wire width (in bits) is known per simulation run. Wire lengths can be estimated

Table 2. Experimental Setup

Traffic Injection Rate	20%
Traffic Model	Localized Traffic (6%)
Framework simulation time	35000 cycles
Process Technology	65nm
Models	PTM[7]
Frequency of Operation (unless mentioned otherwise)	1 GHz
Environment	Linux (2.6.8-24.25-default)+ OSCI SystemC 2.0.1

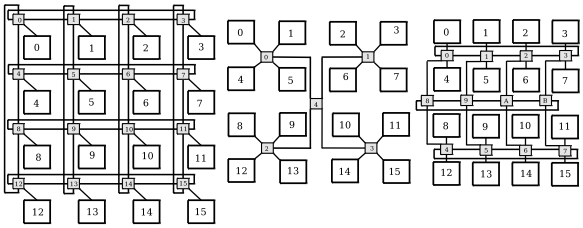


Figure 3. Schematic representation of the three compared topologies (L to R: 2D Torus, Tree, Reduced 2D Torus). Shaded rectangles are Routers and white boxes are source/sink Processing Elements(PE) nodes.

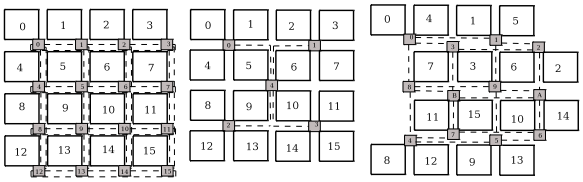


Figure 4. Approximate floorplans of the three compared topologies.

Table 3. Links and pipelining details of NoCs

Topology	Length in mm (no. of links)	Pipelining							
2D Torus	7 (8)	1	2	3	4	5	6	7	8
	1.5 (88)	1	1	1	1	2	2	2	2
Reduced 2D Torus	3.5 (12)	1	2	3	4	5	6	7	8
	2.5 (16)	1	2	3	3	4	5	5	6
	2.0 (44)	1	2	2	3	3	3	4	4
Tree NoC (8+32)	3.5 (8)	1	2	3	4	5	6	7	8
	0.75 (32)	1	1	1	1	2	2	2	2

by approximate floorplans (Figure 4). Minimum wire spacing is obtained from foundry rules. Intacte solves an optimization problem to arrive at optimal number of repeaters, repeater spacing for a given frequency and voltage. The tool also includes flop and driver overheads for power and delay calculations. Other physical parameters are obtained from Predictive Technology Models[7] models for $65nm$.

Power consumed by routers have not been included in the results presented in the paper. This is the next step in our work. This is not a limitation as we are concerned with relative variations due to changes in link microarchitecture and circuit parameter. Identical routers have been used in all experiments in this work.

4. Simulation and Results

Experiments are designed to calculate latency (in clock cycles), throughput (in gigabits/sec) and power in (milliwatts) of various topologies. A comparative study of three related topologies (2D Torus, Reduced 2D Torus and a Tree based NoC) is made. Table 2 lists some of the simulation setup parameters used in the following experiments. We did not observe significant variation in activity factor and hence power and throughput of the NoC by running the simulation for durations greater than 35000 cycles.

4.1. NoC Topologies

In this work we consider three similar topologies for tradeoff studies. Starting from a 2D Torus, two topologies (a hierarchical star topology and reduced Torus) containing equal number of source and sink nodes are derived by removing/reconnecting links. Router and processing elements are identical in all three topologies. The three topologies are shown in Figures 4 (schematic) and 4 (approximate floorplan). Processing elements (PEs) are assumed to be of size 1.5×1.5mm[13] Routers are assumed to be 15% of the PE size. Lengths of longest link in 2D Torus is estimated to be $7mm$ and in Reduced 2D Torus and Tree based NoC is $3.5mm$. Routing policies for all topologies is table based. Routing tables are populated such that longer links have minimum activity. Lengths of links in each of the topologies and pipelining factors is illustrated in Table 3. Pipelining factor corresponds to the longest link in the NoC. Pipelining factor of 1 means the longest link is unpipelined, P=2 indicates it has a two cycle latency and so on.

4.2. NoC Throughput

Throughputs of each of the NoC topologies are calculated in this subsection. *Localized traffic* generation scheme (each traffic generator sends 6% of its traffic to its immediate neighbors) with *self-similar traffic* distribution is used. Throughput is a measure of total data consumption at sink nodes. Total throughput of the NoC (in $bits/sec$) is calculated as total number of bits received ($(phits_r * bits_{phit})$) at sink nodes divided by total (real) time ($(\frac{1}{f} * sim_{cycles})$) spent (Eqn 1).

$$Th_{total} = \frac{phits_r * bits_{phit}}{\frac{1}{f} * sim_{cycles}} \quad (1)$$

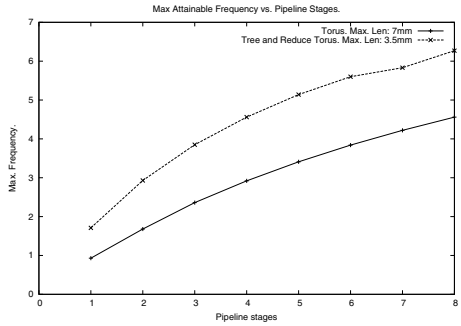


Figure 5. Maximum attainable frequency by links in the respective topologies. Estimated length of the longest link in a 2D Torus is 7mm. Estimated longest link in the Tree based and Reduced 2D Torus is 3.5mm.

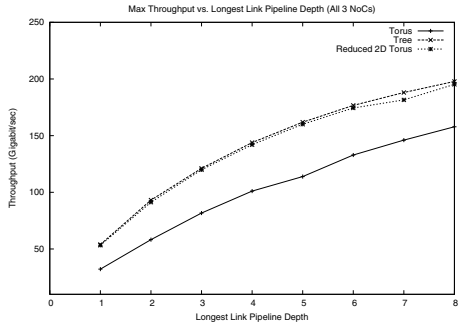


Figure 6. Variation of total NoC throughput with varying pipeline stages in all three topologies.

Maximum achievable frequency of a wire of a given length is shown in Figure 5. Maximum throughput of each NoC is presented in Figure 6. Tree based NoC supports localized traffic the best (atleast two neighbours at one hop distance) and hence shows highest throughput. Both Tree NoC and Reduced 2D Torus show higher throughput because of shorter links resulting in higher frequency of operation. The Reduced 2D Torus has higher throughput than a conventional 2D Torus as the minimum distance between two neighbours is 1 hop (2 hops in case of a Torus).

4.3. NoC Power/Performance/Latency Tradeoffs

Figure 7 shows the combined normalized results of power, throughput and latency experiments on a **2D Torus**. Power consumption of the 2D Torus increases at a higher rate after P=4 due to insertion of flops in the shorter links ($1.5mm$) after P=5. Latency is calculated as the real time spent in transit of all phits in the NoC over the complete simulation time. Decrease in latency after P=5 is not considerable as delays from inserted flops start to dominate

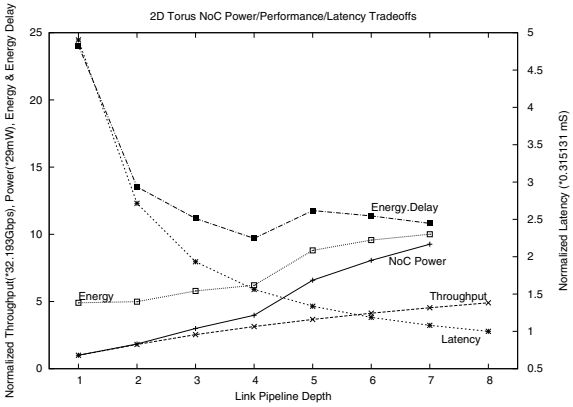


Figure 7. 2D Torus Power/Throughput/Latency tradeoffs. Normalized results are shown.

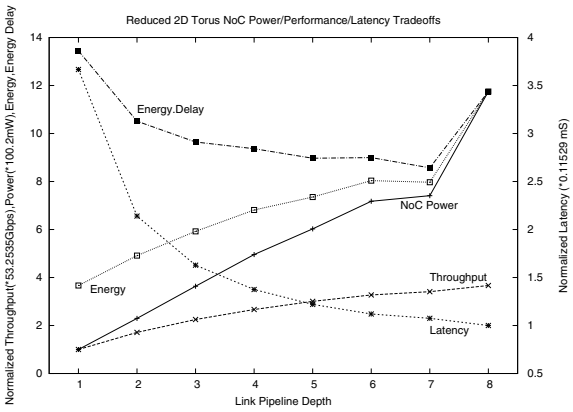


Figure 8. Reduced 2D Torus Power/Throughput/Latency tradeoffs. Normalized results are shown.

clock cycle time and after a certain pipeline configuration latencies will increase (not shown here). From the graph it is seen that growth in power makes configurations more than P=5 less desirable. Link pipelines with P=1,2 and 3 are also not optimal when latency is considered. Rise in throughput also starts to fade as configuration of more than P=5 are used. The optimal point of operation indicated by the results is P=4. At this point the same number of flops as P=1,2 and 3 are used but the least latency (1.56 times minimum) is achieved and power (40% of max) and throughput (64% of max) are at nominal points. The graph also show energy (Power $\times$ Latency) required for the communication. Energy for communication increases with pipeline depth. However the energy delay product reduces initially with increasing pipeline depth and then increases with a minimum around P=4.

Tradeoff results on **Reduced 2D Torus** are shown in Figure 8. Latency and throughput curves show similar trend as in a 2D Torus. Latency reduction and throughput gain after

P=4 is not considerable. The power optimal point of operation indicated by the results is P=3. At P=3 latency is 1.6 times the minimum and power (49% of max) and throughput (61% of max). There is a shallow minima for energy-delay product from P=3-7.

4.4. Power-Performance Tradeoff With Frequency Scaling

We discuss the combined effects of pipelining links and frequency scaling on power consumption and throughput of three example topologies (Figure 4) in this subsection. Maximum possible frequency of operation at full supply voltage $(1.1V)$ is determined using Intacte.

Figure 9 shows NoC power consumption for 3 example topologies over various pipelining factors along with frequency scaling. Maximum frequency of operation of an unpipelined longest link in a **2D Torus** (we consider $7mm$) is determined to be 0.93GHz. This maximum throughput point determined in each pipeline configuration in each topology. Frequency is scaled down from this point and power measurements are made for NoC activity obtained using the SystemC framework for *Localized traffic* with 20% injection rate and 6% localization factor. Allowing for some overheads (extra cycles of latency), the frequency of operation required to achieve equivalent throughput in pipelined links is $0.94 - 0.96$GHz. Higher frequency translates to higher throughput (Eqn. 1). Pipelining factor of 1 is unpipelined and has single cycle delay and a factor of 2 means link has two cycles of delay and so on.Experiments for each topology show existence of crossover frequencies after which it is better to switch to a higher pipelining to save power and achieve higher throughput. Larger buffers are required to drive links at higher frequencies. Power consumed by buffers starts to overshadow the frequency gain at these frequencies. Experiments on the 2D Torus show that a link frequency of 2.5GHz can be achieved by pipelining the link in stages 4 to 7. NoC power consumption can be reduced by 40.97% by switching to a 4 stage pipeline from a 7 stage pipeline. Another interesting result is the effect of larger buffers as upper limits of frequency are reached in a single pipeline configuration. For instance, in P=3 from 2.3GHz to 2.4GHz, buffers start to consume almost the same power as a link with P=4.

Sizes of links of a **Reduced 2D Torus** are estimated to be $3.5mm$, $2.5mm$ and $2.0mm$. NoC power consumption across various pipeline stages differ by smaller amounts compared to the 2D Torus as the number of links are lesser (32+16 bidirectional compared to 12+8+16 bidirectional links). Results show frequencies of 4.22GHz - 4.56GHz can be achieved by both P=4 and P=5 (5% power difference). Complementarily, for a given frequency there exists more than one pipeline configuration with varying power

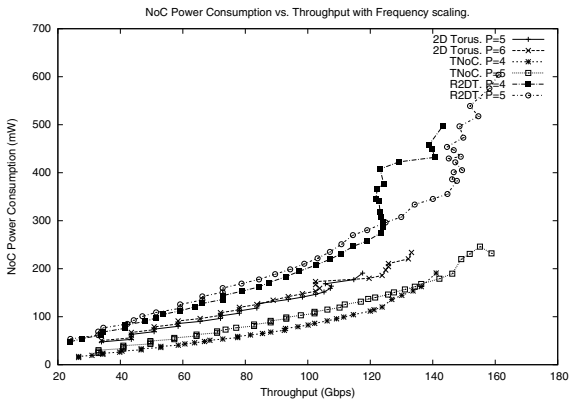


Figure 9. Variation of NoC power with throughput for each topology.

Table 4. Power optimal frequency trip points in a various NoCs.

Pipe Stages	Trip Frequency (in GHz)		
	2DT	R2DT	Tree NoC
1-2	0.93	1.65	1.05
2-3	1.71	2.75	2.1
3-4	2.36	3.55	3.05
4-5	3.4	4.22	4.45
5-6	3.84	5.13	4.75
6-7	4.22	5.3	5.13

consumption. Frequency of 3.5GHz can be achieved by pipelining into 3 to 7 stages. NoC power consumption can be reduced by 26.6% by switching from P=7 to P=3 and still achieve 3.5GHz. Table 4 shows power reduction when switched from one pipeline configuration to the next higher one at 'trip' frequencies.

Estimated sizes of links in a **Tree Based NoC** are $3.5mm$ and $0.75mm$. Results for the frequency scaling experiment follow a similar pattern as the previous two configurations (Figure 9). The differences between power numbers between various configurations are the least in this network as this NoC contains the least number of links amongst the three compared (4+16 bidirectional). Trip frequencies are recorded in Table 4. A maximum of 21.27% of power can be saved (at $f = 3.84$GHz) by switching over to P=4 from P=3 after 3.05GHz. Complementarily, 4GHz can be achieved by P=4 to P=7 and NoC power consumption at P=4 is 76% power consumed at P=7.

4.5. Power-Performance Tradeoff With Voltage and Frequency Scaling

In each of the topologies, frequency is scaled down from the maximum and the least voltage required to meet the scaled frequency is estimated using Intacte and power con-

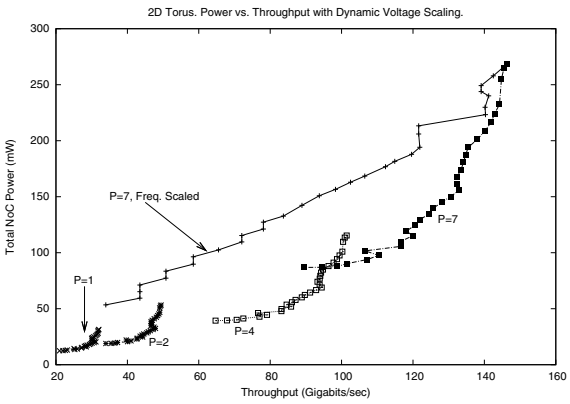


Figure 10. Effects of dynamic voltage scaling on the power and performance of a 2D Torus. Highest frequency of operation for P=1, 2, 4 and 7 are .93GHz, 1.68GHz, 2.92GHz and 4.22GHz. Power consumption of the frequency scaled NoC is shown for comparison.

Table 5. Comparison of 3 topologies. Maximum interconnect network performance and power consumption for varying pipe stages.

Topology	Pipe Stages	Power (mW)	Performance (Gbps)
2D Torus	1	32.01	31.5
	2	49.44	53.27
	4	101.42	115.34
	7	146.41	268.61
Reduced 2D Torus	1	49.05	100.2
	2	91.75	230.95
	4	142.5	496.25
	7	181.7	742.27
Tree Network	1	53.22	52.93
	2	90.66	99.46
	4	141.17	191.07
	7	179.77	307.6

sumption and throughput results are presented in this section. Voltages are scaled from $1.1V$ to $0.65V$. NoC parameters are identical to ones used in Section 4.2. Figure 10 shows results of DVS on the 2D Torus network. Similar to the frequency scaling results there exists an frequency point in a pipelining configuration after which it is power and throughput optimal to switch to a higher pipelining stage. For throughput higher than 90Gbps P=7 offers a highest power reduction of 21.74% at 101Gbps. The frequency scaled curve is obtained by scaling only the frequency and NoC is run at supply voltage. Scaling voltage along with frequency compared to scaling frequency alone can result in power savings of upto 57% and 63% in cases of P=7 and P=4 respectively. Comparison of all the three topologies is presented in Table 5.

5. Conclusion

NoC design specifications can be met by varying a large number of system and circuit parameters. We use 3 example topologies - 16 node 2D Torus, Tree network and Reduced 2D Torus to show variation of latency, throughput and NoC power consumption over link pipelining configurations with voltage and frequency scaling. We find that contrary to intuition, increasing pipeline depth can help reduce latency in absolute time units, by allowing shorter links & hence higher frequency of operation. In a 2D Torus when the longest link is pipelined by 4 stages at which point least latency (1.56 times minimum) is achieved and power (40% of max) and throughput (64% of max) are nominal. Using frequency scaling experiments, power variations of upto 40%, 26.6% and 24% can be seen in 2D Torus, Reduced 2D Torus and Tree based NoC between various pipeline configurations to achieve same frequency at constant voltages. Also in some cases, we find that switching to a higher pipelining configuration can actually help reduce power as the links can be designed with smaller repeaters. Larger NoC power savings can be achieved by voltage scaling along with frequency scaling. Hence it is important to include the link microarchitecture parameters as well as circuit parameters like supply voltage during the architecture design exploration of a NoC.

References

[1] R. Nagpal, M. Arvind, Y. N. Srikanth, and B. Amrutur. Intacte: Tool for interconnect modelling. In *Proc. of CASES 2007*, pages 238–247, 2007.

[2] P. P. Pande, et. al., Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE Transactions on Computers*, 54:1025–1040, Aug. 2005.

[3] R. Kumar, V. Zyuban, and D. M. Tullsen. Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In *Proc. of ISCA '05*, pages 408–418, 2005.

[4] T. Kogel and et. al. A modular simulation framework for architectural exploration of on-chip interconnection networks. In *Proc. of CODES+ISSS'03*, pages 338–351, Oct. 2003.

[5] H.-S. Wang, et. al., Orion: A power-performance simulator for interconnection networks. In *Proc. of, MICRO 35*, 2002.

[6] P. Gupta, L. Zhong, and N. K. Jha. A high-level interconnect power model for design space exploration. In *Proc. of, Computer Aided Design (ICCAD '03). Intl. Conf. on*, pages 551–558, 2003.

[7] http://www.eas.asu.edu/∼ptm/. Predictive technology models.

[8] http://www.ocpip.org/socket/systemc/. Ocp-ip, systemc ocp models.

[9] http://www.systemc.org/. Open systemc initiative.

[10] F. Karim and et. al. An interconnect architecture for networking systems on chips. *IEEE Micro*, 22:36–45, Sept. 2002.

[11] K. Lee, S.-J. Lee, and H.-J. Yoo. Low-power network-on-chip for high-perforamance soc design. *IEEE Transactions on VLSI Systems*, 14:148–160, Feb. 2006.

[12] S. E. Lee, J. H. Bahn, and N. Bagherzadeh. Design of a feasible on-chip interconnection network for a chip multiprocessor (cmp). In *Proc. of, Computer Architecture and High Performance Computing. Intl. Symp. on*, pages 211–218, 2007.

[13] S. Vangal, et. al., An 80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of Solid-State Circuits*, 43:29–41, Jan. 2008.

Session 3A

Low Power Device Technology

A Low Voltage CMOS Proportional-to-Absolute Temperature Current Reference

Sanjay Kumar Wadhwa
Freescale Semiconductor India Pvt. Ltd.
sanjay.wadhwa@freescale.com

Abstract

A CMOS low voltage Proportional-to-Absolute Temperature current reference is presented. The proposed circuit can work with supply voltages as low as 1.1V. The circuit is designed in 90nm CMOS technology for 2.2uA reference current at typical corner, 27C, 1.2V. The circuit has been extensively simulated across all possible combinations of MOSFET, Resistor, BJT, supply voltage and temperature variation corners. Simulation results have been given for a wide temperature variation from -40C to 125C and supply voltage variation from 1.1V to 1.3V.

I. Introduction

Proportional-to-Absolute Temperature [PTAT] current references are used in many applications such as band-gap references, phase lock loop (PLL), hearing aid devices, sensors etc. to compensate for circuit parameters due to temperature change [1]. For example, in a PLL system, if the current controlled oscillator (CCO) gain decreases with increase in temperature, a PTAT current reference can be used to compensate for the gain to a large extent. In current ultra deep sub-micron (UDSM) technologies, the typical supply voltage used is 1.2V or less. Thus, a PTAT current reference which can work with supply voltage equal to or less than 1.2V is required for today's SoCs. In order to achieve low voltage operation, a previously reported circuit technique [1] uses MOSFETs in sub-threshold region instead of bipolar transistors. However, use of MOSFETs in sub-threshold region requires large area devices even for the reference current in nano-amp (nA) range. This increases the die size of the circuit. Secondly, sometimes the availability of accurate models of MOSFETs in sub-threshold region is not guaranteed. In UDSM technologies, accurate modeling of all regions of MOSFET operation is becoming very challenging. This aspect must be kept in mind while designing circuits with MOSFETs in sub-threshold region. The proposed circuit achieves low voltage operation with the use of parasitic bipolar transistors only which are readily available in a standard CMOS process.

II. Circuit Description

A conventional PTAT reference is shown in Fig. 1 [2]. Due to Opamp's virtual short action, voltages at node A and B become nearly equal. The current flowing through MP1 and MP2 is same and is given by

$$I = \frac{V_T \ln(m)}{R_1} = \frac{kT \ln(m)}{qR_1} \quad (1)$$

where V_T is thermal voltage, k is Boltzman constant, q is electronic charge, T is absolute temperature and m is area ratio of Q1 and Q2 as shown in Fig. 1. The typical value of V_{be} is 0.7V at room temperature and it typically varies by approximately -2mV/DegC. Therefore, for a temperature range of -40DegC to 125DegC, the V_{be} will vary from 830mV to 500mV. At supply voltage of 1.2V, it is difficult to design an opamp for such a wide common mode range [3]. In the bandgap reference circuit proposed in [3], the authors modified the opamp architecture to work at sub-1V supply. The common mode voltage was reduced by half by using resistor dividers in parallel to both the PNP transistors. However, due to this, the circuit ceases to remain a PTAT current reference and becomes a bandgap reference circuit.

Fig. 2 shows the proposed PTAT current reference circuit. In Fig. 2, MP1, MP2, Q1, Q2 and R1 form a PTAT current reference. MN1, MN2, MN3, MP3, MP4, MP5 and MP6 constitute an OTA structure without a tail current source. The drain of MN3 is connected to the base terminals of Q1 and Q2 at node base. The gate of MP1 has been connected to the drain of diode connected transistor MP7 at node mp1_g. Similarly, gate of MP2 has been connected to the drain of diode connected transistor MP8 at node mp2_g.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.18

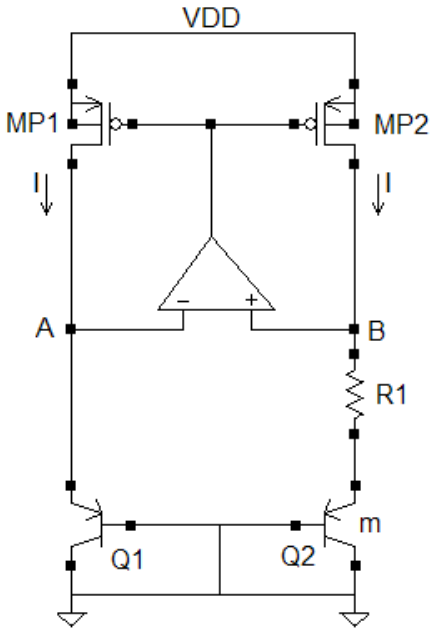


Fig. 1: Conventional PTAT current reference

The drains of MP1 and MP2 are connected to the gates of MN4 and MN5 at nodes mp1_d and mp2_d respectively. The sources of MN4 and MN5 have been connected to the joining points of resistors R2A, R2B and R3A, R3B at nodes mn4_s and mn5_s respectively. MP5 and MP6 act as high swing cascode transistors to improve the output resistance of the OTA. The gate voltage of MP5 and MP6 is tapped from the joining point of R2B and R2C at node casc. The PTAT current flowing through MP1 and MP2 is given by eqn. (1). In the proposed design, the sizes of MP3, MP4, MP7 and MP8 are same as that of MP1 and MP2. Ignoring channel length modulation effect, the currents in MP3, MP4, MP7 and MP8 will be same as in MP1 and MP2 respectively. The drain to source voltage of MP1 is given by

$$V_{DS(MP1)} = V_{GS(MP1)} + I * R2A - V_{GS(MN4)} \quad (3)$$

Similarly, the drain to source voltage of MP2 is given by

$$V_{DS(MP2)} = V_{GS(MP2)} + I * R3A - V_{GS(MN5)} \quad (4)$$

Where I is the current flowing in MP7 and MP8. The minimum supply voltage required for proper operation of the circuit is equal to the sum of V_{DS} of MP1 (or MP2), V_{be} of Q1 (or Q2) and V_{DS} of MN3. In order to achieve low voltage operation, W/L ratio of transistors and values of resistors have been chosen to have a maximum V_{DSsat} of 150mV for MP1 and MP2. Taking a V_{be} value of 830mV at -40C and a minimum V_{DS} of 100mV for MN3, the minimum supply voltage required for the circuit to operate is close to 1.1V. Since there is no tail current source used in the OTA, the voltage headroom required to keep the tail current source in saturation is not required. Also, design of OTA becomes very simple and helps in achieving low voltage operation. The startup circuit used in the proposed circuit is described in detail in Figure 1 in [4].

There is an overall negative feedback in the circuit which keeps the loop stable. The base voltage of Q1 and Q2 is adjusted by the loop depending upon the operating condition of the circuit. To stabilize the loop, R_COMP and C_COMP have been used. In order to analyze the overall negative feedback in the circuit, let's assume that the base voltage is pulled down slightly. Due to this, current in Q1 and Q2 will increase. The voltage at node mp2_d will decrease less as compared to the voltage at node mp1_d due to increased voltage drop across resistance R1. MN4 and MN5 act as level shifters, therefore node mn5_s will decrease less as compared to node mn4_s. MP7, R2A and MP8, R3A act as a voltage dividers and therefore, node mp2_g will be decrease less as compared to node mp1_g. Due to this, the current in MP3 will increase more as compared to current in MP4. Therefore, the gate voltage of MN3 (voltage at node out) will decrease which will pull up the base voltage. Similarly, if base voltage is pulled up slightly, voltage at node mp2_g will be increase less as compared voltage at node mp1_g. Due to this, the current in MP3 will decrease more as compared to current in MP4. Thus, voltage at node out will increase and base voltage will be pulled down. The high gain of OTA along with the overall negative feedback ensures that the voltages at node mp1_g and mp2_g become nearly the same. This essentially means that voltages at node mp1_d and mp2_d will also be nearly equal because both are level shifted by the same amount by MN4 and MN5 respectively.

III. Simulation results

The proposed circuit has been designed in 90nm CMOS technology and simulated across PVT corners for a typical PTAT current of 2.2uA at 27C.

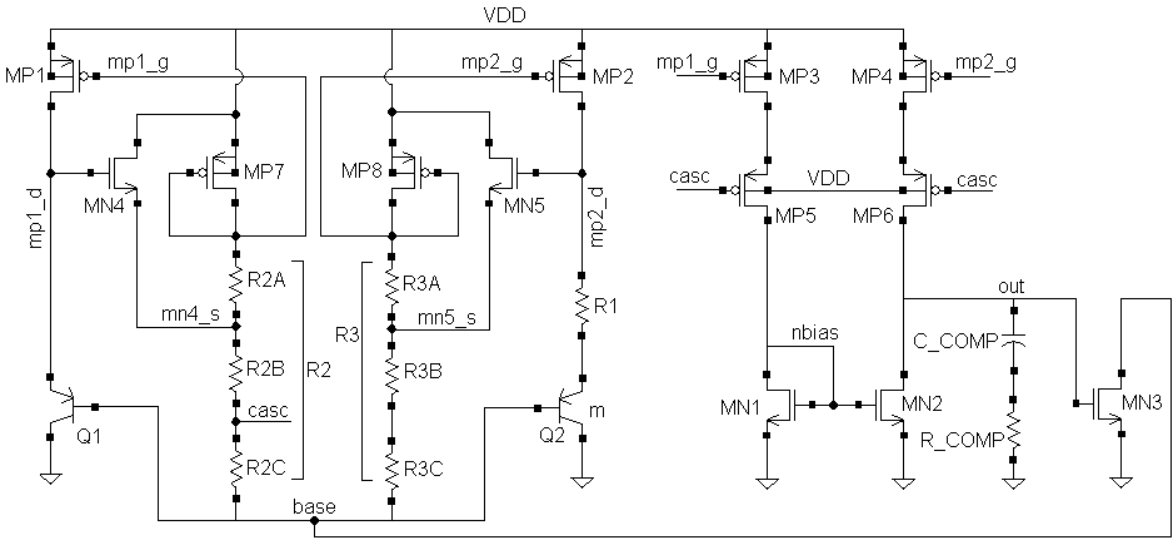


Fig. 2: Proposed PTAT current reference

For simulations, voltage variation has been taken from 1.1V to 1.3V and temperature variation has been taken from -40C to 125C. Fig. 3 shows $I(_{MP1})$ with temperature swept from -40C to 125C at typical corner, 1.2V.

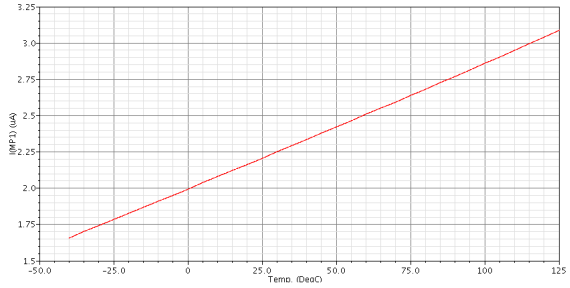


Fig. 3: I(MP1) with temperature sweep at typical corner, 1.2V

Table 1 shows $I(_{MP1})$ across supply voltage variation from 1.1V to 1.3V. It is clear from Table 1 that there is very low variation in PTAT current with supply variation.

Table 1: I(MP1) with supply variation from 1.1V to 1.3V at typical corner

Temp.	@1.1V	@1.2V	@1.3V
@-40C	1.6509uA	1.6634uA	1.6698uA
@27C	2.2127uA	2.2234uA	2.2296uA
@125C	3.0714uA	3.0867uA	3.0961uA

Fig. 4 shows the transient response of the proposed circuit at typical corner, 27C, 1.2V.

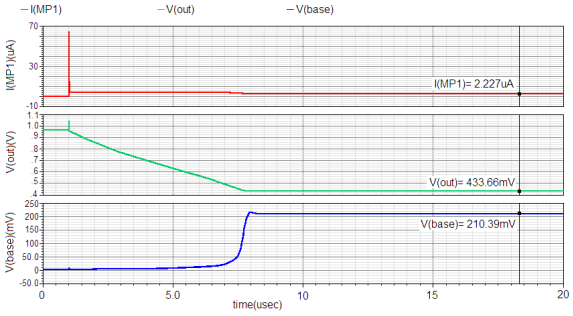


Fig. 4: Transient response of the proposed circuit at typical corner, 27C, 1.2V

Fig. 5 shows the power supply rejection (PSR) of $I(_{MP1})$ at typical corner, 27C, 1.2V. The PSR value @1MHz is -98.96 dB as shown in Fig. 5.

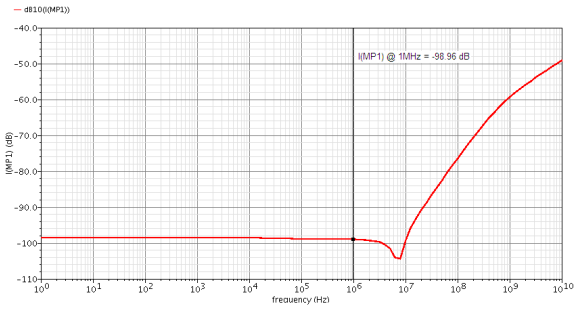


Fig. 5: PSR versus frequency at typical corner, 27C, 1.2V

Table 2 summarizes the simulated results of the proposed circuit across PVTs. The minimum and maximum values of $I(_{MP1})$, $I(_{MP2})$, $I_{DD(ON)}$ and $I_{DD(OFF)}$ have been shown along with their corresponding PVT corners. Wcs, typ and bcs denote

worst case, typical and best case MOSFET corners respectively while bnwp and wnbp denote the skewed corners.

Table 2: Simulated results of the proposed PTAT current reference across PVT

Para-meter	min	typ	max
$I_{(MP1)}$ (uA)	1.092 @ bnwp_4sig_fet, max_resistor, typ_bjt, vdd=1.1, temp=-40	2.227 @ typ_fet, typ_resistor, typ_bjt, vdd=1.2, temp=27	4.5079 @ wnbp_fet, max_resistor, bcs_bjt, vdd=1.3, temp=125
$I_{(MP2)}$ (uA)	1.144 @ bnwp_4sig_fet, max_resistor, typ_bjt, vdd=1.1, temp=-40	2.233 @ typ_fet, typ_resistor, typ_bjt, vdd=1.2, temp=27	4.4456 @ wnbp_fet, max_resistor, bcs_bjt, vdd=1.3, temp=125
$I_{DD(ON)}$ (uA)	13.48 @ wcs_4sig_fet, max_resistor, wcs_bjt, Vdd=1.1, temp=-40	18.27 @ typ_fet, typ_resistor, typ_bjt, vdd=1.2, temp=27	26.34 @ wnbp_fet, max_resistor, bcs_bjt, vdd=1.3, temp=125
$I_{DD(OFF)}$	1.006 nA @ wcs_4sig_fet, max_resistor, typ_bjt, vdd=1.1, temp=-40	16.95 nA @ typ_fet, typ_resistor, typ_bjt, vdd=1.2, temp=27	1.0903 uA @ bcs_4sig_fet, min_resistor, bcs_bjt, vdd=1.3, temp=125

Fig. 6(a), 6(b) and 6(c) show the histograms of $I_{(MP1)}$ obtained from Monte Carlo mismatch simulations at -40C, 27C and 125C respectively. At each temperature corner, the mismatch simulations have been run at all combinations of MOSFET, resistor, BJT and supply voltage corners with 100 samples at each corner. The mean and sigma values at each temperature are shown in Table 3.

Table 3: Mean and Sigma Values of I(MP1) obtained from Monte-Carlo mismatch simulations

Temp.	Mean (uA)	Sigma (uA)
@-40C	1.629152	0.358897
@27C	2.23505	0.447425
@125C	3.328190	0.8654516

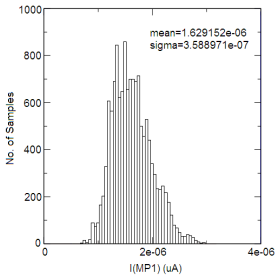


Fig. 6(a): At -40C

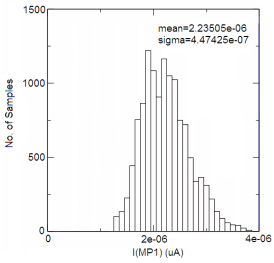


Fig. 6(b): At 27C

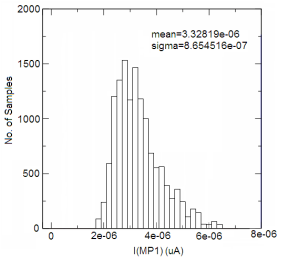


Fig. 6(c): At 125C

IV. Conclusion

The paper described the design of a low voltage PTAT current reference circuit capable of working across a wide temperature range of -40C to 125C. Simulation results show that the proposed circuit is capable to work reliably with supply voltage down to 1.1V.

V. References

[1] Francisco Serra-Graells et el, "Sub-1-V CMOS Proportional-to-Absolute Temperature References," *IEEE J. Solid State Circuits*, vol. 38, pp. 84–88, January 2003

[2] Wang Zongrnin et el, "Low voltage, high performance bandgap reference in standard CMOS technology," *IEEE Int. Workshop VLSI Design & Video Tech,* Suzhou, China, May 28-30, 2005.

[3] Mikko Waltari and Kari Halonen, "Reference Voltage Driver for Low-Voltage CMOS A/D Converters", *IEEE Transactions on Circuits and Systems –II Analog and Digital Signal Processing,* vol. 50, No.12, Dec. 2003, pp. 928 - 932.

[4] Khan Q. A., Wadhwa S. K., Misri K., "Low Power Startup Circuits for Voltage and Current Reference with Zero Steady State Current", ISLPED'03, August 25-27, 2003, Seoul, Korea

2009 22nd International Conference on VLSI Design

Novel MOS Decoupling Capacitor Optimization Technique for Nanotechnologies

Bardia Bozorgzadeh, and Ali Afzali-Kusha
Nanoelectronics Center of Excellence, School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran
b.bozorgzadeh@ece.ut.ac.ir, afzali@ut.ac.ir

Abstract

Designing MOS decoupling capacitors (DECAPs) in nanotechnologies provides many challenges due to the existing trade-offs among transient time response behavior, area, and gate leakage current. In this paper first it is shown that all of these challenges are functions of the MOS DECAP channel length. Then, we propose a method for optimizing the channel length of MOS DECAPs. The technique is applied to 45nm and 32nm technology nodes and the results are extracted using HSPICE simulations. Also the accuracy of the proposed technique is verified. Finally, based on the results, two optimum DECAP configurations which provide trades off among area and gate leakage for different applications in nanotechnologies are proposed.

1. Introduction

Signal integrity has become a critical issue as VLSI technology advances into the nanotechnology regime. Among the signal integrity issues, power supply noise is of particular importance and provides many challenges in the design. These challenges include unpredictability in the timing behavior of logic gates [1], increasing gate delay [2][3], degrading the drive capability of transistors [4], and logic failures [1][4]. In general, a design objective is to keep voltage fluctuations bounded by a given limit in order to limit their corresponding impact on performance. This limit is usually considered to be between 5 and 10% [4]-[7].

Among power supply noise reduction techniques, inserting on-chip decoupling capacitors (DECAPs) is the most common practice [1][4][11]. The DECAPs, which hold a reservoir of charge, are placed around regions of high current demand [1]. When large drivers switch, nearby DECAPs provide a source of current that reduces *IR* and *Ldi/dt* voltage drops to keep the target average and peak supply voltages within their noise budgets [1]. DECAPs are usually made from MOS transistors [8].

Designing DECAPs provides many challenges due to the existing trade-offs among transient time response behavior, area, and gate leakage current [1][8][11][15]. Several works have focused on the DECAP design challenges. For example in [1] and [11] authors have dealt with transient time response and in [8] and [15] the leakage and area have been investigated.

In this paper, we present a novel optimization approach which provides compromise among the DECAP design challenges. Also, two optimum DECAP configurations which provide trades off among area and gate leakage for different applications in nanotechnologies are proposed. The rest of the paper is organized as follows. In Section II, a brief background about DECAPs is provided while in Section III different design challenges are discussed. Section IV contains a method for optimizing DECAP channel length. In Section V, the optimum DECAP configurations are proposed. Finally, summary and conclusions are given in Section VI.

2. Background

A DECAP formed by an NMOS transistor is shown in Fig. 1. This DECAP is modeled as a lumped *RC* circuit, in which *C* determines the charge available and *R* along with determines the transient response of charge delivery to the switching circuit [1]. First order calculations of effective capacitance, and effective channel resistance, at low frequencies are given by [1]

$$C_{eff} = C_{ox}WL + 2C_{ol}W \quad (1)$$

$$R_{eff} = \frac{L}{12\mu C_{ox}W(V_{DD} - V_{TH})} \quad (2)$$

Here, C_{ox} is the oxide capacitance per unit area, C_{ol} is the sum of overlap and fringing capacitances per unit width, μ is the channel mobility, V_{TH} is the threshold voltage, and W and L are the transistor width and length.

DECAPs have traditionally been allocated into the white space available on the die [1][11]. This approach may lead to placing DECAPs at a significant distance from the current load which results in increased power supply noise and oversized capacitors [18]. To be effective, a DECAP should be placed inside the blocks composed of standard cells [1] [11]. In this case, it is more convenient to make DECAPs using both types of

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.37

nMOS and pMOS transistors to form a DECAP cell as shown in Fig. 2. One sample standard-cell DECAP layout is given in Fig. 3 The capacitor areas are the polysilicon gates which are deposited on the top of the channel regions of the MOS transistors [11].

This paper discusses the optimization of the standard-cell DECAPs although the results can be applied to white-space DECAPs too.

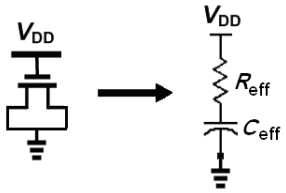


Figure 1. DECAP modeled as an RC circuit [1]

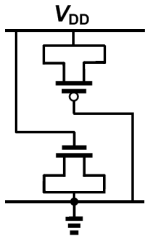


Figure 2. Standard cell DECAP [11]

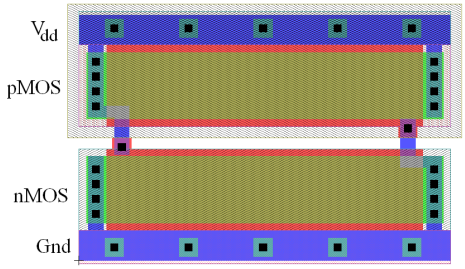


Figure 3. Sample layout of a standard-cell DECAP.

3. Decoupling Capacitor Design Challenges

3.1. Gate Oxide Leakage Current

Gate tunneling current is a critical design issue for DECAPs. The aggressive scaling of T_{ox} results in the presence of significant gate tunneling leakage current, I_{gate} [12]. The current is a strong function of T_{ox} and voltage potential across the gate oxide, V_{OX}, due to its exponential dependence [14]. If V_{OX} is roughly equal to V_{DD}, the leakage current density is the largest [11]. Since in nMOS (pMOS) DECAPs, the gate is tied to V_{DD} (*Gnd*) and the source and drain of the transistor are tied to *Gnd* (V_{DD}), they experience the highest level of the gate leakage [11].

Using BSIM4 model parameters [14] and SPICE simulation, the gate leakage current and power density of nMOS and pMOS transistors in 130nm, 90nm, 45nm and 32nm predictive technology models are obtained and listed in Table 1. As observed from Table 1, the gate leakage current density in 130nm, is small and negligible. As T_{ox} continues to scale, the gate current density, J_{gate}, increases rapidly reaching to its maximum in the 32nm technology which is almost 10 times larger than that of the 130nm one. As a result, the gate leakage power density increases. Since the supply voltage scales down too, the power density increases by about 8 times in the 32nm and 45nm technologies in comparison with the 130nm technology. Also, it is seen that gate current density of pMOS transistors is almost one third of nMOS transistors.

Methods have been suggested in order to reduce the DECAP leakage such as using the thick-oxide MOS transistors [8][11], utilizing high-K gate dielectrics [8][11][13], and gated DECAP technique [15].

In this work, the gate leakage current of DECAPs is minimized using two approaches which are optimizing the DECAP channel length and proposing a new DECAP configuration. These methods may be used in conjunction with the previously proposed methods mentioned above.

Table 1. Gate leakage current and power density for different technologies

Tech.	T_{oxp}* (nm)	V_{DD}	Current Density (μA/μm²)		Power Density (μW/μm²)	
			nMOS	pMOS	nMOS	pMOS
130nm	1.6	1.3	0.14	0.045	0.18	0.06
90nm	1.4	1.2	0.45	0.14	0.54	0.17
45nm	1.1	1	1.26	0.41	1.26	0.41
32nm	1	0.9	1.5	0.49	1.35	0.44

*T_{oxp}= Physical Oxide Thickness

3.2. Transient Time Response Behavior

As the technology scales below 90nm, choosing an appropriate channel length for MOS DECAPs becomes a critical issue. First, considering Eq. (1), it is clear that L should be large in order to have a large capacitance. On the other hand, Eq. (2) suggests that a large L leads to a large channel resistance which degrades the transient response behavior of the DECAPs as described below.

In fact, the MOSFET channel region is similar to a bias-dependent *RC* distributed transmission line [14] which takes a finite time for mobile carriers from the source and drain to form the inversion layer [1]. Thus, the effective capacitance of the MOS transistors is a function of the frequency. At high frequencies, C_{eff} decays since the channel charge cannot respond fast enough to match the gate charge. Thus, in order to achieve the required DECAP budget, more polysilicon gate areas than the one estimated at low frequencies should be used. This leads to two main problems which

are the increases in the chip area and the gate leakage current (static power consumption).

The DECAP transient time response and effective capacitance frequency characteristic were not important issues in earlier technologies [1]. Currently, as f increases into the gigahertz range, it is important to consider these problems, and techniques should be utilized in order to deal with them.

The response time of the channel charge is controlled by the device transit time, which is quadratically related to channel length [1]. The channel length of DECAP, L, controls its frequency response. If L is small enough, the effective capacitance remains relatively constant at high frequencies [11]. Designers use fingering technique in the DECAP layout in order to minimize the effective capacitance reduction at high frequencies. This leads to utilizing multiple parallel small channel length transistors instead of one large one [1][11]. An example of this concept is shown schematically in Fig. 4(a). A long channel length DECAP is implemented with three parallel DECAPs which each channel length is one third of the long channel one. The equivalent lumped RC model of these DECAPs is shown in Fig. 4(b). Using Eqs. (1) and (2), we obtain the low frequency effective capacitance and channel resistance of each finger as one third of those of the long channel one. For three fingers, we obtain

$$Z = \frac{R/3 + \dfrac{1}{(C/3)s}}{3} = R/9 + 1/(Cs) \qquad (3)$$

Therefore, the total low frequency capacitance of three finger configuration is the same as the long channel one while the RC delay of the three finger one is one ninth of the long channel one. The transient characteristic of the DECAP improves at high frequencies in comparison to that of the long channel one.

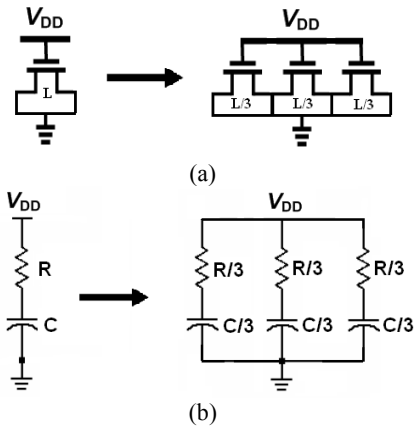


Figure 4. (a) Concept of fingering in decap; (b) equivalent lumped circuit model.

So, considering the transient response, it is desired to implement DECAPs using multiple fingers with the minimum channel length. However, the source and drain contacts of the fingered transistors occupy area which cannot be neglected. A sample layout of Fig. 3 which is implemented with two fingers is shown in Fig. 5. As seen, the added source/drain contact occupies 6λ = $3L_{\min}$ of the standard-cell width using λ-based MOSIS CMOS design rules [17]. If small channel length is chosen for implementing multiple fingered DECAPs, the area overhead due to source/drain contacts is increased, which increases the total chip area. Next, we discuss the optimization of the channel length for implementing multiple finger DECAPs.

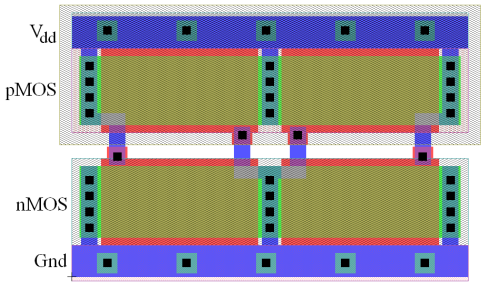


Figure 5. Sample layout of standard-cell decap with two fingers.

4. Optimizing Channel Length of DECAPs

We have used HSPICE simulations to extract the frequency response of the effective capacitance of the MOS DECAPs. HSPICE uses BSIM4 which has the charge-deficit non-quasi-static (NQS) model for simulating high frequency behaviors of MOS transistors [14]. It should be noted that although typical high-speed clock rates today are in the GHz range, it is important to study frequency response well beyond the clock frequency [1][16]. This is due to the fact that most of the spectral power density of digital signals lies within the frequencies up to $f_{knee} = 1/(2t_{rise})$ where t_{rise} is the signal rise time and f_{knee} is the 3dB cut-off frequency of the spectral power density [1]. We assume that the clock frequency is 2GHz with a conservative assumption of 50ps for t_{rise} and the analyses are carried out up to f_{knee} = 10 GHz.

The simulations are performed for 45nm and 32nm technologies with V_{DD} = 1V and 0.9V, respectively. The effective capacitance of nMOS and pMOS decoupling capacitors, with different gate lengths and a fixed width ($W = 9L_{\min}$), is plotted in Fig. 6. As shown, at low frequencies, C_{eff} is constant regardless of the gate length. However as frequency increases, C_{eff} with longer gate lengths begin to decay. The decay is more in the case of pMOS DECAPs. For example, when L = $12L_{min}$ in the 45nm technology, at 10 GHz, the nMOS capacitance drops only by 0.6%, whereas the pMOS capacitance drops by 40%. This implies that nMOS

DECAPs should have longer optimum channel lengths compared to pMOS. Also, for the 32nm technology, since the minimum channel length has been decreased, the equal reduction in C_{eff} occurs in larger coefficients of $L_{\min}$ in comparison with the case of the 45nm technology.

In order to find the optimum channel length, an empirical cost function of implementing DECAPs is defined here as:

$$\psi = \Delta C + \frac{3L_{\min}}{L_{gate}} \tag{4}$$

where ΔC is the effective capacitance reduction at high frequencies. $3L_{\min}$ is the occupied area due to source/drain contacts and L_{gate} is the DECAP channel length, thus ($3L_{min}/L_{gate}$) is the area overhead due to the source/drain contacts.

If short channel length devices are used for implementing multiple fingered DECAPs, ΔC is almost zero while the area overhead due to source/drain contacts is large. Thus, the cost is high. In long channel lengths $3L_{\text{min}}/L_{\text{gate}}$ is negligible but ΔC is high which leads to more polysilicon gate area than the one estimated at low frequency. Thus, the area and gate leakage increases, and the cost becomes high again. Somewhere, between two channel length extremes, the minimum of ψ is obtained.

In order to find the optimum DECAP channel length, ΔC and $3L_{min}/L_{gate}$ are calculated for different channel lengths and the channel length which minimizes the cost function in Eq. (4) is determined as the optimum channel length. Table 2 summarizes the optimum gate lengths for the nMOS and pMOS DECAPs in the 45nm and 32nm technologies at 10GHz. The shaded gate lengths are the optimum channel lengths. As observed, the costs of pMOS DECAPs are higher than those of the nMOS DECAPs. Additionally, advancing into the 32nm technology decreases the cost of implementing DECAPs.

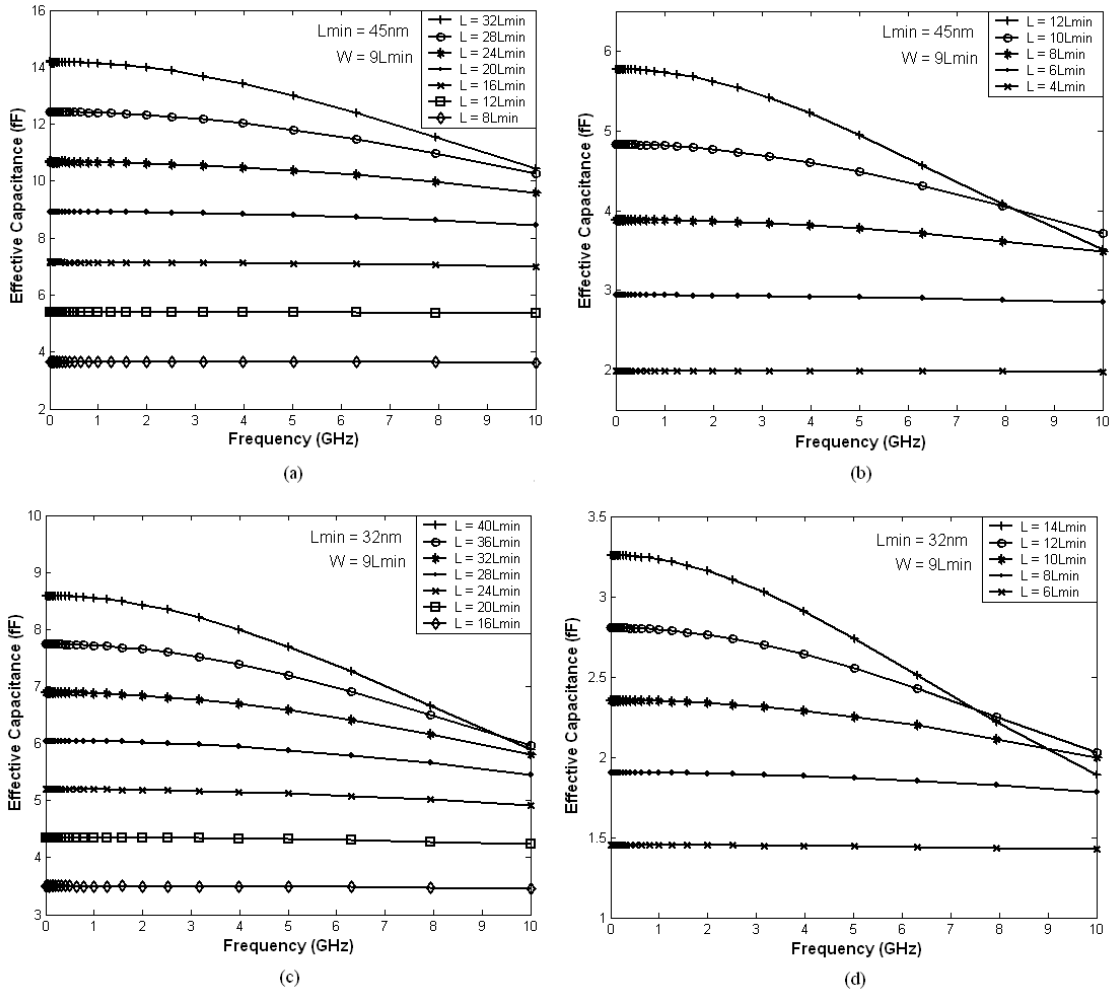


Fig. 6. Effective capacitance frequency response of DECAPs in (a) nMOS (45nm technology), (b) pMOS (45nm technology), (c) nMOS (32nm technology) (d) pMOS (32nm technology)

In order to show that the proposed method for finding the optimum channel length of DECAPs is correct, a 40nf decoupling capacitance is implemented using nMOS DECAPs in the 45nm technology with different channel lengths and the result are shown in Fig. 7. As is observed from the figure, when very short channel lengths are used, the required area is high, thus the cost is high. Increasing the multiple fingered DECAPs gate length, although increases the gate leakage smoothly, however it reduces the area drastically which leads to decrease in cost function until $L = 19L_{min}$ is reached where the cost should is minimized as expected.

The optimum channel lengths for different cut-off frequencies (f_{knee}) are shown in Fig. 8 which shows when the frequency increases, the optimum channel length of the MOS transistor decreases. This decrease is far more pronounced for nMOS transistors. In addition, at a given frequency, the optimum nMOS channel length is more than two times larger than that of the pMOS transistor. Note that the results are different from the conventional belief that the optimum channel length is about 10 times of the minimum channel length [15].

Table 2. The summary of finding optimum channel length of MOS DECAPs at 10GHz

Tech.	L_{gate} ($\times L_{min}$)	ΔC (at 10GHz)	$3L_{min}/L_{gate}$	Cost (Ψ)
45nm nMOS	18	3.3%	16.7%	20%
	19	4.1%	15.8%	19.9%
	20	5.1%	15%	21%
45nm pMOS	7	6%	43%	49%
	8	10.3%	37.5%	47.8%
	9	16%	33.3%	49.3
32nm nMOS	21	3.3%	14.3%	17.6%
	22	3.9%	13.6%	17.5%
	23	4.6%	13%	17.6
32nm pMOS	8	6.3%	37.5%	43.8%
	9	10.1%	33.3%	43.4%
	10	15.1%	30%	45.1%

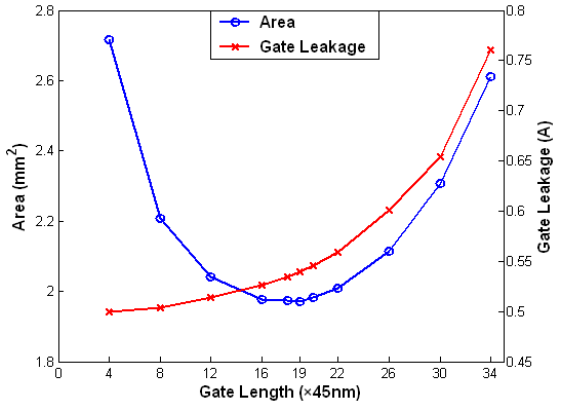


Figure 7. Implementing 40nF DECAP with different gate lengths.

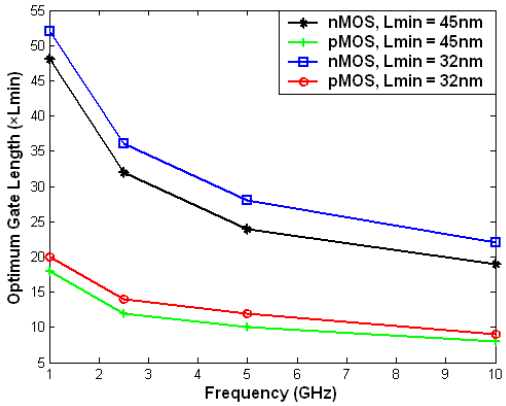


Figure 8. Optimum gate length of MOS DECAPs at different frequencies

5. DECAP Configurations

Three optimum DECAP configurations are shown in Fig. 9. The layouts have been drawn in the 45nm technology with the optimum gate length at 10 GHz. Here, the λ-based CMOS design rules from MOSIS have been used [9]. The DECAP configuration shown in Fig. 9(a) is a conventional standard-cell DECAP which is built by both of nMOS and pMOS transistors. The configuration shown in Fig. 9(b) has been built only with nMOS transistor and based on the results given in Table II, minimizes the required area for implementing the needed DECAP budget. Fig. 9(c) is built using only pMOS transistors and according to the results of Table I, is suggested for low gate leakage current applications.

In order to compare these configurations, the specifications of these three DECAP configurations are listed in Table 3. As expected, the second configuration (only nMOS) has the maximum effective capacitance per area while the third configuration (only pMOS) has the minimum gate leakage current per effective capacitance.

In [1], it is discussed that since the pMOS DECAPs have poor frequency response, it is not suitable for implementing DECAPs with pMOS transistors. However, as the results presented here reveal, the third configuration (only pMOS) outperforms the first configuration (nMOS + pMOS) in terms of both area and leakage concerns. Thus, the first configuration (nMOS + pMOS) has no advantage over the third configuration.

Table 3. Specifications of DECAP Configurations

Config.	C_{eff} (fF)	Area (μm^2)	C_{eff} / Area (fF / μm^2)	I_{gate} (nA)	I_{gate} / C_{eff} (nA / fF)
1st	15.09	1.637	9.22	140.01	9.28
2nd	21.04	1.637	**12.85**	284.22	13.51
3rd	9.023	0.902	10.00	39.55	**4.38**

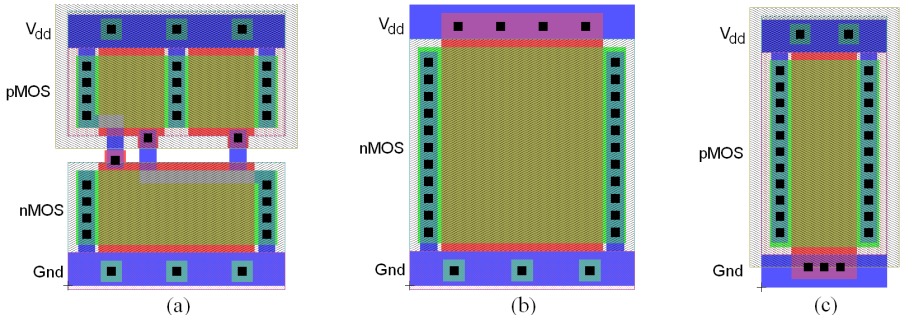


Figure 9. Layouts of three DECAP configurations in 45nm technology with the optimum gate length: (a) 1st, (b) 2nd, and (c) 3rd configuration.

6. Summary and Conclusions

Based on minimizing the cost function of implementing DECAPs, a novel optimization technique for determining the optimum channel length of the MOS DECAPs was presented here. This approach was applied to the 45nm and 32nm technologies, and the results showed that, on contrary to the conventional belief of setting the DECAP channel length to about 10 times the minimum channel length, the optimum channel length is different from this value and should be calculated individually for each technology node and operating frequency. Finally, two optimum DECAP configurations were discussed. One only used nMOS transistors and was suitable for the applications where the area was the main concern. The other was built only by pMOS transistors and was a good candidate for low leakage applications.

7. References

[1] K. Arabi, R. Saleh, X. Meng, "Power Supply Noise in SoCs- Metrics, Management, and Measurement," IEEE Design & Test of Computers, vo. 24, no. 3, pp. 236-244, May-June 2007.

[2] C. Tirumurti et al., "A Modeling Approach for Addressing Power Supply Switching Noise Related Failures of Integrated Circuits," Proc. Design, Automation and Test in Europe Conf. (DATE 04), IEEE CS Press, 2004, pp. 1078-1083.

[3] S. Pant et al., ''Vectorless Analysis of Supply Noise Induced Delay Variation,'' Proc. Int'l Conf. Computer-Aided Design (ICCAD 03), IEEE CS Press, 2003, pp. 184-191.

[4] S. Zhao, K. Roy, and C.-K. Koh, "Decoupling capacitance allocation for power supply noise suppression", in Proc. of International Symposium on Physical Design, pp. 66–71, 2001

[5] International technology roadmap for semiconductors, ITRS, report 2006.

[6] K. Shakeri and J. Meindl, "compact physical IR-drop models for chip/package co-design of gigascale integration (GSI)", *IEEE Trans. On Electron Devices,* 52(6), June 2005.

[7] M. Swaminathan, J. Kim, I. Novak, "Power distribution network for systems-on-package: status and challenges", *IEEE Trans. On Advanced Packaging,* 27(2), May 2004.

[8] J. Fu, Z. Luo, X. Hong, Y. Cai, S. X. –D. Tan, Z. Pan, "VLSI on-chip power/ground network optimization considering decap leakage currents", in proc. of Asia and South Pacific Design Automation Conf., (ASP-DAC 2005), pp. 735 - 738 Jan. 2005.

[9] M. K. Gowan, L.L. Biro and D. B. Jackson, "Power considerations in the design of the Alpha 21264 microprocessor," Proc. of Design Automation Conf., pp. 726-731, Jun. 1998.

[10] http://www.eas.asu.edu/~ptm

[11] X. Meng, "Decoupling Capacitor Design Issues in 90nm CMOS", M.A.Sc Thesis, University of British Columbia, 2006.

[12] D. Lee, D. Blaauw, and D. Sylvester, "Gate oxide leakage current analysis and reduction for VLSI circuits" IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v. 12, no. 2, pp. 155-166, Feb. 2004.

[13] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," in *Proc. IEEE*, vol. 91, no. 2, pp. 305-327, Feb. 2003.

[14] X. Xi, M. Dunga, J. He, W. Liu, K. M. Cao, X. Jin, J. J. Ou, M. Chan, A. M. Niknejad, and C. Hu, "BSIM4.6.1 MOSFET Model User's Manual," University of California, Berkeley, 2007.

[15] Y. Chen, H. Li, K. Roy, and C. -K. Koh, "Gated Decap: Gate Leakage Control of On-Chip Decoupling Capacitors in Scaled Technology," IEEE Custom Integrated Circuits Conference, pp. 775-778, Sep. 2005.

[16] H. Johnson and M. Graham, High-Speed Digital Design, Prentice-Hall, 1993.

[17] N.H.E. Weste and David Harris, CMOS VLSI Design. A Circuit and System perspective. Third Edition, Boston: Addison-Wesley, 2005.

[18] M. Popovich, R. M. Secareanu, E. G. Friedman, and O. L. Hartin, "Efficient. Placement of Distributed On-Chip Decoupling Capacitors in Nanoscale ICs," Computer-Aided Design, IEEE/ACM Int'l Conf. on (ICCAD 2007), pp.811-816 Nov. 2007.

Switched-Capacitor based Buck Converter Design using Current Limiter for better Efficiency and Output Ripple

Tamal Das*, Pradip Mandal, Member, IEEE
Department of Electronics & Electrical Communication Engineering
Indian Institute of Technology-Kharagpur, Kharagpur-721302, India
Email: *tamalfuture@gmail.com, pradip@ece.iitkgp.ernet.in

***Abstract*—In this paper we are addressing power efficiency and output ripple of an embedded switched-capacitor based DC/DC Buck Converters. Here we propose to use current pump based switched-capacitor circuit in buck converter. The current pump circuit limits transition current of the switched-capacitors and hence, improves power efficiency and reduces output ripple. We have also proposed an equivalent macro model of this type of current pump based switched-capacitor converter which would help to get a better essence of the closed loop stability of the system and would reveal clearly trade-offs among load current, flying capacitance and clock frequency. A transistor level implementation of the proposed buck converter in 0.18μ technology is provided. For a load current of 8mA (maximum) the achieved power efficiency is 72.7% and the output ripple is 27mV. The flying capacitors in the converter are 2x108pF and the load capacitor is 125pF.**

I. Introduction

DC-DC buck converter using switched-capacitor circuits is a recent trend for high dropout regulation [1],[6]-[9]. Power efficiency and output ripple of this converter, however, are the concern specifically for embedded applications where sizes of its flying capacitors and load capacitor are limited by chip area. Cascading switched-capacitor with a linear regulator is an approach for the ripple reduction. Using this technique in [1] a fully integrated on-chip regulator is developed. However, switching current in that regulator is quite high. Fig.1(a) shows the basic block diagram of switched capacitor based buck converter. In this circuit the switches (transistors) are voltage controlled and therefore, current through them is not controlled specially during their phase transitions. This phase transition current can be reduced by introducing two current sources between the switch and the rails as shown in Fig.1(b). This is similar to the technique used in [2] & [3] for charge pump based boost converters. Note that the average value of the current source should be determined by the output load current. Fig.2 shows the current flows in its two phases.

Fig.3 shows the difference in the output voltage waveforms for these two topologies. In case of hard switching [Fig.1(a)] since at the time of transition the current is not controlled the output voltage goes to its peak abruptly and it goes down exponentially (approximately linear) till the end of that phase. This pattern repeats in the next phase. This generates a shaw tooth output voltage [Fig.3(a)]. Whereas, for the topology

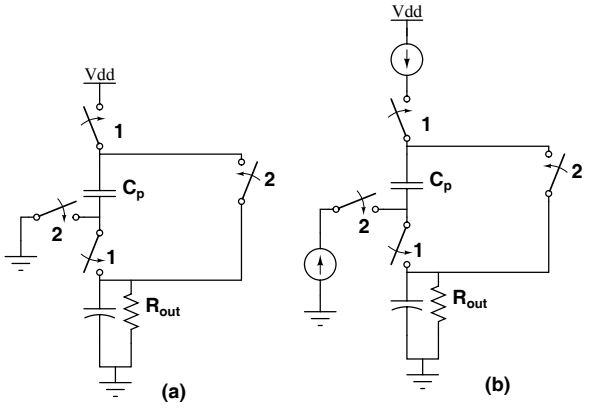


Fig. 1. Switch Capacitor Buck Converter (a)Hard Switched (b) Current Pumped topology

Fig.1(b) the current through the capacitance and the load is controlled by a current constant current source. So, the output voltage effectively remain constant (product of load resistance and the current of that source). Now we can have a new nomenclature called current pump based switching which give us a ideally flat output voltage pumping only the required current through them. But as the output resistance of the current source practically is not infinite and hence we will get some ripple. Fig.4 shows the improvement of the conduction current and also the average current at the transition and during the on time respectively.

In this paper we are proposing a DC/DC Buck converter using current pumped switches which can easily be fully integrable. We are also proposing a technique to get a macro model approximation of the system. The organization of the paper is: in section II working principle of differential mode converter with current pumps; in section III the realization in transistor level of the same is described; in section IV and V the realization of macro model of the proposed buck converter and its closed loop AC behavior are discussed respectively.

In this paper we are proposing a DC/DC Buck converter using current pumped switches which can easily be fully integrable and also we are proposing a technique to get a macro model approximation of the system.The organization

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.43

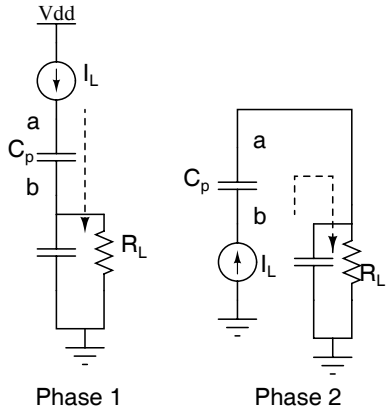


Fig. 2. Current Pumping Concept for Buck Converter

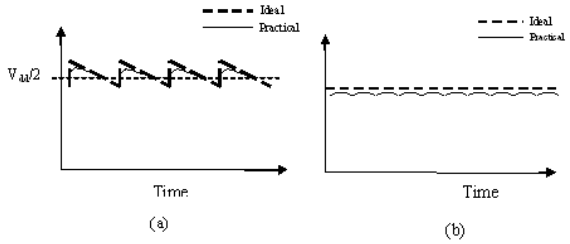


Fig. 3. Output Voltage Waveform for (a)Hard Switching (b) Current Pumping

of the paper will be: in section II working principle of differential mode converter with current pumps; in section III the realization in transistor level of the same is described; in section IV and V the realization of macro model of the proposed buck converter and its closed loop AC behavior are discussed respectively.

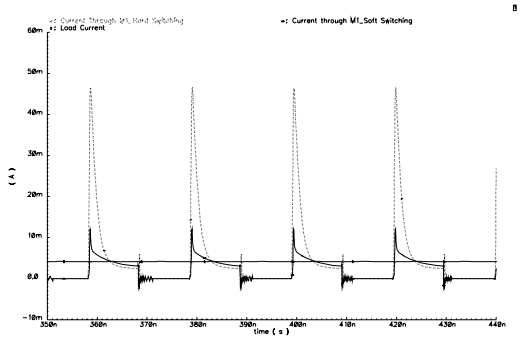


Fig. 4. Comparison of Current through switches for Current Pumping and Hard Switching

II. Working Principle of Differential Mode Current Pump Buck Converter

It is obvious that to get high output load current we need to operate the converter at higher frequency. Now from the Fig.1(a) it is seen that in each phase any one of the current source must be switched off or that current has to be bypassed to another path. In first case it may lead to phase switch over problem and it is needless to say that the second case leads to unnecessary power consumption. Hence only solution is to opt for dual mode current pumping technique i.e. use another same switched-capacitor converter driven by 180^0 out of phase clock signal. Fig.5 shows the differential version.

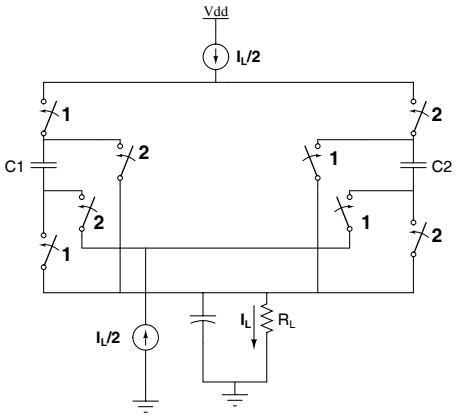


Fig. 5. Current Pumping in Differential Mode

Fig.6 shows the circuit conditions of differential mode current-pumped switching in two different phases; it shows that the supply current by the current sources is reduced by 50%.

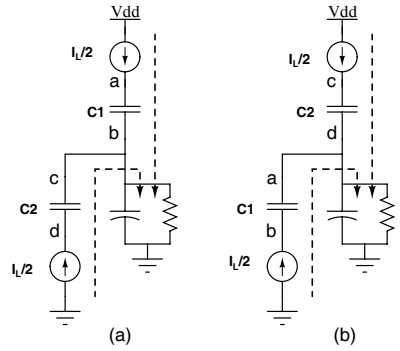


Fig. 6. Differential Mode Current Pumping in Two Phases

III. Realization of Current Pump Buck converter

Now Fig.6 shows the transistor level implementation of proposed topology. Transistors M0 and M1 are acting like current source. Transistors M2 and M6 are acting like pMOS current-pumped switches while transistors M3 and M7 are acting like nMOS current pumped switches. The main challenge here is to

pull the current from ground and implement the nMOS current pumped switches. When the transistor M3 (M7) will become on at that very moment the voltage at node E(E′) becomes negative ,and in each half cycle either of the C_p capacitors works as voltage source; taking these opportunities we can easily connect the source of the nMOS pass transistor to that terminal and the drain of it can be grounded and therefore, the current mirror realization to supply the gate voltage of this pass transistor is obvious. Now for good load regulation we can have a feedback path which will sense the output voltage or required load current and will monitor the gate voltage of the M0 and M1 transistors.

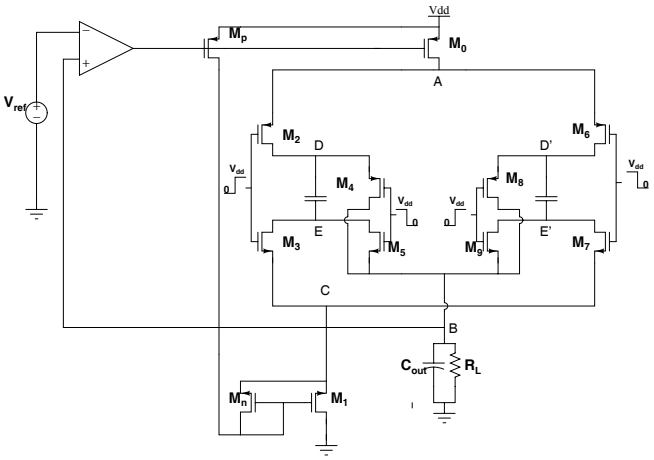


Fig. 7. Proposed Topology of Current Pumped Differential Mode DC/DC Buck converter

IV. Macro Model of the Converter

In [9] the necessity of the dynamic behavior and AC response analysis are how important from the application point of view is addressed. Here for the proposed topology we have also analysed for the same.

The performance and stability of closed loop system can only be described analytically by its equivalent macro model. To get the macro model from the above current pump circuit one should analyze all node voltages at the steady state of the operation. It will be shown that the difference between the average voltage at the drain of M0 and bottom plate of C_p during the on cycle of M2 (M6) and M5 (M9) is V_{dd}/2.

Fig.8 shows some typical waveforms which help to find the average output voltage and the difference between average voltages at node A and B which will be needed to model the flying capacitor C_p which needs to be modeled to get a dc operating point for the macro model.

Let us take a common assumption that if any voltage level is defined with respect to $V_p{}^-$ then it will be designated with an extra prime(′) mark. Now it is obvious that

$$(\Delta V)_A = (\Delta V)_C = (\Delta V)(say) \tag{1}$$

$$Now, (V_{A1} - V_{A2}) = (V_{B1} - V_{B2}) = (\Delta V)_B \tag{2}$$

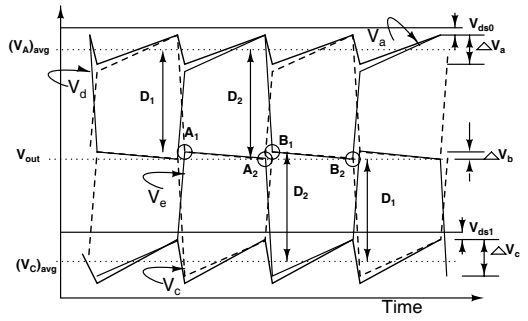


Fig. 8. Waveform for average output voltage calculation

$$D1 = V_{B1}^{'} = (V_P^+)^{'} - V_{A2}^{'}$$
$$D2 = V_{B2}^{'} + \frac{(\Delta V)_C}{2} = (V_P^+),- V_{A1}^{'} + \frac{(\Delta V)_A}{2}$$
$$\Rightarrow V_{B2}^{'} = (V_P^+)^{'} - V_{A1}^{'}$$

$$Therefore, (V_{B1}^{'} + V_{A2}^{'}) = (V_{B2}^{'} + V_{A1}^{'}) = (V_P^+)^{'} \tag{3}$$

$$\begin{aligned} \Rightarrow (V_{B1} + V_{A2}) &= (V_{B2} + V_{A1}) \\ &= (V_P^+)^{'} - 2[V_{ds1} + \frac{(\Delta V)_C}{2}] \\ &= (V_P^+) - [V_{ds1} + \frac{(\Delta V)_C}{2}] \\ &= V_{dd} - |V_{ds0}| - V_{ds1} - (\Delta V) \end{aligned} \tag{4}$$

Therefore, from Eq.2 and Eq.4 and as V_{A1}=V_{B1} and V_{A2}=V_{B2},

$$\begin{aligned} \frac{(V_{A1} + V_{A2})}{2} &= \frac{(V_{B1} + V_{B2})}{2} \\ &= \frac{V_{dd} - |V_{ds0}| - V_{ds1} - (\Delta V)}{2} = V_{out} \end{aligned} \tag{5}$$

Now again

$$\begin{aligned} (V_A)_{avg} - V_{out} &= (V_{dd} - V_{ds0} - \frac{(\Delta V)_A}{2}) \\ &\quad - \frac{V_{dd} - |V_{ds0}| - V_{ds1} - (\Delta V)}{2} \\ &= \frac{V_{dd} - \quad (|V_{ds0}| - V_{ds1})}{2} \end{aligned} \tag{6}$$

If $|V_{ds0}|$=V_{ds1}, the difference between these two average voltage level will become V_{dd}/2; and we can take it as V_{dd}/2 for our macro model.

Fig.5 shows the circuit condition during any half cycle. To linearize the model one should replace the flying capacitors C_p by some combination of any linear component. From the Fig.8 we can easily have the macro model as shown in Fig.9. The choice of second DC source as V_{dd}/2 between B and C′ is some what arbitrary and hence to meet the average voltage at the source of the pass transistor M1 we have connected a resistor R′ and which may be negative resistance whose value depends on the relative value of V_{ds1} and V_{ds2} .

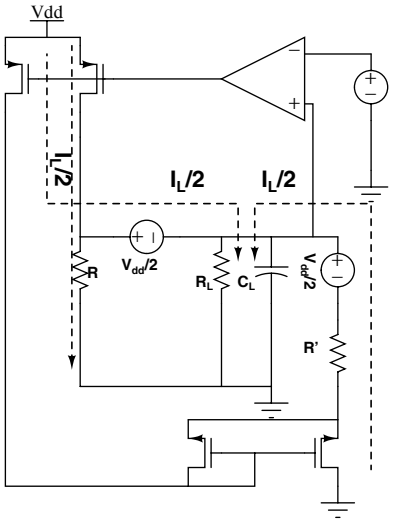


Fig. 9. Macro Model Of Proposed Topology

A. Calculation of Resistance R and R′:

From the Fig.8 it is seen that at the drain of M0 an average voltage is maintained at the steady state which is modelled as resistor R. From that average voltage and the maximum output required current we can easily calculate the value R in two different ways: from the definition of $(\delta \mathrm{V})_{avg}$ and another is described as follows. To get a better essence switched capacitor circuit we have tried to find the relationship between switch capacitor resistor($1/\mathrm{fc}_p$) and R.

Firstly let, (I_{sat2}) = $(\mathrm{I}_{sat})_6$ = I_{sat}.

$$\begin{aligned} Now, (V_A)_{avg} &= 2(\Delta V)^{'} - (\frac{\Delta V}{2}) - V_{ds1} \\ &= 2I_{sat}(\frac{t_1}{C_p}) - \frac{1}{2}(\frac{I_L}{2}\frac{t_2}{C_p}) - V_{ds1} \\ &= \frac{I_L}{2}[\frac{1}{fC_p}(2\frac{I_{sat}}{I_L}\frac{t_1}{t_2} - \frac{1}{2}) - \frac{2v_{ds1}}{I_L}] \\ &= \frac{I_L}{2}[\frac{1}{fC_p}\{(\frac{\Delta v^{'}}{\Delta v}) - \frac{1}{4}\} - \frac{2V_{ds1}}{I_L}] \end{aligned} \tag{7}$$

Therefore, the resistor R can be modelled as

$$R = \frac{1}{fC_p}\{(\frac{\Delta v^{'}}{\Delta v}) - \frac{1}{4}\} - \frac{2V_{ds1}}{I_L} \tag{8}$$

It can be shown that the resistance depends on either minimum V_{ds1} or minimum V_{ds0};last equation shows the dependency on V_{ds1}.Now the value of R′ is some what depends on V_{dd} as the value for 2^{nd} DC source is taken as V_{dd}/2 arbitrarily. Therefore to model the $(\Delta \mathrm{V}_C)_{avg}$ we affix the resistance there and hence it can be calculated as follows Since after subtracting V_{dd}/2 we Will get $(\mathrm{V}_C)_{avg}$ otherwise we may get another average voltage level say $(\mathrm{V}_{C}{}')_{avg}$. From the difference of these two average voltage level we can get the value of R′ and this is given by as follows:

$$(V_{C'})avg = (V_B)_{avg} - \frac{V_{dd}}{2} \tag{9}$$

$$again, (V_{C'})avg + \frac{I_L R^{'}}{2} = (V_C)_{avg} \tag{10}$$

$$\Rightarrow (V_B)_{avg} - \frac{V_{dd}}{2} + \frac{I_L R^{'}}{2} = V_{s1} + \frac{(\Delta V)_C}{2}$$

$$from Eq.5$$

$$\Rightarrow \frac{I_L R^{'}}{2} = \frac{(|V_{ds0}| - V_{ds1})}{2}$$

Therefore the resistance R′ can be modelled for maximum specified load current as

$$R^{'} = \frac{(|V_{ds0}| - V_{ds1})}{I_L} \tag{11}$$

And this value is very close to zero.

Now the main thing is to see whether this model will work for lower load than maximum specified load current or not. Now for no load current the gate voltage of the pMOS pass transistor will be so fixed that it will supply only half load current and the nMOS pass transistor will not supply any current as drain-source voltage will become negligible.For in between these two extreme cases the operation of pMOs pass transistor and effect of R can be visualized easily by seeing the upper closed loop path. Now for that cases operation of nMOS will be determined by not only the gate voltage but the source voltage and hence voltage drop due to R′ also. For lower current the gate voltage will decrease and source voltage also putting drain voltage equal to 0.

B. Properties of Pass Transistors and Error Amplifier

As it was already said that success of the model and hence prepared design flow of the whole system from it depends on the error amplifier performance and not only that the length of the pass transistors are also claims some attention as the output voltage level is related to them through average voltage at there drain and source respectively. For full load the gate voltage at the pMOS pass transistor will be minimum while for no load condition this will be maximum. Therefore the depending on the minimum output voltage of error amplifier will fix the size of the pMOS and again the output voltage swing of the error amplifier should support the required gate voltage range of pMOS. A clear relationship between the minimum and maximum gate voltage required for Popper closed loop control over full range of load current. For nMOS transistor control over the current through depends more strongly on its source voltage. Now

$$I_L = K_p^{\prime}(\frac{W}{L})[V_{gs0min} - |v_{thp}|]^2(1 + \lambda|V_{ds}|) \tag{12}$$

$$\frac{I_L}{2} = K_p^{\prime}(\frac{W}{L})[V_{gs0max} - |v_{thp}|]^2(1 + \lambda|V_{ds}|) \tag{13}$$

And they are related as

$$V_{g0max} = (\frac{\sqrt{2}-1}{\sqrt{2}})(V_{DD} - |v_{thp}|) + \frac{1}{\sqrt{2}}V_{g0min} \tag{14}$$

For ripple reduction and for good load regulation the gain of the amplifier also should be high with acceptable UGB

as we need lower settling time also. To meet these two requirements the error amplifier should be cascode amplifier type. As telescopic cascode though has high gain its short output voltage swing restricts its usage. The folded cascode differential amplifier is most suitable for this operation.

V. AC Analysis of the Macro Model

Fig. 10 shows high frequency small signal model for the macro model of the proposed topology considering macro model for the error amplifier also. Here the current through the current mirror (M_p and M_n) is neglected and hence the voltage at the gate of the M1 is just extracted to eliminate them to avoid a cumbersome AC analysis. Here the error amplifier is taken as an macro model having transconductance g_{mA} and output resistance R_{outA}. Now the open loop voltage gain from v_{ref} to the v_b can be found using super position principle for two current sources.The current mirror part of the lower nMOS pass transistor can be replaced by an equivalent small gate-source voltage for that nMOS and that will be given by:

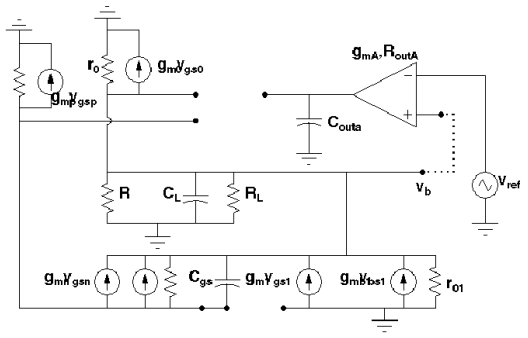


Fig. 10. Small Signal Equivalent of the Macro Model

$$v_{gs1} = (\frac{g_{mA}g_{mp}R_{oA}}{g_{mn}})(\frac{1}{1+sC_{outA}R_{oA}}) \tag{15}$$

Now the open loop gain will be given by:

$$\frac{v_b}{v_{ref}} = \frac{g_{mA}g_{m0}R_{oA}}{1+sC_{outA}R_{oA}}[\mathcal{Z}||(\frac{1}{sc_{gs1}+(g_{m1}+g_{mb1)}})] + \frac{g_{mA}g_{mp}R_{oA}\mathcal{Z}}{g_{mn}(1+sC_{outA}R_{oA})}[\frac{sc_{gs1}r_{o1}+g_{m1}r_{o1}}{\mathcal{Z}+r_{o1}}] \tag{16}$$

$$where, \mathcal{Z} = r_o||R||R_L||\frac{1}{s(C_L+C_{in})} = R_{eff}||\frac{1}{s(C_L+C_{in})}$$

which will be giving the following transfer function:

$$A_v = (g_{mA}R_{outA})(\frac{g_{mp}}{g_{mn}})(\frac{C_{gs1}}{C_L+C_{in}}) [\frac{s^2+\frac{Y}{X}s+\frac{Z}{X}}{(s+p_1)(s+p_2)(s+p_3)}] \tag{17}$$

where,

$$p_1 = -\frac{1}{C_{outA}R_{oA}}$$
$$p_2 = -\frac{1+R_{eff}(g_{m1}+g_{mb1})}{R_{eff}(C_L+C_{in})} \simeq p_3 \tag{18}$$

$$X = R_{eff}(C_L+C_{in})C_{gs1}(\frac{g_{mp}}{g_{mn}})$$
$$Y \simeq R_{eff}(C_L+C_{in})(g_{m0}+\frac{g_{mp}}{g_{mn}}g_{m1}) \tag{19}$$
$$Z \simeq 2\{R_{eff}(g_{m1}+gmb1)+1\}g_{m0}$$

It is seen that the zeros are very far to create any disturbance to the closed loop performance of the system. Now it is also seen that if the maximum specified current is increased the stability of the system would also increase as the poles (p_2 & p_3)are gone away from the dominant pole p_1 whose shifting can be neglected.

Now for DC-DC converters it is well known that AC analysis does not confirm its stability but the macro model what we have proposed is giving the essence of stability issues at low time scale.

VI. Simulation Results

We have designed a current pump based DC-DC buck converter with 0.18μ CMOS process. For the closed loop application the necessary error amplifier is also designed with 55dB DC gain with 100MHz UGB using folded cascode topology. Table 1 shows the performance comparison between the hard switched and current pump based converter and points to the advantages. Without closed loop control output voltage would be V_{DD}/2. For closed loop control we used V_{ref}=1.35^V.

TABLE I
Performance Comparison Between Hard switched and Current Pumped Topology

Load Current (mA)	η(%)		Output Ripple (mV)	
	Hard switching	current pump	Hard switching	current pump
7.99	64	72.7	100	27

Fig.11 shows typical waveforms at different nodes at 8^{mA} Load current which is supporting our qualitative view (Fig.8).

Fig.12 shows the dynamic behavior of the converter demonstrating its good load regulation.

Table 2 shows the strength of the macro model of the design. We have targeted a load current and from the macro model

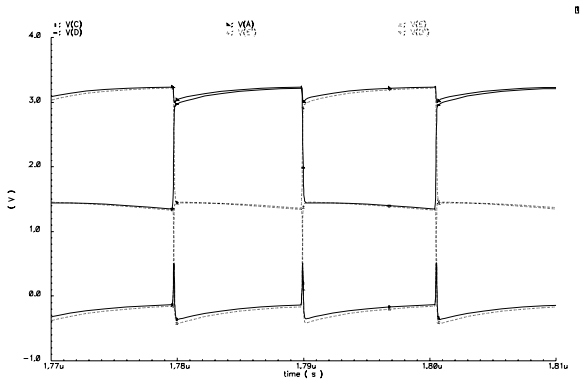


Fig. 11. Typical Waveforms from the actual Circuit at 8_{mA} Load Current

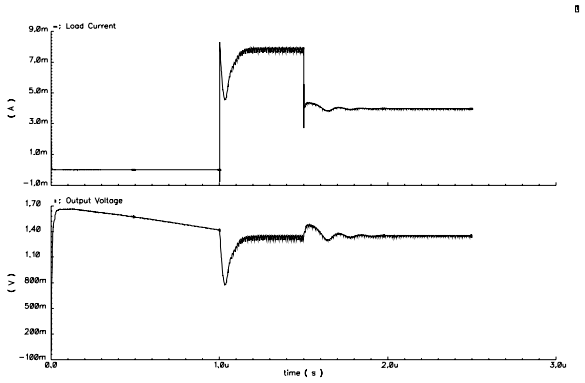


Fig. 12. Dynamic Behavior of the Converter for 4 to 8^{mA} Load Current Step

TABLE II
COMPONENT PARAMETERS OF MACRO MODEL AND ACTUAL TOPOLOGY

Maximum Load Current(mA)		C_p(F)		Output Voltage(V)	
Macro Model	Actual Topology	Macro Model	Actual Topology	Macro Model	Actual Topology
4.00	4.00	50p	53p	1.35	1.35
6.00	5.99	75p	78p	1.35	1.3499
8.00	7.99	100p	108p	1.35	1.34999

TABLE III
PERFORMANCE OF CURRENT PUMPED TOPOLOGY

Frequency (MHz)	Load Current (mA)	η (%)	Output Ripple (mV)
50	8	72.7	27
	7	69.8	25
	6	64.6	20
	5	61.0	16
	4	54.0	14
	3	50.0	12
	2	37.0	10

we got the R and R' and from that C_f. The table shows the accuracy of the model. The size of flying capacitances shows its integratibility.

Table 3 shows the performance of the converter for 8^{mA} as maximum load current from the efficiency and output ripple point of view designed.

VII. SUMMARY AND CONCLUSION

We have proposed to use current pump based switched-capacitor circuit for dc-dc buck converter. The current pump circuit helps to reduce switching current during phase transition of the switch capacitors. This improves overall power efficiency of the converter and reduces output ripple of the converter. Stability and design trade offs of the converter has been analysed with its macro model and small signal equivalent circuits. Finally, a transistor level implementation and its simulated performance of the converter are provided. In the implementation the achieved power efficiency is 72.7% (i.e. 8% more than conventional one) and the output ripple is 27mV (i.e. reduced by a factor of 3.7).

REFERENCES

[1] P. Mandal, K. Bhattacharya, "A Low Voltage,Low Ripple On Chip Hybrid DC-DC Converter," *International Symposium on Integrated Circuits*, 2007.

[2] Soon-Kyun Shin et.al, "A High Current Driving Charge Pump with Current Regulation Method" *IEEE Custom Integrated Circuits Conference(CICC)*, 2005.

[3] A. Cabrini, A. Fantini and G. Torelli, "High-efficiency Regulator for on-chip charge pump voltage elevators" *Electronics Letters*, vol. 42, No. 17, 2006.

[4] Jaroslav Dudrik, Juraj Oetter, "High-Frequency Soft-Switching DC-DC Converters for Voltage and Current DC Power Sources" *Acta Polytechnica Hungarica*, Vol. 4, No. 2, 2007.

[5] Rajapandian Ayyanar, Ned Mohan, "A Novel Soft-Switching DC-DC Converter with Wide ZVS-Range and Reduced Filter Requirement" *Power Electronics Specialists Conference*, July 1999 Page(s):433 - 438 vol.

[6] Manal H. Hashem Et. Al. , Switched Capacitor Snubber-Assisted Zero Current Soft Switching PWM High Frequency Inverter with Two-Lossless Inductive Snubbers" *IEEE PEDS*, 2005.

[7] Mummadi Veerachary, "Control of Switched Capacitor Step-Down Buck Converter" *IECON - 32nd Annual Conference*, 2006.

[8] Kohei Onizuka, Hiroshi Kawaguchi, Makoto Takamiya and Takayasu Sakurai, "Stacked-chip Implementation of On-Chip Buck Converter for Power-Aware Distributed Power Supply Systems" *Solid-State Circuits ConferenceIEEE Asian* , 2006.

[9] A. Barrado, A. Lzaro, J. Pleite, R. Vzquez, J. Vzquez, E. Olas, "Linear-Non-Linear Control (LnLc) for DC-DC Buck Converters: Stability and Transient Response Analysis." *Applied Power Electronics Conference and Exposition*, 2004, Page(s):1329 - 1335 vol.2.

Session 3B

System Synthesis

2009 22nd International Conference on VLSI Design

Reversible Logic Synthesis with Output Permutation

Robert Wille[1] Daniel Große[1] Gerhard W. Dueck[2] Rolf Drechsler[1]

[1]Institute of Computer Science
University of Bremen
28359 Bremen, Germany

[2]Faculty of Computer Science
University of New Brunswick
Fredericton, Canada

{rwille,grosse,drechsle}@informatik.uni-bremen.de
gdueck@unb.ca

Abstract

Synthesis of reversible logic has become a very important research area. In recent years several algorithms – heuristic as well as exact ones – have been introduced in this area. Typically, they use the specification of a reversible function in terms of a truth table as input. Here, the position of the outputs are fixed. However, in general it is irrelevant, how the respective outputs are ordered. Thus, a synthesis methodology is proposed that determines for a given reversible function an equivalent circuit realization modulo output permutation. More precisely, the result of the synthesis process is a circuit realization whose output functions have been permuted in comparison to the original specification and *the respective permutation vector. We show that this synthesis methodology may lead to significant smaller realizations. We apply* Synthesis with Output Permutation *(SWOP) to both, an exact and a heuristic synthesis algorithm. As our experiments show using the new synthesis paradigm leads to multiple control Toffoli networks that are smaller than the currently best known realizations.*

1. Introduction

According to *Moore's Law* the number of transistors in an integrated circuit doubles every 18 months. Due to this exponential growth, physical boundaries will be reached in the near future. Furthermore, power consumption of circuits becomes a major issue. Quantum computers [12] are an alternative to classical systems. Here, information is stored in so called qubits instead of bits. In comparison to present computers, many problems can be handled more efficiently with the help of quantum computers.

Since all quantum computations are reversible, the synthesis of reversible logic has become an intensely studied topic. In contrast to classical irreversible gates, there are restrictions for reversible gates, e.g. fan-out and feed-back are not allowed. Consequently a network for reversible logic consists of a cascade of reversible gates. In the past different types of reversible gates have been introduced, e.g. (multiple control) Toffoli [18] and Fredkin [2] gates, Peres gates [13], and elementary quantum gates [1].

For the synthesis of reversible logic several approaches – heuristic as well as exact ones – have been proposed. A method based on enumeration that uses network equivalences to rewrite a limited set of gates has been presented in [16]. Proposed heuristics methods are based on spectral techniques [10], positive polarity Reed-Muller expansions [5], or transformation based synthesis [11]. In [9] a method is introduced that synthesizes the reversible function in a first step and then based on transformations (using so called templates) a realization with fewer gates is computed. Techniques of group theory can also be used in the synthesis of reversible logic functions [17]. The authors of [15] introduced a non-search based algorithm running transformations to synthesize reversible functions with CNOT gates. Minimal networks for functions with up to three variables have been synthesized by the approach introduced in [23]. An exact synthesis method based on reachability analysis is described in [6]. In [3, 4, 20] approaches based on *Boolean satisfiability* (SAT) and in [22] a method employing *Quantified Boolean Formula* (QBF) satisfiability are used for exact synthesis.

Usually, the specification of the reversible function to be synthesized is given as a truth table. Thus, each output is set to a fixed position. Since in general the output ordering for a given reversible function f is irrelevant, we propose a synthesis methodology that determines an equivalent circuit realization for f modulo output permutation. That is, the result of the synthesis is a circuit whose outputs have been permuted. Note that no extra gates are invested to achieve the output permutation. In fact, output permutation becomes an integral part of the synthesis process such that the final permutation corresponds to an "update" of reversible function specification. Hence, the synthesis result is a circuit realization *and* the computed output permutation vector.

Based on this idea we introduce first algorithms to apply *Synthesis with Output Permutation* (SWOP). The algorithms focus on synthesis of multiple control Toffoli networks. As the main objective the number of gates is minimized as done by many other researchers (see e.g. [5, 9, 11, 15, 16, 20]). The proposed methodology can also be adapted for other gate libraries as well as other objectives.

The application of output permutations has been recognized before in [11]. It was suggested that for functions with few input variables, all output permutations could be considered. However, neither an analysis of the effect of output permutation nor techniques facing the increasing complexity in case of larger circuits have been considered. This work is an initiative to address this missing domain.

To find the best permutation of outputs for a function,

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.40

Table 1. Function specification

c	b	a	o_3	o_2	o_1
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	0

i.e. the one which leads to the smallest network realization, all $n!$ permutations have to be checked in general (where n is the number of variables of the reversible function). We show how this complexity can be reduced for incompletely specified function (i.e. functions with garbage outputs). Furthermore, we present an exact and a heuristic approach applying synthesis with output permutation. We show that significantly smaller networks (even smaller than the ones known as minimal till today) can be obtained if this new synthesis paradigm is used.

The paper is structured as follows. First, preliminaries are given in Section 2. Section 3 describes the general idea of SWOP while Section 4 gives some theoretical consideration. We introduce an exact and heuristic synthesis algorithm which applies output permutation in Section 5. Results are given and discussed in Section 6. Finally, we conclude the paper and give directions for future work in the last section.

2. Preliminaries

To keep the paper self-contained, this section briefly reviews the basics of reversible logic. For a more detailed insight we refer to the respective publications.

A reversible logic gate realizes an n-input n-output function that maps each possible input vector to a unique output vector. In other words this function is a bijection. Many reversible gates have been studied. Multiple control Toffoli gates [18] (also known as generalized Toffoli gates) are widely used. In the rest of this paper we only consider Toffoli gates that are defined as follows:

Definition 1 *Let $X := \{x_1, \ldots, x_n\}$ be the set of domain variables. A* multiple control Toffoli gate *has the form $TOF(C, t)$, where $C = \{x_{i_1}, \ldots, x_{i_k}\} \subset X$ is the set of control lines and $t = \{x_j\}$ with $C \cap t = \emptyset$ is the target line. The gate maps $(x_1, \ldots, x_n)$ to $(x_1, \ldots, x_{j-1}, x_j \oplus x_{i_1} \ldots x_{i_k}, x_{j+1}, \ldots, x_n)$. If no control lines are given (C is empty), then the target line is inverted, i.e. the input vector of the gate is mapped to $(x_1, \ldots, x_{j-1}, x_j \oplus 1, x_{j+1}, \ldots, x_n)$.*

Due to restrictions in quantum mechanics the only possible topology for a network is a cascade of gates.

Definition 2 *The* cost *of a reversible network is defined as the number of its gates.*

Note that, as an additional quality criterion for reversible logic also *Quantum Costs* [1] are used in literature. However, in this work we aim to minimize the number of gates which is done by many other researches as well (see also introduction).

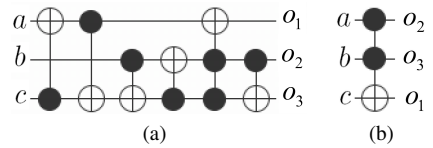


Figure 1. Minimal Toffoli networks

Since all quantum circuits are reversible, to realize a non-reversible function (i.e. an n-input m-output function with $n > m$) it must be embedded into a reversible one. Therefore, it is often necessary to add constant inputs and garbage outputs [8]. The garbage outputs are by definition don't cares and can be left unspecified. Functions with garbage outputs are called *incompletely specified functions* in the following.

3. General Idea

The input of most synthesis approaches is the specification of the reversible function $f : \mathbb{B}^n \to \mathbb{B}^n$ to be synthesized as a truth table. In this table each specified output has a fixed position.

Example 1 *Consider the function specification shown in Table 1. The reversible function maps (c, b, a) to $(b, a, ab \oplus c) = (o_3, o_2, o_1)$. A minimal Toffoli network for this function is shown in Figure 1(a). The cost of this network is* 6.

If the synthesis approach follows the proposed methodology the synthesis result for the given reversible specification is an equivalent circuit realization whose outputs have been permuted.

Example 2 *In Figure 1(b) a Toffoli network is depicted which computes the same reversible function than the Toffoli network shown in Figure 1(a). But in contrast, the three output functions have been "reordered" to another position in the output vector. More precisely, the Toffoli network shown in Figure 1(b) maps (c, b, a) to $(ab \oplus c, b, a) = (o_1, o_3, o_2)$. This reduces the overall costs from* 6 *gates to a single gate, i.e.* 5 *gates have been saved. In total, the result of a synthesis procedure for this example would be the network shown in Figure 1(b)* and *the new output vector (o_1, o_3, o_2).*

Motivated by this example, the question considered in this paper is:

> *How can we efficiently compute good permutations of outputs for a given reversible function to be synthesized such that smaller Toffoli networks result?*

This leads to an extension of common synthesis algorithms we call *Synthesis with Output Permutation* (SWOP) in the rest of this paper. As shown in the following, SWOP may lead to significantly smaller circuits regardless of whether exact or heuristics approaches are used. Since the consideration of all permutations can be expensive with respect to runtime, we propose different strategies which handle the increasing complexity.

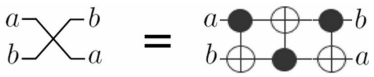


Figure 2. Realization of a permutation

4. Theoretical Consideration

In this section we show the best case benefit that can be achieved by applying SWOP. Therefore, we compare common synthesis to SWOP and determine the maximal number of gates that can be saved if we allow output permutation for an arbitrary reversible function specification. Furthermore, we discuss the worst case complexity to determine the best permutation and show how this can be reduced for incompletely specified functions by exploiting the information on garbage outputs.

4.1. Best Case Benefit

Figure 2 depicts the gates needed to permute two signals in a reversible circuit with multiple control Toffoli gates (in total three gates are required). Since the best position of the outputs is unknown at the beginning of the synthesis process, outputs may be placed arbitrarily in the function specification. Then, the three gates of Figure 2 are needed to permute the value of a signal to the position given by the specification. If in contrast output permutation is considered during the synthesis, the number of gates of the resulting network may be significantly smaller as the following proposition shows.

Proposition 1 *The number of gates in a reversible circuit obtained by common synthesis approaches may be up to* $3 \cdot (n-1)$ *higher than the number of gates in a circuit where synthesis with output permutation is applied (with* n *is the number of variables).*

Proof: Let c be the minimal costs of a circuit obtained by enabling output permutation during synthesis. To move one output line to the position given by the specification three Toffoli gates are required (see Figure 2). At most $n-1$ lines need to be moved. It follows that the cost of the minimal circuit, where no output permutation is allowed, is less than or equal to $c + 3(n-1)$. ■

4.2. Complexity

Finding the best output permutation causes a significant increase in complexity for synthesis. In general, all possible permutations have to be checked, which results in $n!$ different networks in total.

However, it is well known that many practical logic functions contain garbage outputs (see Section 2). The garbage outputs are by definition don't cares and can be left unspecified. Thus, permutations of the garbage outputs need not be considered. This reduces the complexity for SWOP. Instead of $n!$ only $\frac{n!}{g!}$ different permutations are checked (while n is number of variables and g the number of garbage outputs of the reversible function f).

Example 3 *Figure 3 shows all* $n!$ *possible permutations for an incompletely specified function with* $n = 3$ *variables and* $g = 2$ *garbage outputs (denoted by* g_1 *and* g_2*). Since the garbage outputs are left unspecified, the permutations that only swap garbage outputs can be skipped (i.e. the last three permutations of Figure 3). Thus, only* $\frac{3!}{2!} = 3$ *permutations instead of all* $3! = 6$ *permutations are considered.*

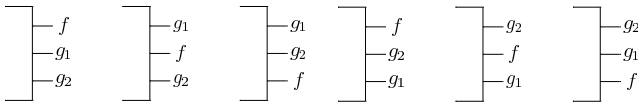


Figure 3. Permutations with garbage outputs

5. Applying Output Permutation

In a naive way, synthesis with output permutation can be easily applied to existing approaches just by encoding all permutations, synthesize each in one turn, and keep the best one. This results in an increase of factor $\frac{n!}{g!}$. In this section we introduce the application of output permutation to exact as well as heuristic approaches using dedicated strategies. Empirical tests show that the increase of the runtime by the proposed approaches is less than the theoretical complexity increase. This is due to the learning technique exploited in the exact approach and due to the heuristic selection of permutations in the heuristic approach.

5.1. Exact Approach

Exact synthesis algorithms determine a *minimal* realization for a given function, i.e. a network with the minimal number of gates. Ensuring minimality is obviously more expensive, but helps e.g. to synthesize smaller networks (or compositions of networks) and to define lower bounds for heuristic approaches. Thus, research in this area is essential.

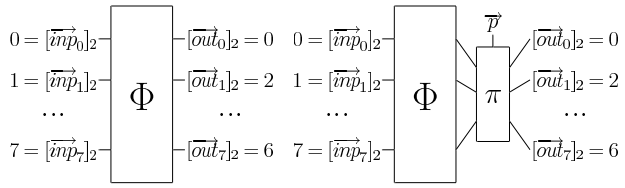
Recently exact algorithms for synthesis of multiple control Toffoli gates using *Boolean satisfiability* (SAT) have been introduced [3, 20]. The basic idea is to check if there exists a Toffoli network representation for a reversible function with c gates (starting with $c = 1$), where c is increased in each iteration if no realization is found. The respective checks are performed by representing the problem as an instance of SAT. This instance is solved by a common SAT solver [3] or by the specialized solve-engine *SWORD*, which additionally uses problem specific knowledge [19, 20]. Due to page limitation we refer to the respective publications for a detailed description of the encodings. In this paper the concrete SAT encoding is simplified as follows:

Definition 3 *Let* $f : \mathbb{B}^n \rightarrow \mathbb{B}^n$ *be a reversible function to be synthesized. Then, the SAT instance of the respective synthesis problem is given as*

$$\Phi \wedge \bigwedge_{i=0}^{2^n-1} ([\overrightarrow{inp_i}]_2 = i \wedge [\overrightarrow{out_i}]_2 = f(i)),$$

where

- $\overrightarrow{inp_i}$ *is a Boolean vector representing the inputs of the network to be synthesized for truth table line* i*,*
- $\overrightarrow{out_i}$ *is a Boolean vector representing the outputs of the network to be synthesized for truth table line* i *and,*
- Φ *is a set of constraints representing the synthesis problem according to [3, 20].*



(a) original (b) with SWOP extension

Figure 4. Encoding for exact synthesis

As an example Figure 4(a) shows the abstracted representation of the synthesis problem for the function specified in Table 1 (the values of the truth table are given as integers).

To apply SWOP to the exact approach and still ensuring minimality, all permutations are considered. This can be done – as mentioned above – by $\frac{n!}{g!}$ separate synthesis calls. However, exploiting the advanced techniques of the used SAT solvers leads to a faster synthesis. Therefore, just one additional Boolean vector is needed.

Definition 4 *Let $f : \mathbb{B}^n \rightarrow \mathbb{B}^n$ be a reversible function to be synthesized. Then, $\overrightarrow{p} = (p_{\lceil \log_2 \frac{n!}{g!} \rceil}, \ldots, p_1)$ is a Boolean vector representing the binary encoding of a natural number $p \in \{1, \ldots, \frac{n!}{g!}\}$ which indicates the chosen output permutation of the network.*

Using this vector, the SAT encoding is slightly extended: According to the assignments to $\overrightarrow{p}$ (set by the SAT solver) a value for p is determined, which selects the current output permutation. Depending on this permutation the respective output order is set during the search. More formally, the encoding of Definition 3 is extended as follows:

$$\Phi \wedge \bigwedge_{i=0}^{2^n-1} ([\overrightarrow{inp}_i]_2 = i \wedge [\overrightarrow{out}_i]_2 = \pi_{\overrightarrow{p}}(f(i)))$$

The extended encoding of the synthesis problem for the function specified in Table 1 is shown in Figure 4(b).

If the solver finds a satisfying assignment for the SWOP instance, one can obtain the network from the result as described in [3, 20] *and* the best permutation is provided by the assignment to $\overrightarrow{p}$.

Overall, this extension allows exact SWOP with only one synthesis call in contrast to $\frac{n!}{g!}$ separate ones. Furthermore, since the variables of $\overrightarrow{p}$ are an integral part of the search space, the permutations are checked much more efficiently. Because of modern SAT techniques (in particular *conflict analysis* [7]), during the search process reasons for conflicts are learned. This learned information prevents the solver from reentering non-solution search space, i.e. large parts of the search space are pruned. In contrast, this information is not available when each permutation is checked by separate calls of the solver. Thus, exact synthesis with output permutation is possible in feasible runtime when learning is exploited. Experimental results for exact SWOP are given in Section 6.

5.2. Heuristic Approach

To apply SWOP in a heuristic approach, the algorithm presented in [9] is considered. We avoid the construction of all possible permutations which would lead to a complexity increasing of $n!$ since in [9] garbage outputs are not supported. We propose a SWOP-based synthesis heuristic using a sifting algorithm inspired by [14] and hence reduce the above complexity to n^2. Because of the heuristic behavior of sifting maybe not the best permutation is determined. However, as the experiments in Section 6 show, significant improvements can be achieved in feasible runtimes.

```
(1)  HeuristicSWOP(f : B^n → B^n)
(2)    /* f is given as truth-table */
(3)    perm = {1, 2, ..., n};
(4)    c_best = synthesize(perm);
(5)    best_perm = perm;
(6)    for i = 0 to n − 2 do
(7)      for j = i + 1 to n − 1 do
(8)        tmp_perm = swap(perm, i, j);
(9)        c_tmp = synthesize(tmp_perm);
(10)       if (c_tmp < c_best)
(11)         best_perm = tmp_perm;
(12)       end–if
(13)     end–for
(14)     perm = best_perm;
(15)   end–for
```

Figure 5. Heuristic SWOP

The pseudo-code for the sifting algorithm is given in Figure 5. First, an initial permutation is chosen and the realization for this specification is synthesized (lines 3 and 4). As initial permutation we used the one given by the specification of the function. The gate count of this first realization is stored. After this, for each output the best position within the current permutation is searched. This is done by swapping the position of the current output with each of the other positions leading to new permutations (line 8). For each of this new permutations the respective realization is synthesized (line 9). If the gate count of such a realization is smaller than the current best known gate count (line 10), the current permutation is stored as being the best one (line 11). When each position for one output have been checked, the best permutation of these checks is used for the remaining outputs (line 14).

In summary, for each of the first $n - 1$ outputs, the algorithm will find a new position, that will result in a realization with the fewest gates – when synthesized with the heuristic algorithm from [9]. Therewith the complexity of SWOP can be reduced while still improving the obtained results as the next section will show.

6. Experimental Results

This section provides experimental results for SWOP. In total four different aspects are studied: (1) the reduction of the complexity of SWOP when garbage outputs are considered, (2) the results of exact SWOP in comparison to previous exact approaches, (3) the results of heuristic SWOP in comparison to the common heuristic approach, and (4) the quality (with respect to the number of gates) of the circuits synthesized by SWOP in comparison to the currently best known realizations.

For exact synthesis we used the algorithm introduced in [20] (the SWOP extension was implemented on the top of this approach). As heuristic approach the template matching algorithm described in [9] has been used. The respective benchmark functions have been taken from [21]. All experiments have been carried out on an AMD Athlon 3500+ with

Table 2. SWOP considering garbage outputs

				SWOP		OPT. SWOP		
BENCH.	n	g	c	$n!$	TIME (S)	$\frac{n!}{g!}$	TIME (S)	IMPR
4mod5	5	4	5	120	233.18	5	7.37	31.6
decod24	4	0	5	24	0.10	24	0.10	1.0
gt4	4	3	3	24	<0.01	4	<0.01	1.0
gt5	4	3	1	24	0.01	4	<0.01	>1.0
low-high	4	3	4	24	3.71	4	0.39	9.51
0-1-2	4	1	4	24	0.03	24	0.02	1.5
maj4_1	5	4	6	120	3500.90	5	2125.62	1.6
maj4_2	5	4	5	120	191.92	5	4.19	45.8
alu	5	4	6	120	2013.72	5	61.24	32.9
mini_alu_1	4	2	5	24	0.28	12	0.19	1.5
mini_alu_2	5	3	7	120	930.60	20	474.42	1.9
mini_alu_3	5	3	5	120	9.60	20	2.07	4.6

1 GB of main memory. All runtimes are given in CPU seconds. The timeout was set to 3600 CPU seconds (denoted by *TO* in the following).

6.1. SWOP with Garbage Outputs

In a first series of experiments we compare the different complexities which may occur when Toffoli networks for functions containing garbage outputs are synthesized. Here – as described in Section 4.2 – instead of $n!$ permutations only $\frac{n!}{g!}$ are considered.

Table 2 shows a comparison of the exact SWOP approach with both numbers of permutations for each incompletely specified function. The first three columns provide the name of the function, the number n of variables and the number g of garbage outputs, respectively. The minimal costs c (i.e. the minimal number of gates) of a Toffoli network representation is given in column c. Then, the runtimes of SWOP with $n!$ and with $\frac{n!}{g!}$ permutations are given (denoted by TIME). Furthermore, the improvement of the optimized SWOP (i.e. the synthesis with only $\frac{n!}{g!}$ permutations) over SWOP with all $n!$ permutation is provided (i.e. runtime of SWOP divided by runtime of OPT. SWOP).

As expected the reduction of permutations leads to better runtimes for all benchmarks. Improvements up to a factor of 45 can be achieved in the best case.

6.2. Exact SWOP

In this section we compare exact SWOP with the previous exact algorithm from [20]. The results are shown in Table 3.

Here again, the first column provides the name of the function, n and g denote the number of variables and the number of garbage outputs, respectively. The next columns give the minimal costs c determined by the two approaches and the corresponding runtimes. The last column shows information relating the complexity, i.e. the runtime overhead when output permutation is considered ($\frac{\text{SWOP-Time}}{\text{Syn-Time}}$) compared to the factor ($\frac{n!}{g!}$) resulting from the complexity analysis.

It can be seen that for many functions SWOP found smaller networks than the ones generated by the previous exact synthesis approach. Thus, removing the restriction for the output ordering leads to smaller networks for many of well known benchmark functions.

As expected the runtime for SWOP is higher in comparison to the runtime of pure exact synthesis. The reason is that the search space is obviously larger due to the number of output permutations that can be chosen. However, the increase is not as high as the number $\frac{n!}{g!}$. This can be seen in the last column of Table 3. For all benchmarks (except *4mod5* and *3_17*) the runtime of SWOP divided by the runtime of the previous synthesis approach is significantly smaller than the worst case complexity ($\frac{n!}{g!}$). As explained this is due to search space pruning, possible when the encoding is extended such that all permutations can be checked at once. Moreover, for some benchmarks (e.g. *maj4_2* or *alu*) the runtime of SWOP is even smaller than for a single exact solution. This reduction is caused by the fact, that smaller networks are found and thus the synthesis terminates earlier.

Table 3. Exact synthesis vs. exact SWOP

			EXACT SYNTHESIS		EXACT SWOP		$\frac{\text{SWOP-Time}}{\text{Syn-Time}}$
BENCH.	n	g	c	TIME (S)	c	TIME (S)	VS. $\frac{n!}{g!}$
4mod5	5	4	5	0.9	5	7.4	$8.4 > 5$
decod24	4	0	6	0.1	**5**	0.1	$1.7 < 24$
gt4	4	3	4	<0.1	**3**	<0.1	$1.0 < 4$
gt5	4	3	3	<0.1	**1**	<0.1	$1.0 < 4$
low-high	4	3	5	0.2	**4**	0.4	$2.2 < 4$
0-1-2	4	1	5	<0.1	**4**	<0.1	$0.5 < 24$
maj4_1	5	4	6	438.0	6	2125.6	$4.8 < 5$
maj4_2	5	4	6	13.6	**5**	4.2	$0.3 < 5$
alu	5	4	7	423.3	**6**	61.2	$0.1 < 5$
mini_alu_1	4	2	5	<0.1	5	0.2	$6.3 < 12$
mini_alu_2	5	3	8	2460.0	**7**	474.4	$0.2 < 20$
mini_alu_3	5	3	5	0.2	5	2.1	$12.2 < 20$
3_17	3	0	6	<0.1	**5**	<0.1	$9 > 6$
graycode6	6	0	5	<0.1	5	13.5	$224.7 < 720$
mod5d1	5	0	7	11.8	7	184.1	$15.6 < 120$
mod5d2	5	0	8	9.9	8	1097.6	$109.9 < 120$
mod5mils	5	0	5	0.1	5	1.7	$21.0 < 120$
rand0	4	0	8	15.3	**7**	26.4	$1.7 < 24$
rand1	4	0	8	5.8	**7**	28.3	$4.9 < 24$
rand2	4	0	9	154.5	**8**	150.3	$1.0 < 24$
rand3	4	0	9	231.5	9	1895.6	$8.2 < 24$
rand4	4	0	9	151.1	9	569.9	$3.8 < 24$

6.3. Heuristic SWOP

In this section we compare the results of heuristic synthesis with output permutation. In fact, the results obtained by common heuristic synthesis (according to [9] in its newest version) are compared with SWOP when all permutations are considered (ALL PERMS) and with SWOP when the sifting algorithm introduced in Section 5.2 is used (SIFTING).

The results are given in Table 4 showing the gate counts of the resulting realizations as well as the time needed for their synthesis.

As can be clearly seen, the effect of output permutation is significant for most of the functions. For example, for the function *aj-e13* the realization is reduced by 30 percent from 40 gates to 28 gates. The best absolute reduction of gates can be observed for function *hwb8*. Here, 35 gates are saved in total when output permutation is applied.

But not only the improvements are of interest. Even a comparison of the best and the worst permutation (shown in column c for ALL PERMS) give some interesting insight. For example, consider the function *hwb5*. One output permutation results in a circuit with 38 gates, while another permutation results in 62 gates. Since a heuristic minimization procedure is used, the results will most likely not be optimal. In fact, according to Proposition 1 the difference between the best and the worst permutation for *hwb5* can not be greater than 12 for minimal realizations – yet it is 24.

Table 4. Heur. synthesis vs. heur. SWOP

		Heuristic Synthesis		Heuristic SWOP all perms		Heuristic SWOP Sifting	
Bench.	n	c	Time (s)	c	Time (s)	c	Time (s)
3_17	3	6	0.03	6-7	0.32	6	0.25
4_49	4	17	0.40	**14**-22	4.09	**16**	1.09
4mod5	5	9	0.03	9-21	10.02	9	0.75
5mod5	6	18	0.13	**14**-37	254.14	18	3.59
aj-e10	5	33	0.63	**22**-51	107.03	**30**	8.21
aj-e11	4	12	0.09	**11**-22	2.46	**11**	0.55
aj-e12	5	26	0.35	**25**-57	103.37	**25**	8.11
aj-e13	5	40	0.97	**28**-51	112.70	**34**	12.31
ex1	3	4	<0.1	4-8	0.08	4	0.06
graycode3	3	2	<0.1	2-5	0.01	2	0.01
graycode4	4	3	0.01	3-9	0.32	3	0.07
graycode5	5	4	0.03	4-13	4.72	4	0.31
graycode6	6	5	0.08	5-18	67.25	5	1.08
hwb3	3	7	0.06	**6**-11	0.32	7	0.29
hwb4	4	15	0.35	**10**-21	3.70	**10**	0.69
hwb5	5	55	1.66	**38**-62	153.71	**44**	16.49
prime5	6	15	0.20	**13**-40	227.05	**13**	3.09
prime5a	6	16	0.10	**14**-41	291.58	**14**	3.92
ham3	3	5	0.01	**3**-5	0.02	**4**	0.03
hwb6	6	125	7.08	–	TO	**91**	89.20
hwb7	7	283	33.26	–	TO	**259**	656.82
hwb8	8	676	152.13	–	TO	**641**	4525.22
ham7	7	23	0.34	–	TO	23	49.47
rd53	7	16	0.26	–	TO	**13**	10.04

Finally it is shown, that sifting provides good results in a fraction of the CPU time. For most functions with more than six variables it is not feasible to minimize the function considering all permutations. However, sifting offers significant improvements for most of these functions (see the bottom rows of Table 4).

6.4. Reductions Achieved by SWOP

Finally, the quality (with respect to the number of gates) of some circuits synthesized by SWOP is compared to the currently best known realizations obtained by common synthesis approaches. Table 5 shows a selection of functions with the gate count of the currently best known realization (Best known c). The source of this realization is given in column Src. The gate count when output permutation is considered is given in column SWOP c.

Synthesis with output permutation enables the realization of smaller networks than the currently best known realizations. As an interesting example the realizations of the *hwb4* function is observed in more detail. For the initial function specification a *minimal* realization with 11 gates have been synthesized by the *exact* approach described in [3]. Now, using output permutation we are able to synthesize a smaller realization with only 10 gates using a *heuristic* approach.

7. Conclusions and Future Work

In this paper we introduced synthesis with output permutation (SWOP). We discussed the best case benefit and introduced different strategies facing the increasing complexity of our new synthesis paradigm. Output permutation have been applied to a representative of exact as well as heuristic synthesis, respectively. On our set of functions we showed that significant reductions (with respect to the number of gates) can be achieved, i.e. considering output permutation is beneficial. For some cases we synthesized multiple control Toffoli networks which are smaller than the currently best known realizations – even smaller than the ones today known as minimal.

Table 5. Best results obtained by SWOP

	Best known		SWOP	
Bench.	c	Scr.	c	Δc
decod24	6	[20]	5	1
alu	7	[20]	6	1
gt5	3	–	1	2
3_17	6	[20]	5	1
4_49	16	[21]	14	2
aj-e13	40	–	28	12
hwb4	11	[3]	10	1

For future work we plan to integrate the proposed methodology in approaches that also consider other cost metrics during synthesis (like e.g. [22, 23]).

8. Acknowledgements

This work was supported by the German Academic Exchange Service (DAAD).

References

[1] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVinchenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *The American Physical Society*, 52:3457–3467, 1995.

[2] E. F. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982.

[3] D. Große, X. Chen, G. W. Dueck, and R. Drechsler. Exact SAT-based Toffoli network synthesis. In *ACM Great Lakes Symposium on VLSI*, pages 96–101, 2007.

[4] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares. In *Int'l Symp. on Multi-Valued Logic*, pages 220–225, 2008.

[5] P. Gupta, A. Agrawal, and N. Jha. An algorithm for synthesis of reversible logic circuits. *IEEE Trans. on CAD*, 25(11):2317–2330, 2006.

[6] W. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski. Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis. *IEEE Trans. on CAD*, 25(9):1652–1663, 2006.

[7] J. Marques-Silva and K. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. on Comp.*, 48(5):506–521, 1999.

[8] D. Maslov and G. W. Dueck. Reversible cascades with minimal garbage. *IEEE Trans. on CAD*, 23(11):1497–1509, 2004.

[9] D. Maslov, G. W. Dueck, and D. M. Miller. Toffoli network synthesis with templates. *IEEE Trans. on CAD*, 24(6):807–817, 2005.

[10] D. M. Miller and G. W. Dueck. Spectral techniques for reversible logic synthesis. In *6th International Symposium on Representations and Methodology of Future Computing Technology*, pages 56–62, 2003.

[11] D. M. Miller, D. Maslov, and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In *Design Automation Conf.*, pages 318–323, 2003.

[12] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[13] A. Peres. Reversible logic and quantum computers. *Phys. Rev. A*, (32):3266–3276, 1985.

[14] R.Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Workshop on Logic Synth.*, pages 3a–1–3a–12, 1993.

[15] M. Saeedi, M. Sedighi, and M. S. Zamani. A novel synthesis algorithm for reversible circuits. In *Int'l Conf. on CAD*, pages 65–68, 2007.

[16] V. Shende, A. Prasad, I. Markov, and J. Hayes. Reversible logic circuit synthesis. In *Int'l Conf. on CAD*, pages 353–360, 2002.

[17] L. Storme, A. D. Vos, and G. Jacobs. Group theoretical aspects of reversible logic gates. *Journal of Universal Computer Science*, 5:307–321, 1999.

[18] T. Toffoli. Reversible computing. In W. de Bakker and J. van Leeuwen, editors, *Automata, Languages and Programming*, page 632. Springer, 1980. Technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci.

[19] R. Wille, G. Fey, D. Große, S. Eggersglüß, and R. Drechsler. Sword: A SAT like prover using word level information. In *VLSI of System-on-Chip*, pages 88–93, 2007.

[20] R. Wille and D. Große. Fast exact Toffoli network synthesis of reversible logic. In *Int'l Conf. on CAD*, pages 60–64, 2007.

[21] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler. RevLib: an online resource for reversible functions and reversible circuits. In *Int'l Symp. on Multi-Valued Logic*, pages 214–219, 2008. RevLib is available at http://www.revlib.org.

[22] R. Wille, H. M. Le, G. W. Dueck, and D. Große. Quantified synthesis of reversible logic. In *Design, Automation and Test in Europe*, pages 1015–1020, 2008.

[23] G. Yang, X. Song, W. N. N. Hung, and M. A. Perkowski. Fast synthesis of exact minimal reversible circuits using group theory. In *ASP Design Automation Conf.*, pages 1002–1005, 2005.

Cone Resynthesis ECO Methodology for Multi-Million Gate Designs

Suresh Raman
Intel Technologies India Pvt Ltd, Bangalore

Mike Lubyanitsky
Intel Corporation, Santa Clara, CA 95052

***Abstract*—In this paper, we talk about techniques to incrementally resynthesize logic cones within a large design impacted by multiple RTL changes in order to accommodate a late functional ECO. In design methodologies where the RTL is hierarchical and the post route netlist is flat, mapping a change in the behavioral description to the post layout netlist is very complicated and may not even be feasible if the RTL is not written in a synthesis friendly manner. We try to attack this problem by introducing a technique that causes minimum perturbation to the gate level netlist, thereby retaining to a large degree, the goodness metrics of timing convergence, routability and layout cleanliness that were achieved during the various design milestones. This paper talks about the cone resynthesis ECO methodology in detail and highlights its usefulness during tight product deliverable schedules.**

***Index Terms* – Engineering change order (ECO), formal verification, cut point, cone synthesis, placement, routing**

I. Introduction

ECO (Engineering Change Order) is the process of introducing a change during the late stages of the design cycle when most of the quality metrics in terms of timing, routability and layout convergence have already been met. ECOs are a necessity since they have many advantages over the default flow in terms of shorter fabrication time with focused-ion-beam milling, lower fabrication cost with metal-only changes, shorter design time as opposed to running the complete flow, lower design cost and most importantly, predictable results to incremental changes. In terms of classification, an ECO may either be a functional change or a non-functional one. Functional ECOs are typically triggered by late enhancement requests from customers or a late bug identified during pre or post-silicon validation. Non-functional ECOs are however, changes required to be made to the netlist to fix timing and crosstalk problems or capacitance/transition violations and hence, do not warrant a RTL change. Non-functional ECOs are generally easier to converge since they are localized in scope as opposed to functional ECOs that involve adding or reconnecting numerous gates. ASIC design involves integrating several of the complex functional ECOs late into the tapeout cycle and many of them involve multiple changes in the RTL, where accurate identification of nets in the gate level netlist that map exactly to the RTL changes becomes a very intricate and daunting task. Additionally, the ECO cycles during a metal layer stepping or a potential base layer stepping have very aggressive schedules and any slippage is costly in terms of product launch, impacting the bottomline. The complexity of a functional ECO could be gauged on several orthogonal scales such as (a) mask complexity, where a full set of masks is very expensive whereas metal-only fixes require no base layer changes, (b) design complexity, depending on the number of RTL modules modified, (c) combinational logic complexity, depending on if existing logic gates are rewired or additional gates (or spare cells) are required, (d) sequential logic complexity, depending on number of flops/latches touched by the ECO and the downstream impact to the clock tree and scan flows and lastly, (e) size complexity in terms of the number of gates added by the ECO. There have been commercial tools that tackle this issue such as ECO Compiler from Synopsys and Conformal ECO from Cadence. However, each of these has certain shortcomings – ECO Compiler does a good job at implementing large ECOs of more than 500 gates. However, the implementation is not physically aware and importantly, the tool is no longer supported. On the other hand, Conformal ECO from Cadence is difficult to use in a hierarchical design where ports get added and additionally, the quality of the netlist is somewhat inferior to a corresponding manual implementation in terms of gate count. In such a scenario, cone resynthesis is a promising technology where the fanin cone to the impacted sequential elements is incrementally resynthesized, placed and taken through the entire back end and timing convergence loop, thus avoiding huge turnaround times associated with the complete respin of the block. Moreover, one could envisage practical scenarios during a chip design cycle where a functional bug fix in the RTL is not manually ECO'able using spare cells (metal only stepping) or normal standard cells (base stepping). In such a situation, the cone resynthesis flow is a very handy tool to implement the ECO with minimum change to the existing design. In this paper, we talk about the entire methodology in detail and illustrate its effectiveness on a live design.

To our knowledge, there is little literature talking about the ECO resynthesis methodology spanning both front end [1][2] and back end. Lin et al [2] try to solve the problem of late implementation using a programmable rectification module to reduce mask cost and improve turn around time. Lot of previous work on ECO techniques focus on the back-end and timing aspects of the design. There are several papers that talk about ECO routing techniques as well as ECO timing

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.28

convergence. Chen et al [3] talk about the ECO timing optimization problem using spare cells and present a dynamic programming algorithm for spare cell rewiring within an essential bounding polygon without loss of solution optimality. Li et al [4] propose a tile based ECO routing flow using routing graph reduction for promoting tile propagation speed. Xiang [5] addresses the issue of fixing coupling capacitance violations using an ECO routing algorithm that processes signal wire segments on a layer one by one while trying to find a clean routing solution satisfying all constraints following which the total deviation is minimized based on the shortest path algorithm. A survey of the most popular incremental algorithms in physical design is done by Coudert et al [6][7][8] where they outline a set of problems in synthesis, placement, and routing and suggest possible solutions.

In this paper, we talk about an ECO design automation flow that spans from the ECO implementation in the logical netlist to the back end convergence of the design. We focus on the building blocks of this capability that enables integrating complex ECOs in a matter of days that would otherwise have taken a long time to converge.

II. PROBLEM FORMULATION

Cone resynthesis flow could be thought of as a complete framework that takes as input (a) the frozen flat gate level netlist and (b) the modified RTL with the functional change and outputs a flat netlist with the modified RTL mapped to gates and further, that is converged w.r.t. timing and layout DRC (Design Rule Check) violations. Fig. 1 illustrates the entire flow using a block diagram representation.

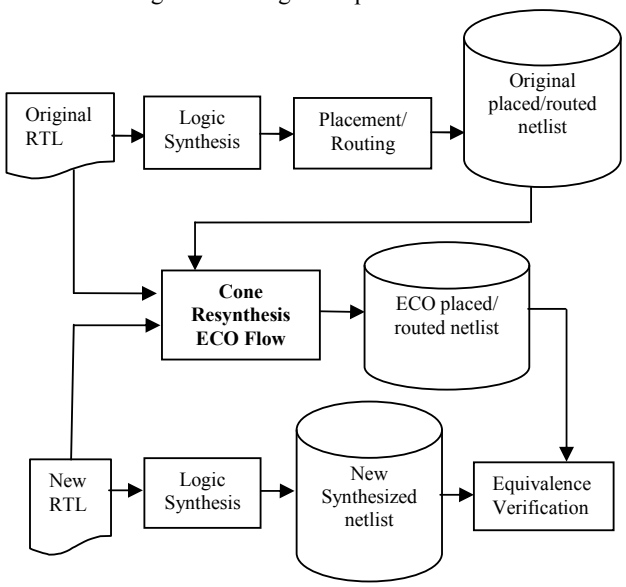


Fig. 1: Block Diagram of Cone Resynthesis Flow

The entire system comprises of two parts: (a) a front end subsystem that generates the logic level netlist that is formally equivalent to the ECO'ed RTL and (b) a back end subsystem that places the newly added cells, and incrementally routes the corresponding nets with minimum disturbance to the majority of the nets in the design. We now talk about the front end and back end flows individually in detail.

A. Front End Cone Resynthesis Flow

The high level description of this flow is to extract a new fanin cone to the impacted logic and swap the old cone with the new one. Given a hierarchical RTL that has the functional ECO change and the frozen gate level netlist that maps to the original RTL, the first step is to run formal equivalence verification (FV) between the new RTL and the old netlist in order to obtain the failing cut points. Fig. 2 shows a sample design with logic cones fanning in into what are called *"cut points"*. Cut points are either D-pins of sequential elements (latches/flip-flops) or macro data pins or primary inputs/ outputs, and are the points where the fanin cone is evaluated and compared between the golden and revised netlists.

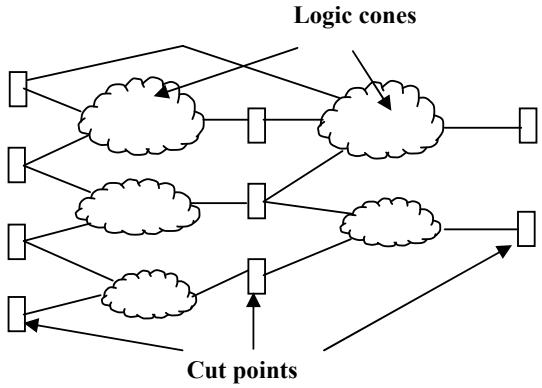


Fig. 2: Sample Design depicting fanin cones and cut points

Since the golden netlist (RTL) is hierarchical, we would require giving the formal verification tool the list of sub-modules to be compared. Care must be taken that these sub-modules are flattened to leaf level without which the underlying hierarchies would be considered as black boxes and in case of a mismatch, the failing cut point would be reported as a black box pin rather than a sequential element within that hierarchy. Such mismatches would gate the resynthesis flow from extracting the entire cone to the impacted cut points.

The output of the formal equivalence verification run is a list of all failing cut points where there are mismatches between the RTL and the netlist. These points could either be D-pins of latches or flops or could be a submodule port. The next step is to isolate the lowest level sub-hierarchies in RTL that completely contain all these mismatches. Once these sub-hierarchies are identified, a change list requires to be generated that specifies the exact changes to be made to the netlist in terms of cell additions as well as the new net connections and disconnections. Ports as endpoints of fanin cones could be added when the scope of change is the top level. This is simply added as just another end point after the port is created manually. Connections to these lower level ports also known as hierarchical pins could be automated

using a script. In order to generate the change list, the new RTL with the functional ECO requires to be resynthesized, and this is done using the Design Compiler tool from Synopsys. Depending on the runtimes, resynthesis could be done either at the top level or at the level of the sub-hierarchy that fully contains the mismatching cut points. Once the resynthesized netlist is ready, the change list containing the new fanin cone to the failing cut points is extracted and incrementally plugged in into the original gate level netlist, whereas the original cone is disconnected. The inputs to the new cone are swapped with the names that match the original post route netlist to enable proper connections. Further, the dangling cone is removed through a reverse breadth first recursive traversal until either a sequential element, a macro output, a primary input or an instance used by another cone is reached. Alternatively, the original cone could be just disconnected and the inputs tied to the start points of the original cones. Fig. 3 illustrates the sequence of steps where the original fanin cones A, B, and C to the flip flops FF3, FF4 and the primary output PO are swapped with the new cones from the resynthesized gate level netlist namely, A', B' and C' respectively. The ECO'ed netlist in the figure shows the new cones as well as the original ones disconnected from their previous connections.

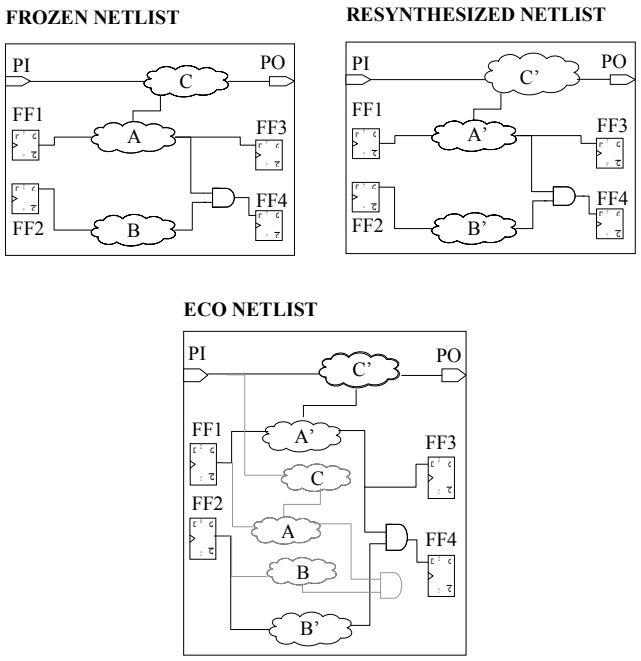


Fig. 3: Illustration of cone-in ECO flow

A point to note is that all the new instances and nets in the resynthesized netlist must be prefixed with an ECO tag in order to avoid name clashes with an already existing instance or net. The change list with the net additions and deletions is incrementally applied to the post layout netlist and verified for logic equivalency against the ECO'ed RTL model. If the ECO involves addition of new sequential elements, then FV must ignore scan connections since the new latch or flop is not yet hooked on to the scan chain. The entire front end cone resynthesis algorithm is formalized in Fig.4

Algorithm: Front_End_Cone_Resynthesis (N_{OLD}, R)
$N_{OLD} = \{n_1, n_2, n_3, ..., n_P\}$; where P is size of original netlist
R = New RTL model with functional ECO
$FI(k)$ = Fanin cone to point k
n represents nets and c represents cells
Generate ECO netlist N_{ECO} equivalent to R, such that $\Delta(N_{OLD}, N_{ECO})$ is minimum
Initialize N_{ECO} to N_{OLD}
Resynthesize R to obtain new netlist, $N_{NEW} = \{n_1, n_2,.., n_Q\}$
Run formal verification for N_{OLD} against R and generate $K = \{k_1, k_2, ... , k_M\}$ where K is list of M mismatching cut points
For each cut point, $k \varepsilon K$
  Identify $FI_{OLD}(k) = \{n_i, c_i\}$ such that $n_i, c_i \varepsilon N_{OLD}$
  Identify $FI_{NEW}(k) = \{n_j, c_j\}$ such that $n_j, c_j \varepsilon N_{NEW}$
  For each $\{n_j \varepsilon N_{NEW}\}$ and $\{c_j \varepsilon N_{NEW}\}$
    Add n_j and c_j to N_{ECO}
  For each $\{n_i \varepsilon N_{OLD}\}$ and $\{c_i \varepsilon N_{OLD}\}$
    Remove n_i and c_i from N_{ECO}

Fig. 4: Front End Cone Resynthesis Flow Algorithm

B. Back End Cone Resynthesis Flow

After completing the front end cone resynthesis flow, the logic netlist with the new connectivity information is ready. However, the new cells and nets require going through the entire back end physical design before the database could be declared as a feasible candidate for tapeout. Specifically, any new sequential elements inserted during the ECO must be added to the scan chain and clock tree routed to this element in an optimal manner such that the skew and latency requirements are met. Since the number of added sequentials is typically less during ECO cycles, identification of spatial proximity to the nearest scan chain and optimum clock net branch to tap to the new sequential cells is usually performed manually. Once the clock tree is ECO'ed, the next steps are to identify a valid placement for the new logic cells and to route the new nets in a timing driven manner so that the timing and routability metrics are met within a minimum number of iterations. Since the number of cells impacted by the ECO is not always small, manually placing and routing them may not be an option. Hence, the back end convergence loop necessitates a sequence of optimization flows that can significantly reduce the turn-around time to churn out a fully converged netlist. We now discuss the various options that might be exercised to achieve the same.

The methodology for ECO placement of the new cells is a function of the design complexity. If the design is complex, a don't touch attribute is applied on all the pre-existing cells and nets in the design following which the ECO cells are placed incrementally in timing driven mode. This ensures that the perturbation to the design is minimal and the convergence milestones that have been achieved in terms of timing and layout for the major part of the design are retained. However,

it is not necessary to fix the cell placement if we could potentially afford another round of DRC and hold fixes. Doing so gives the placement tool higher degrees of freedom to meet setup requirements and moreover, in most of the tools, post clock tree optimization in ECO mode is allowed to change the design state minimally. A point worth noting is that since the new netlist was generated by a synthesis tool that works off a wire load model, the cells in the ECO'ed paths might be improperly sized, resulting in timing violations. In order to overcome this issue, the placement step must be run twice with varying capabilities for optimization. The first run places the cells in the best possible locations in a post-route environment with propagated clocks and annotated wire delays, thus ensuring that the placement engine has the most accurate parasitic data in its memory. Inspite of this, there might be timing violations since the cell sizes were determined in the absence of physical information, and purely based on wire load models that are largely inaccurate for deep submicron geometries. Therefore, we require rerunning the placement engine, this time in "sizing only" mode where the cells on the ECO paths are upsized or downsized depending on the slack availability. The timing information could be obtained using the native timing engine within the placer or could be generated from the actual sign-off timing engine. Although more accurate, the latter results in larger runtimes and is acceptable only in cases when the design complexity is manageable and product timelines allow longer runway for convergence. Even after multiple placement runs, there is a possibility that there are remaining violations if the placement has been arrived at in a constrained environment where the rest of the logic is not allowed to move. In such a scenario where global optimization might not be enabled, the placement engine might yield a local optima putting the onus on the designer to try out various flavors to increase the inherent "freedom" to the placer, by removing don't touch attributes on related logic cones and by running the placer with high effort.

Once the placement is achieved with decent timing quality, the next step is to route the newly created nets in a timing driven fashion and in a way that introduces minimum routing DRC violations with the existing nets. If the incremental routing results in high local congestion and hence huge DRCs, the fallback option would be to fully route the entire design, keeping in mind that doing a full route must be the last resort since it undoes most of the timing fixes done during the earlier stages. At this point, one could potentially fix most of the maximum transition and capacitance violations reported by the router timing engine, assuming that it correlates to a fair degree with the final signoff timer. Fixing these violations upstream could potentially help reduce iterations between timing and placement/routing. On a similar note, one could also fix the hold violations at the placement or routing stage itself by inserting buffers or downsizing drivers on the nets in the failing paths. The stage at which each of these violations is fixed is purely based on the design methodology/legacy and how well the placement and router timing engines correlate with the final signoff timer. Post placement and routing, all the nets in the design are extracted and the RC parasitics are fed to the timing engine to generate the timing health indicators for the design. The setup, hold, slope and C_{max} violations reported by the signoff timer are fixed in subsequent loops and repeated until the design is timing and layout clean. Each of the steps in the entire flow is captured in Fig. 5.

Algorithm Back_End_Cone_Resynthesis *(N_{ECO})*
N_{ECO}: Logical netlist equivalent to ECO'ed RTL
Generate layout database converged w.r.t. all physical design quality metrics

1. Stitch newly added sequentials into nearest scan chains
2. Connect clock pins of newly added sequentials to existing clock tree
3. Apply don't touch to all existing cells and nets in design
4. Run timing driven placement for ECO cells with propagated clocks and annotated wire delays
5. Rerun placement in sizing only mode
6. Route ECO nets incrementally in timing driven mode
7. Extract RC parasitics for the entire design
8. Run sign-off timing engine
9. Generate setup, hold, slope and C_{max} violation reports
10. If no timing violations, fix layout DRCs and exit
11. Else fix remaining violations by buffer insertion, gate sizing, and load isolation
12. Go back to Step 6

Fig. 5: Back End Cone Resynthesis Flow Algorithm

III. RESULTS

We ran the entire cone resynthesis flow on four large blocks in our design that have 500K-800K placeable instances. Synopsys Design Compiler was used for all synthesis runs as well as to extract the fanin cones to the failing cut points reported by formal verification. Table 1 depicts the complexity of the blocks as well as the ECOs in terms of the change magnitude in the number of cells.

BLOCK	CELL COUNT	CHANGE MAGNITUDE (# cells)
A	400K	4280
B	402K	4120
C	836K	2328
D	226K	1130

Table 1: Design/ECO complexity

Table 2 shows the results of ECO runs on blocks with varying complexity in terms of the magnitude of the cell count change. Metrics of interest such as number of cells modified in order to close timing and the estimated equivalent manual effort are mentioned. Manual effort numbers are projected estimates based on available design data that had similar cell changes and ECO complexity. The functional ECOs chosen as testcases are the ones that absolutely could not be manually

ECO'ed or are difficult to ECO on the gate level netlist.

BLOCK	ESTIMATED MANUAL EFFORT (man days)	ACTUAL EFFORT WITH CONEIN FLOW (man days)	NUMBER OF CELLS CHANGED TO CONVERGE TIMING
A	40	12	5650
B	35	12	6793
C	50	14	18724
D	5	3	2124

Table 2: Estimated Manual Effort versus cone resynthesis effort for varied designs

The ECOs in Blocks A and B involved changes in a common module that was multiply instantiated four times, that warranted identifying the corresponding manual ECO in four different physical implementations of the affected module. We underwent the manual ECO process for four weeks where we tried identifying the nets in the gate level netlist that exactly mapped to the RTL change, and this effort yielded success for two of the four instantiations. This was because two of the four instantiations were synthesized in an optimized fashion whereas the remaining two were fairly complicated with a large number of intermediate signals yielding a netlist difficult for an ECO. Manual implementation involved tracing thousands of start points in the fanin cones to the impacted registers and this was required to exactly pinpoint the logic to be modified. We went through the cone resynthesis flow for this ECO and summarize the results in terms of savings with this flow versus the corresponding manual effort.

For Block C, that had 836K instances, the functional ECO was particularly difficult, wherein a considerable amount of manual effort was expended in order to generate a metal only ECO. Even after significant effort that spanned several man days, we received formal verification mismatches between the RTL and the modified netlist, thus requiring an alternative implementation using cone resynthesis. In the absence of this flow, complete respin of this block would have taken around 8-10 weeks making it impossible to tape out the chip on schedule. However, with the cone resynthesis flow, it took only 2 weeks to converge on the entire design in terms of timing as well as layout cleanliness. For another block D with 226K cells, that had a functional ECO, we simultaneously ran the conein flow as well as pursued manual implementation and found that the conein flow signoff netlist took less time compared to the corresponding manual effort. In this case, we changed 2124 cells to converge on timing with the cone resynthesis flow as opposed to only 115 cells with the corresponding manual effort. In addition to these four blocks, we have run the cone resynthesis algorithm on several ECOs of low and medium complexity that had to be implemented during a metal-only stepping and found favorable results in terms of netlist quality and implementation time that did not exceed 2-3 days for ECOs impacting a maximum of 1K gates.

In conclusion, experimental data shows that the cone resynthesis flow is a very useful tool in speedy convergence of the design and can be employed effectively in scenarios where the RTL change looks difficult to map to the actual gates. For ECOs with relatively less complexity, the cone resynthesis flow might yield a higher gate count compared to a manual implementation and hence higher convergence turn around time, thereby warranting effective judgement on the part of the designer to make a choice on either of the approaches. Either way, cone resynthesis presents itself as a viable and predictable alternative during tapeout crunch situations and could be deployed across various types of ECOs.

IV. CONCLUSIONS

To summarize, cone resynthesis automation flow enables more than 3X reduction in turn around times for timing and layout convergence for difficult to implement functional ECOs. However, the flow does have a drawback where the number of cells that it adds might be much more than what a manual ECO would do. This is due to the fact that the flow is based on simple logic equivalence where the entire fanin cone to a cut point is swapped completely and the algorithm is not smart enough to find the farthest logic to the cut point where the cones start diverging. The size of these cones is typically much small compared to the design size and hence, the extra gate count is usually acceptable. However, in cases where the designer has "good enough" confidence to identify the gates that map to the functional change, the cone resynthesis flow may not be recommended due to the extra overhead in terms of cell count and layout convergence.

V. ACKNOWLEDGMENTS

The authors would like to acknowledge Patrick Tsui, Sitanshu Jain and Srinivas Jammula for their continuous support during the development cycle of the flow.

REFERENCES

[1] D. Brand, A. Drumm, S. Kundu, and P. Narain, "Incremental Synthesis," In Proc. IEEE/ACM International Conference on Computer Aided Design, Nov. 1994, pp. 14-18.

[2] C. Lin, Y. Huang, S. Chang, W. Jone, "Design and Design Automation of Rectification Logic for Engineering Change," In Proc. of the 2005 conference on Asia South Pacific design automation, pp. 1006-1009.

[3] Y. Chen, J. Fang and Y. Chang, "ECO timing optimization using spare cells," In Proc. IEEE/ACM International Conference on Computer Aided Design, Nov. 2007, pp. 530-535.

[4] Y. Li, J. Li and W. Chen, "An efficient tile-based ECO router using routing graph reduction and enhanced global routing flow," In IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol.26, no.2, pp. 345-357, Feb. 2007.

[5] H. Xiang, K. Chao, and M. D. F. Wong, "An ECO routing algorithm for eliminating coupling-capacitance violations," In IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol.25, no.9, Sep. 2006, pp. 1754-1762.

[6] O. Coudert, J. Cong, S. Malik, and M. Sarrafzadeh, "Incremental CAD," In Proc. IEEE/ACM International Conference Computer-Aided Design, Nov. 2000, pp. 236-243.

[7] J. Cong, and M. Sarrafzadeh, "Incremental Physical Design," In International Symposium on Physical Design, 2000, pp. 84-92.

[8] A.B.Kahng, and S. Mantik, "Mismatches of Incremental Optimizers and Instance Perturbations in Current Place-and-Route tools," In Proc. IEEE/ACM International Conference on Computer Aided Design, 2000.

[9] Synopsys Design Compiler X-2005.09-SP4 User Guide

A General Approach to High-Level Energy and Performance Estimation in SoCs

Sandro Penolazzi, Ahmed Hemani and Luca Bolognino
Dept. of Electronic, Computer and Software Systems, School of ICT, KTH, Kista, Sweden
Email: {sandrop, hemani, lucab}@kth.se

***Abstract*—We present a high-level methodology for efficient and accurate estimation of energy and performance in SoCs at Functional Untimed Level. We then validate the proposed method against gate level for accuracy and against TLM-PV for speed. We show that the method is within 15% of gate-level accuracy and in average 28x faster than TLM-PV, for the benchmark applications selected.**

I. Introduction

FUNTIME is an early estimation framework that provides estimates of energy and performance of SoCs. The level of abstraction that forms the basis for estimation is the Functional Untimed Level, which gives the name to the framework: FUNTIME. Important benefits of the FUNTIME method are:

Low Engineering Effort: the FUNTIME method does not require any extra engineering step, like building a Transaction Level Model. The method is naturally absorbed in the algorithmic design phase and architectural exploration and mapping. We emphasize that a simulation model of architecture is not required.

Speed: since algorithmic simulation is the basis for energy and performance estimation, the speed is very high compared to approaches that involve simulation of the architectural implementation at various levels of details, like TLM or RTL.

Accuracy: the FUNTIME method claims that it can achieve accuracy that is not too far from gate level, while remaining at algorithmic level, as shown in Fig. 1. Although this claim might sound extra-ordinary, it is based on the state-of-the-art SOC engineering practice of buy-and-assemble: we assume the existence of IPs that are characterized for energy and performance, based on detailed gate level simulation.

In the past, we have already outlined the overall FUNTIME method and focused on the possibility to use it for inferring the total amount of transactions occurring in a platform without having to run any architectural simulation [1]. In the present paper, we enhance FUNTIME to also allow energy and performance estimation.

II. Related Work

Many methods for system-level modeling have been developed by several research groups. Some of them are Metropolis [2], SPADE [3], SpecC [4] and TAPES [5]. What all these methodologies have in common is the fact of relying on architectural simulation to perform system-level estimation, either based on SystemC or other languages.

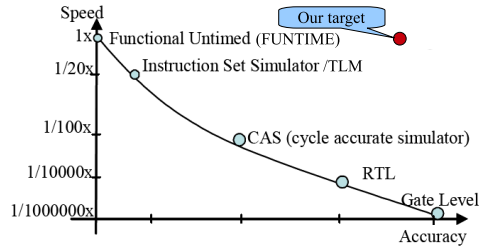


Fig. 1. Speed vs. accuracy for different levels of abstractions

On the contrary, FUNTIME differs mainly in these two aspects: a) the ability to avoid architectural simulation, b) the ability to also generate energy reports. Similarly to Metropolis and SpecC, FUNTIME adopts the concept of separating computation from communication, but it does not need to define any modeling language like in SpecC or Metropolis, which rely on SystemC. FUNTIME shares with SPADE the advantage of adopting untimed modeling. In addition, FUNTIME does not constrain designers to an exact architecture topology as TAPES does.

III. The FUNTIME method: 3 levels of abstraction

FUNTIME operates at 3 different levels of abstraction, as detailed in the present section and shown in Fig. 2. Using a bottom-up approach, we distinguish A) IP Level, B) FUNTIME Level and C) Refinement Level.

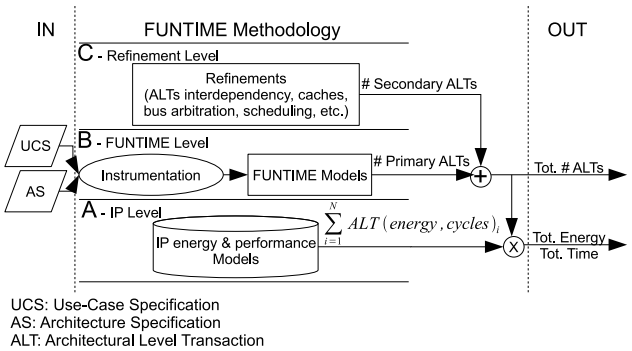


Fig. 2. The FUNTIME Methodology flow (A-IP Level, B-FUNTIME Level, C-Refinement Level)

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.25

A. IP Level

Due to the increasing complexity of modern SoCs, it is sensible believing that, even in the future, new architectures will be built more and more by reusing and assembling previously existing IPs, taken from a library and properly configured according to the specifications requirements. This is mainly aimed at reducing the time to market and at easing the whole design process.

It is therefore reasonable to believe that spending time in characterizing and building models for such IPs is worthwhile and represents a one-time effort made by the IPs provider. Models describing IPs architectural properties and configuration space are already available under the industrial XML-standard SPIRIT [6]. In the context of the present work, we consider the advantage of having IP-level models for energy and performance as well. This is shown in Fig. 2A.

In particular, in the scenario that we envisage, an IP energy and performance model characterizes each IP transaction in terms of energy and number of cycles, where a transaction directly expresses one of the IP functionalities. For instance, for a Processor IP, each instruction belonging to its instructions set represents one such transaction. Other examples of transactions could be read/write operations for a bus, tx/rx for an I/O device, encode/decode for a codec, etc. The final IP model can be expressed in form of look-up tables, mathematical formulas or by a combination of the two. In addition, the characterization accounts for variations in the IP configuration space, both in terms of static parameters, i.e. the VHDL generics, and run-time parameters, i.e. baud rate, frequency, etc.

In Section IV, an example of how an IP provider could build an energy and performance model for a processor is presented. Note that, in case an IP energy and performance model is not available, FUNTIME can still produce estimates for that IP in terms of total number of transactions. This is shown in Fig. 2.

B. FUNTIME Level

Once each IP transaction has been characterized for energy and performance, the total energy and performance for a full application can be calculated provided that the total number and type of transactions triggered by such an application is known.

Traditionally, this information can be collected from Instruction-Set Simulation (ISS), transaction-level simulation or any other architecture-based simulation. In our approach instead, since the idea is to avoid architectural simulation, the inference of architectural transactions is achieved by instrumenting the application, itself devoid of any architectural detail, to be architectural aware. An instrumented application is what we define a FUNTIME Model. To make the instrumentation possible, the FUNTIME Level needs to receive in input both an architecture specification (AS) and a use-case specification (UCS). The AS is an XML file that symbolically tells which IPs compose the architecture and how they are connected to each other. The UCS tells which applications are going to run on the architecture and how they are mapped. Transactions inferred at this level are defined *primary* transactions (P-ALTs). The process flow is illustrated in Fig. 2B.

Applications can be implemented either in software or in hardware. In the former case, we say that they are mapped to software IPs like a Processor, while in the latter case they can be mapped to a hardware macro. In the present work, we only focus on the generation of FUNTIME Models for software applications, which is detailed in Section V. The generation of FUNTIME Models for hardware applications is part of our future work.

Note that the process of retrieving amount and type of transactions is in general not only limited to the resource where the application is mapped, but can be straightforward also for interconnects and memories used by such a resource. For instance, by knowing that an application is mapped to a processor connected to a memory through a bus, it is possible to infer also the amount of bus and memory transactions, by counting how many store/load instructions have been issued.

C. Refinement Level

From Subsections III-A and III-B, once each IP transaction has been characterized for energy and performance, and the total occurrence of such transactions for the execution of an application has been determined, the total energy and performance can finally be estimated. However, in most cases this calculation would lead to a wrong estimation both for energy and timing, compared to the actual values collected from gate level.

The reason is that there are other implications, ignored in the two steps above, that need to be kept into account. Such implications are mainly related to transactions interdependency/ordering issues, as well as to hardware/software optimizations (i.e. caches or power management), and resource sharing (i.e. bus arbitration, scheduling).

The first direct effect implied by the elements above is a variation of the total application length, in terms of number of cycles and, consequently, in the final energy too. In traditional cycle-accurate architectural simulations, this would not represent an issue, as the exact number of cycles would come along as part of the simulation itself. In contrast, in our approach this contribution needs to be factored in separately.

In this paper, we demonstrate that this can be done by refining the trace of P-ALTs collected at the FUNTIME Level below. Such a refinement produces what we have defined as *secondary* ALTs (S-ALTs). This is shown in Fig. 2C. The sum of P-ALTs and S-ALTs represents the total number of triggered ALTs, from which total energy and performance numbers can finally be extracted.

In Section VI and VII, we detail how a possible refinement can be done to take into account the implications due to transactions interdependency and to the presence of caches.

IV. A LEON3 ENERGY AND PERFORMANCE MODEL

In this Section, we propose an overview on how an IP energy and performance model for a processor can be defined.

In the context of our research, we have done it for a SPARC-based Leon3 processor [7], however the methodology is general and can be applied to other processors as well. The entire process relies on having an RTL representation of the IP and on extracting energy/timing information from its equivalent gate level. The main steps are the following.

First, a set of significant configurations is chosen by changing the VHDL generics in the RTL model; such configurations include a version with/without cache, with/without mul/div unit, for a total of 4 different configurations. Enabling the cache generic only specifies that the cache controller has to be synthesized. The cache itself is located outside the core module and is not synthesized. For the 4 configurations, synthesis has been carried out in our case by using Cadence RTL Compiler and the 90nm TSMC library.

As a second step, gate-level simulations are run to extract energy and number of cycles associated to each processor instruction for all the configurations selected. The method used to make this characterization relies on executing for 100 consecutive times a basic sequence composed of 5 *nops*, 1 *instruction under test* and 5 *nops*, for a total of 1100 executed instructions, as shown in Expression 1.

$$100 \times (5 \cdot NOP, IUT, 5 \cdot NOP) \tag{1}$$

The first selected IUT is the *nop* itself. From Expression 1, this will lead to the execution of 1100 *nops*, for which energy is also calculated. The energy of a single *nop* is therefore given by Eq. 2.

$$E_{NOP} = \frac{E_{1100\_NOP}}{1100} \tag{2}$$

Once the energy of a single *nop* is known, calculating the energy for the other IUTs is quite straightforward: since in these cases there will be 100 executed IUTs and 1000 nops, the energy of 100 IUTs is given by Eq. 3

$$E_{100 \cdot IUT} = E_{1000\_NOP+100\_IUT} - 1000 \cdot E_{NOP} \tag{3}$$

and therefore the energy for a single IUT is

$$E_{IUT} = \frac{E_{100\_IUT}}{100} \tag{4}$$

As a third step, the energy values for the whole instruction set are collected in a LUT together with the number of cycles, as shown in the fragment reported in Table I. Note that reporting energy rather than power has the advantage of making the model independent of the frequency at which the IP runs, and therefore also more general.

TABLE I
LEON3 LUT FOR ENERGY AND PERFORMANCE

Instr.	Energy[pJ]	Cycles
add	92.74	1.00
ld	75.17	1.00
nop	25.34	1.00
st	108.72	2.00
...	...	...

V. FUNTIME MODELS FOR SOFTWARE APPLICATIONS

The generation of FUNTIME Models for applications mapped to a processor relies on the ability to relate the source code lines of the application to the assembly instructions contained in the corresponding basic blocks for the target processor. This is obtained in the following steps:

- An application is compiled for a *host* environment (typically a common x86-based PC).
- Mixed source/assembly code files are generated for the same application for the *target* processor (the Leon in our example).
- The application is executed natively in the host environment and a code coverage tool is used to extract the number of times that each source code line has been executed.
- By relating this information to the instructions and basic blocks of the same application for the target processor, it is possible to generate a report of primary transactions (P-ALTs), as shown in Table II.

TABLE II
EXAMPLE OF TRANSACTIONS REPORT

Total transactions =	2 397 366
save	673
st	159 266
mov	123 528
...	...

Note that this approach to generating and executing FUNTIME Models for software applications is very general, which leads to high re-usability, and requires no extra engineering effort: it just needs a compiler, for the host and target architectures, a code coverage tool and a small script that automatically performs the source-to-assembly mapping operation and that is application and architecture independent.

VI. TRANSACTIONS INTERDEPENDENCY REFINEMENT

The first refinement that we operate aims at isolating and modeling the effect of transactions interdependency over the total application length and energy estimated by using IP Models and FUNTIME Models.

IP energy and performance models are implemented assuming an ideal condition where each transaction is completed in the least possible amount of cycles, provided that the transaction is not related to any previous one and that all what is required to make it happen (ex. instructions, data) is already available at the IP inputs before the transaction gets triggered. In this way, the amount of cycles finally associated to each transaction only depends on the intrinsic properties of the IP and is not related to what is around it. However, in real systems running real applications, such ideal conditions are generally not satisfied. This non-ideality can be interpreted as the injection of extra cycles besides the theoretical ones reported in the IP model.

For a processor IP, transactions are the instructions belonging to its instruction set, and the occurrence of extra cycles can

be interpreted as "stall" conditions implied by instructions/data dependency. As an example, in a system without cache, if a *load* instruction is issued right after a *store* to the same memory location, a stall condition will occur, since the data to be loaded has not yet been stored into memory. The goal of the present refinement is therefore to estimate how many such stalls occur when executing real applications, without running any architectural simulation, like gate level or TLM.

Our approach relies on the idea that, even if all applications are different from each other, their instructions are executed such a high number of times and in so many different combinations, that it makes sense and it is possible to find an average behavior, independent from the application chosen. The following empirical method can generally be used. A small number of reference applications is taken and used as a base for calibration. In detail, such applications are first run on a platform and the average number of cycles per instruction is collected for each such application. Afterwards, calibration is performed by taking a weighted average of the cycles per instruction over the reference applications. In each application, the weighting factor is given by the percentage of times that each instruction is issued over the total number of instructions. The resulting numbers are then taken as a model and applied to new applications to estimate the total applications length in terms of cycles. The difference between the total application cycles found by using the refined values and the total application cycles found by using the ideal values represents the contribution of the stalls and, consequently, the effect of transactions dependency. Since a stall basically corresponds to a state where the processor is not executing any useful operation, its behavior in terms of energy and cycles can be identified with the behavior of a *nop* instruction. By using Eq. 5, it is therefore possible to estimate how many stalls occur and to refine accordingly the transactions report produced by the FUNTIME Model.

$$\#Stall = \frac{Tot\_estimated\_cycles - Tot\_ideal\_cycles}{Cycles\_per\_NOP} \quad (5)$$

Similarly, the extra energy contribution due to the stalls can be calculated by using Eq. 6.

$$E_{stalls} = \#stalls \times E_{NOP} \quad (6)$$

In the context of our research, the method described above has been applied to a SoC based on a Leon3 processor (without cache) that exchanges information with a memory through an AMBA AHB bus. A set of 3 different applications (an FFT, a Quicksort and a Fibonacci series) has been taken as a reference and used for calibration. Table III shows a comparison between the ideal cycles and the refined ones for a bunch of Leon3 instructions. This approach to refining transactions interdependency has been validated and presented in Section VIII.

VII. Cache refinement

The presence of caches is a further element that affects the ideal numbers estimated by using IP Models and FUNTIME Models. The refinement that we intend to apply in this case aims at quantifying and modeling the variation of an application length and energy, compared to the case with no cache, given the cache miss ratio. As opposed to the effect of transactions interdependency, which adds extra cycles, caches reduce the total amount of application cycles, since there are fewer wait states caused by communication with memory.

TABLE III
Comparison between ideal and real average cycles per instr.

Instr.	Ideal cycles	Real avg. cycles
add	1.00	7.58
ld	1.00	15.19
nop	1.00	3.23
st	2.00	5.82
...	...	...

Also in this case, the way in which the refinement is implemented relies on the assumption that the percentage of cycles variation for a given cache miss ratio, compared to the case without cache, is independent from the application studied. This assumption is again sustained by the fact that an average behavior can be detected, as a consequence of the high number of instructions and instruction sequences generated by each application. Similarly to the previous refinement, a small set of applications can therefore be taken as a base for calibration and the results of calibration can then be applied to refine new applications. Calibration in this case consists in finding an interpolation function.

Referring to the same simple SoC (Leon3 with 1k, 2k, 4k cache) and the 3 reference applications as for the transactions interdependency refinement, the interpolation function results in this case in a straight line, expressed in Eq. 7:

$$\%\_Cyc = 1.09 \cdot Miss\_Ratio + 19.19 \quad (7)$$

where $\%\_Cyc$ represents the percentage of total cycles compared to the case without cache. Note that, for an ideal case of miss ratio = 0, an application would complete in about 20% of the time with respect to the case with no cache. Validation for cache refinement is also presented in Section VIII.

VIII. Validating the FUNTIME approach

The validation of the FUNTIME methodology has been carried out on a set of common benchmark applications, mapped to the same reference SoC used in the sections above: a Leon3 processor exchanges information with a memory through an AMBA AHB Bus. The validation consists of the following 2 subsections: in Subsection VIII-A, the Leon3 IP model (Section IV), the FUNTIME Model for software applications (Section V) and the refinements (Sections VI and VII) are validated against gate level for energy and timing estimation accuracy; Subsection VIII-B validates the FUNTIME methodology in terms of execution speed against TLM-PV.

A. FUNTIME vs. gate level for energy and timing accuracy

Gate level has been chosen as a reference for FUNTIME validation, since it represents the most accurate estimation

TABLE IV
1 SOURCE OF INACCURACY: LEON3 ENERGY & PERFORMANCE MODEL

	FFT	Quicksort	Fibonacci	Viterbi	Queens	Dhrystone	MD5
# Instructions	1 371 584	4 008 651	1 585 313	2 452 903	4 266 783	1 950 813	1 802 380
# Cycles	6 485 094	31 649 232	13 902 019	17 511 989	24 193 885	15 938 144	14 428 588
Real energy [mJ]	0.2396	1.1508	0.4906	0.6250	0.8571	0.5641	0.5228
Estimated energy [mJ]	0.2720	1.1490	0.4781	0.6597	0.9369	0.5788	0.5258
Error [%]	13.52	-0.16	-2.55	5.55	9.31	2.61	0.57

TABLE V
2 SOURCES OF INACCURACY: LEON3 ENERGY & PERFORMANCE MODEL + ESTIMATED NUMBER OF CYCLES (STALLS)

	Calibration benchmarks						
	FFT	Quicksort	Fibonacci	Viterbi	Queens	Dhrystone	MD5
# Instructions	1 371 584	4 008 651	1 585 313	2 452 903	4 266 783	1 950 813	1 802 380
Real # cycles	6 485 094	31 649 232	13 902 019	17 511 989	24 193 885	15 938 144	14 428 588
Estimated # cycles	7 217 187	31 153 927	13 251 730	18 484 266	24 878 824	15 660 388	14 082 568
Error [%]	11.29	-1.56	-4.68	5.55	2.83	-1.74	-2.40
Real energy [mJ]	0.2396	1.1508	0.4906	0.6250	0.8571	0.5641	0.5228
Estimated energy [mJ]	0.2923	1.1352	0.4600	0.6597	0.9559	0.5711	0.5162
Error [%]	22.01	-1.35	-6.23	9.87	11.52	1.24	-1.27

TABLE VI
3 SOURCES OF INACCURACY: LEON3 ENERGY & PERFORMANCE MODEL + ESTIMATED NR. OF CYCLES (STALLS) + ESTIMATED NR. OF INSTRUCTIONS

	Calibration benchmarks						
	FFT	Quicksort	Fibonacci	Viterbi	Queens	Dhrystone	MD5
Real # instructions	1 371 584	4 008 651	1 585 313	2 452 903	4 266 783	1 950 813	1 802 380
Estimated # instructions	1 273 927	3 824 253	1 517 196	2 398 390	4 006 083	1 880 389	1 707 214
Error [%]	-7.12	-4.6	-4.3	-2.22	-6.11	-3.61	-5.28
Real # cycles	6 485 094	31 649 232	13 902 019	17 511 989	24 193 885	15 938 144	14 428 588
Estimated # cycles	6 731 528	29 427 456	12 185 120	17 783 529	23 680 975	15 131 674	13 233 901
Error [%]	3.8	-7.02	-12.35	3.55	-2.12	-5.06	-8.28
Real energy [mJ]	0.2396	1.1508	0.4906	0.6250	0.8571	0.5641	0.5228
Estimated energy [mJ]	0.2734	1.0660	0.4145	0.6636	0.9280	0.5382	0.4803
Error [%]	14.1	-7.37	-15.51	6.18	8.27	-4.60	-8.13

methodology for SoCs. A first set of comparisons is presented in Tables IV, V and VI. These tables validate the accuracy of the FUNTIME Methodology for 7 benchmarks by introducing a growing number of sources of inaccuracy. Note that the first 3 applications from the left also represent the calibration suite referenced in the refinement sections. In all these 3 tables, the Leon3 configuration without cache has been selected.

In Table IV, the only source of inaccuracy introduced is the IP energy and performance model of the Leon3, while the number of actual cycles and instructions is measured. Therefore, no refinement and no FUNTIME Model is used. This table basically validates the accuracy of the Leon3 energy and performance model. Table V introduces 2 sources of inaccuracy: the first refers to the transactions interdependency refinement used to estimate the total number of cycles (ideal + stalls contribution); the second refers to the Leon3 energy and performance model. The number of actual instructions is measured. Finally, Table VI introduces 3 sources of inaccuracy: the first two are like in Table V, while the third refers to the estimated number of transactions (instructions) found by means of the FUNTIME Model. Note that, in this case, the number of transactions is always underestimated. This depends on the fact that, when associating source lines to assembly code, the FUNTIME Models cannot take into account the contributions of *calls* to external routines for which the assembly code is not directly available. This is for instance the case for common library functions, i.e. *malloc*, *strcpy*, *printf*, etc. We have been working on modeling and factoring in this contribution separately, but the process is not yet complete. Note that in Table VI, which introduces the highest number of sources of inaccuracy, the maximum error of FUNTIME for energy estimation is around 15% of the gate level. These results confirm the expectations shown in Fig. 1.

Validation for the cache refinement is instead presented in Fig. 3. In this case, the Eq. 7 found as an interpolation for the calibration benchmarks has been used to estimate the cycles variation also for the other 4 benchmarks on the right when using a Leon3 with 1k, 2k, 4k cache. Fig. 4 shows both the interpolation straight line extracted from the calibration benchmarks (dashed line), and the real interpolation straight line for the other 4 applications (thick line). The small difference between the two proves that using Eq. 7 as a reference for all the applications is sensible and the error introduced is minimal.

B. FUNTIME vs. TLM-PV for execution speed

For speed comparison, TLM-PV has been chosen as a reference. The reason is that this represents the fastest high-

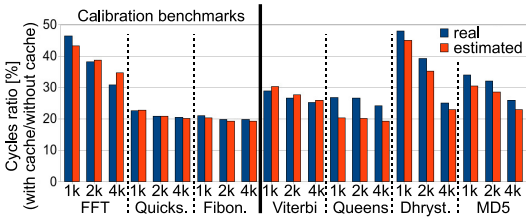


Fig. 3. Validating the cache refinement

TABLE VII
TIMING COMPARISON BETWEEN FUNTIME AND TLM-PV

	#ALTs	TLM-PV	FUNTIME	Speedup
Jpeg2k_128x128	82M	3.00 sec	0.62 sec	5
Jpeg2k_256x256	272M	10.30 sec	0.69 sec	15
Jpeg2k_512x512	986M	38.66 sec	1.04 sec	37
H264_176x144	1.68B	50.75 sec	1.41 sec	36
H264_352x288	2.55B	79.80 sec	1.58 sec	51

level methodology for system-level estimation commonly used at present. For this purpose, we built our own TLM-PV in SystemC for the reference SoC architecture. The implementation has been as abstract as possible, since it exclusively represents the transactions occurring across the platform among the different IPs. In addition, communication is handled using bidirectional blocking interfaces. For this experiment, a set of applications has been chosen with a number of executed instructions much higher than the benchmark applications of the previous experiment. The reason is to make the difference between FUNTIME and TLM-PV execution speed more evident. The applications chosen are the image compression codec JPEG2000 and the video compression codec H264. The first has been applied to three images of sizes 128x128, 256x256 and 512x512. The second has been used for two videos of 3 frames each, with resolution of 176x144 and 352x288. The number of total executed instructions ranges from 80 milion to 1.6 billion. The results of this comparison are reported in Table VII and show a mean speed improvement of 28 times for FUNTIME compared to the TLM-PV. Besides, note that the advantage of FUNTIME over TLM-PV increases as the number of instructions increases. This confirms the capacity of FUNTIME to be used for complex and real use-case scenarios, and is consistent with the speedup expectations of Fig. 1.

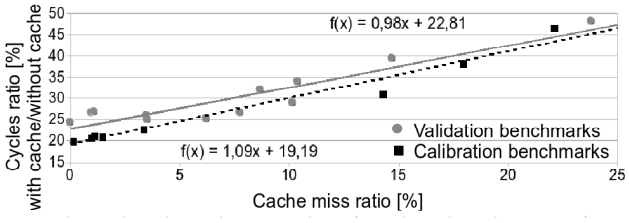


Fig. 4. Cycles variation as a function of the cache miss ratio

IX. A REAL-CASE EXAMPLE

To demonstrate the capacity of the FUNTIME Methodology, FUNTIME has been used to estimate energy and execution time for a real use case of encoding a full movie (608x336 for 193 204 frames) using the H264 video codec.

It took about 28 hours to execute the application on the host machine. This is because the codec was compiled with no optimization. On the other hand, the instrumentation and the generation of energy/timing reports only took 0.72 secs. By using the values in Table VII, it is reasonable to estimate that the same experiment performed in TLM-PV with the simplified CPU model would take around 128 hours.

In Fig. 5, we report energy figures for the Leon3 processor, the AHB Ctrl and a 16k memory, as well as the estimated execution time of the application, assuming that the SoC is clocked at 400MHz. IP-level energy and performance Models have been implemented both for the Leon3 and the AHB Ctrl, while Cacti [8] has been used for memory energy estimation.

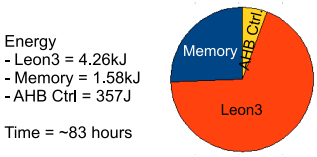


Fig. 5. Energy distribution

X. CONCLUSION AND FUTURE WORK

We have described a high-level approach to perform efficient and accurate energy and performance estimations in SoCs. We have based the presentation on 3 different levels of abstraction: IP Level, FUNTIME Level and Refinement Level. We have then validated the methodology against gate level for estimation accuracy, and against TLM-PV for estimation speed. The results give an accuracy of FUNTIME within 15%, and a mean speedup around 28x. We are presently working on extending and completing the refinement process to also account for the implications due to bus contention, scheduling, power management policies and resource sharing issues in general. The generation of FUNTIME Models for hardware applications is also part of our future work.

REFERENCES

[1] S. Penolazzi, M. Badawi, and A. Hemani, "A Step Beyond TLM Inferring Architectural Transactions at Functional Untimed Level," in *VLSI-SoC*, Rhodes, Greece, 2008.

[2] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. Sangiovanni-Vincentelli, "Metropolis: an integrated electronic system design environment," *Computer*, vol. 36, no. 4, pp. 45–52, April 2003.

[3] P. Lieverse, P. V. D. Wolf, K. Vissers, and E. Deprettere, "A methodology for architecture exploration of heterogeneous signal processing systems," *J. VLSI Signal Process. Syst.*, vol. 29, no. 3, pp. 197–207, 2001.

[4] D. D. Gajski, Z. Jianwen, R. Domer, A. Gerstlauer, and Z. Shuqing, *SpecC: Specification Language and Methodology*. Springer, 2000.

[5] T. Wild, A. Herkersdorf, and G.-Y. Lee, *Design Automation for Embedded Systems*. Springer, 2006, vol. 10, ch. TAPES-Trace-based architecture performance evaluation with SystemC.

[6] *SPIRIT User Guide v1.2*, 2006.

[7] L. Bolognino, "Synthesis and Power Modeling of a Leon3 SPARC V8 Processor," Master's thesis, Royal Institute of Technology (KTH), 2008.

[8] S. Thoziyoor, N. Muralimanohar, and N. P. Jouppi, "Cacti 5.0," HP Laboratories, Tech. Rep., 2007.

2009 22nd International Conference on VLSI Design

Exploiting Hybrid Analysis in solving Electrical Networks

V. Siva Sankar, H. Narayanan, Sachin B. Patkar
{sivasankar,hn,patkar}@ee.iitb.ac.in
Department of Electrical Engineering,
Indian Institute of Technology, Bombay,
Mumbai-400076,
India

Abstract

In this paper we use topological hybrid analysis (mixture of nodal analysis and loop analysis) to solve circuits with resistors, voltage sources, current sources and diodes with exponential characteristics. In topological hybrid analysis [3], from the given network two smaller circuits are derived and solved simultaneously satisfying certain boundary conditions and this results in a solution of the original network.

Our main emphasis is on non planar circuits with a large conductance range. The reason for this is that for non planar circuits preconditioned Conjugate Gradient method seems to perform very well but its convergence will be adversely affected once the ratio of maximum to minimum conductance becomes as high as 10^8. To overcome this problem we use Hybrid analysis and a variation of Conjugate Gradient method. Using this method we analyzed circuits containing resistors with large range of values, voltage sources and current sources and having size up to 1 million nodes and 3 million edges on 3GHZ pentium IV processor with 2GB RAM in less than 4 minutes. Also, we report the simulation timings for circuits containing diodes.

1. Introduction

The static DC analyzer is at the core of a circuit simulator. As it decides the performance of the simulator, speeding it up is of paramount importance. The work reported in this paper is part of an ongoing effort [5, 6] at the Electrical Department, IIT Bombay towards building specialized circuit simulators that can handle large scale (10^6 nodes) circuits with diodes, resistors, current and voltage sources. In earlier work we have reported good results with planar linear circuits on which sparse LU and cholesky decomposition are very successful. Through the Newton-Raphson (NR) technique, we can then solve planar diode circuits which at each iteration reduce to linear circuits. However non planar circuits present serious difficulties since sparse LU performs poorly beyond $20K$ node size circuits and cholesky decomposition beyond about $100K$. On the other hand, the conjugate gradient method (CG) performs excellently well if the ratio of maximum to minimum conductance is in the range $1 - 10^4$. But, in practice exponential diode circuits during NR iteration produce the ratio of maximum to minimum conductance as 10^9 or 10^{10}. This is correlated with high condition number and CG essentially does not converge.

Curiously, a method which has largely been dismissed as purely of theoretical interest appears to give satisfactory practical results for this problem. This is the hybrid analysis of Kron *topologically* generalized as in [1, 3]. Essentially by scaling the values of resistors one puts them in the range 10^{-k} to $10^k\Omega$ (k=4, adequate for us). The resistors are partitioned into those in $1 - 10^4\Omega$ range and those in 10^{-4} to 1Ω. The method treats the former as resistors in loop analysis and the latter as conductances in nodal analysis. Two networks denoted by $\mathcal{N}_{AL}$ and $\mathcal{N}_{BK}$ are appropriately constructed on these two groups. We write nodal analysis equations for $\mathcal{N}_{AL}$ network and loop analysis equations for $\mathcal{N}_{BK}$ network and couple them in terms of boundary current/voltage variables. This results in a coefficient matrix of the form

$$\begin{bmatrix} A_r G A_r^T & H^T \\ -H & BRB^T \end{bmatrix}$$

where the top left block is the node conductance matrix of $\mathcal{N}_{AL}$ and the bottom right, the loop resistance matrix of $\mathcal{N}_{BK}$. This matrix is positive definite but not symmetric. A variant of CG (called Modified CG [1]) works well for this case and the convergence behavior is as though we are working with a network where the range of resistances is only $1 - 10^4\Omega$. For randomly generated non planar circuits with exponential diodes, resistors, voltage and current sources, this method is very effective (see section 6).

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.27

Brief description of topological hybrid analysis and MCG method are given in section 2. Section 3 presents an approach to obtain an electrical equivalent circuit from a specific set of linear equations and solve the circuit to get the solution of the linear system of equations. Experimental results are discussed in section 4. Conclusions are given in section 5.

2 Preliminaries

Let $\mathcal{G}$ be a graph on the set of edges S. A forest of $\mathcal{G}$ is the maximal set of branches of $\mathcal{G}$ which contains no loops. If the graph is connected (i.e in a single connected piece (called component)) the forest of $\mathcal{G}$ would be called a tree of $\mathcal{G}$. Let $P \subseteq S$. By $\mathcal{G} \times P$, the contraction of $\mathcal{G}$ on P, we mean the graph obtained by fusing the end points of edges in $S - P$ and deleting them. By $\mathcal{G} \circ P$, the restriction of $\mathcal{G}$ on P, we mean the graph obtained by open circuiting (deleting) all the edges in $S - P$ and retaining only the nodes which are end points of edges in P.

3 Hybrid Analysis

The solution of a network with voltage sources and other devices can be reduced to one without voltage sources by the simple and fast procedure of voltage-shift [4]. We will therefore assume that our linear network has only resistors and current sources. We will further assume that our circuit is non planar. As mentioned before, direct methods (sparse LU and cholesky decomposition) are too slow beyond 100K nodes. Preconditioned conjugate gradient method does not work well if the ratio of maximum to minimum conductance is higher than 10^4. Topological hybrid analysis combined with modified CG seems tailor made for this situation [2, 3]. We will call it the $\mathcal{N}_{AL} - \mathcal{N}_{BK}$ method. The steps involved are as follows:

Let the given network be $\mathcal{N}$ with graph $\mathcal{G}$ as in Figure 1. The devices in $\mathcal{N}$ are decomposed into A and B. The only requirement is that the characteristics of the devices in A and B are independent of each other. In our case we scale the resistances so that they are in the range 10^{-k} to 1 (A branches) and 1 to 10^k (B branches). Let t_A be a tree (or forest) of the sub graph of $\mathcal{G}$ on A. Extend t_A to a tree (or forest) t of the graph $\mathcal{G}$. Call as K the branches in t_A which lie in the fundamental circuit of a $B - t$ branch with respect to the tree t. Call as L the branches in $B - t$ whose fundamental circuits with respect to t contain branches of t_A. Now build the network $\mathcal{N}_{AL}$ on $\mathcal{G} \times (A \cup L)$ as shown in Figure 2 with device characteristics of A as in $\mathcal{N}$ but devices in L with no constraints (can be thought intuitively as unknown current sources). Next build the network $\mathcal{N}_{BK}$ on $\mathcal{G} \circ (B \cup K)$ (see Figure 3).

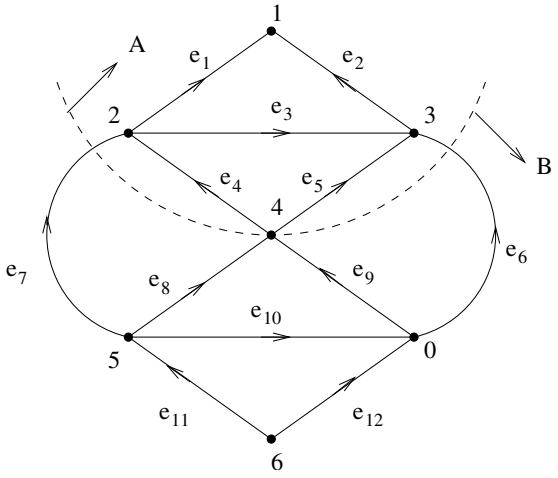


Figure 1. Given Network $\mathcal{N}$

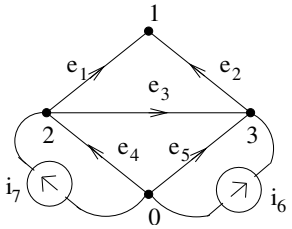


Figure 2. Network $\mathcal{N}_{AL}$

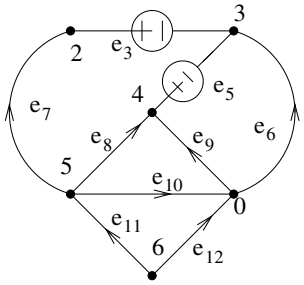


Figure 3. Network $\mathcal{N}_{BK}$

The result in [2, 3] states that solving $\mathcal{N}$ is equivalent to solving $\mathcal{N}_{AL}$ and $\mathcal{N}_{BK}$ simultaneously matching *the boundary conditions* by making i_L the same in both and v_K the same in both.

In our case we assume every branch to be composite as in Figure 4. The device characteristic can be seen to be

$$i - J = G(v - E) \tag{1}$$

$$v - E = R(i - J) \tag{2}$$

We then write nodal analysis equations for $\mathcal{N}_{AL}$ taking

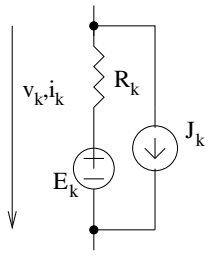


Figure 4. Composite branch

i_L to be current sources yielding the equations

$$A_{rA}G_AA_{rA}^T v_{n_A} + A_{rL}i_L = -A_rJ_A + A_rG_AE_A$$

We write loop analysis equations for $\mathcal{N}_{BK}$ keeping v_K as voltage sources.

$$B_BR_BB_B^T i_{1_B} + B_K v_k = -B_BE_B + B_BR_BJ_B$$

Here $\begin{bmatrix} B_B & B_K \end{bmatrix}$ is a fundamental circuit matrix of $\mathcal{G}\circ(B\cup K)$ and $\begin{bmatrix} A_{rA} & A_{rL} \end{bmatrix}$ is a reduced incidence matrix of $\mathcal{G} \times (A \cup L)$. We denote by B_L the sub matrix of B_B with all rows but columns corresponding to L and by A_{rK} the sub matrix of A_{rA} with all rows but columns corresponding to K. Matching i_L, v_K for both yields $i_L = B_L^T i_{l_B}$, $v_K = A_{rK}^T v_{nA}$.

So finally we get equations of the form

$$\begin{bmatrix} A_{rA}G_AA_{rA}^T & A_{rL}B_L^T \\ B_KA_{rK}^T & B_BR_BB_B^T \end{bmatrix} \begin{bmatrix} v_{n_A} \\ i_{1_B} \end{bmatrix} = \begin{bmatrix} -A_rJ_A + A_rG_AE_A \\ -B_BE_B + B_BR_BJ_B \end{bmatrix} \quad (3)$$

Solving which we get node voltages of $\mathcal{N}_{AL}$ and loop currents of $\mathcal{N}_{BK}$. The branch voltages v_A and branch currents i_B can be obtained through KCL and KVL and i_A and v_B are obtained using device characteristics.

3.1 Modified Conjugate Gradient Method (MCG)[1]

The steps for this method are the same as preconditioned CG except for one difference. Whenever we need to compute the *dot product* $< x_1\ x_2,\ y_1\ y_2 >^\sim$ of vectors (x_1, x_2) and (y_1, y_2) partitioned consistently with the partition in 3, we compute it as $x_1^T y_1 - x_2^T y_2$ rather than $x_1^T y_1 + x_2^T y_2$. With this definition of *dot product* modified CG is as follows; To solve $\hat{A}x = b$, with a preconditioned matrix M

- Given an initial guess x_0, take $r_0 = b - \hat{A}x_0$, solve $Mz_0 = r_0$.
- Set $p_0 = z_0$.
- For k=1,2,3....
- compute $\hat{A}p_{k-1}$.
- set $x_k = x_{k-1} + a_{k-1}p_{k-1}$, where $a_{k-1} = \frac{<r_{k-1},z_{k-1}>^\sim}{<p_{k-1},\hat{A}p_{k-1}>}$
- compute $r_k = r_{k-1} - a_{k-1}\hat{A}p_{k-1}$.
- solve $Mz_k = r_k$ and set $p_k = z_k + b_{k-1}p_{k-1}$ where $b_{k-1} = \frac{<r_k,z_k>^\sim}{<r_{k-1},z_{k-1}>^\sim}$.

In [1] it is shown that this method converges provided it does not encounter any vector x such that $< x, x >^\sim = 0$ during the progress of the algorithm.

4 Adapting MCG to Hybrid Analysis

The key step in the adaptation is to interpret matrix vector multiplication Ap_{k-1} and the solution of $Mz_k = r_k$, graph theoretically or network theoretically. In our case the pre conditioner is

$$M = \begin{bmatrix} A_{rA}G_AA_{rA}^T & 0 \\ 0 & B_BR_BB_B^T \end{bmatrix}$$

The solution of $Mz_k = r_k$ can be interpreted as the solution of two decoupled networks on $\mathcal{G}\circ A$ and $\mathcal{G}\times B$ respectively interpreting r_k as current or voltage sources. There is no need to compute the coefficient matrix in 3 or the matrix M explicitly. One should just have the appropriate network available while multiplying $\begin{bmatrix} A_{rA}G_AA_{rA}^T & A_{rL}B_L^T \\ B_KA_{rK}^T & B_BR_BB_B^T \end{bmatrix} \begin{bmatrix} v_{n_A} \\ i_{1_B} \end{bmatrix}$. Since H is a sparse matrix containing 0 and 1 as elements, we need to worry about computing $(A_{rA}G_AA_{rA}^T)v_{n_A}$ and $(B_BR_BB_B^T)i_{1_B}$ only. Consider the former. First, $A_{rA}^T v_{n_A}$ is the branch voltage vector corresponding to the node potential vector v_{n_A} (with respect to a datum node) in the graph $\mathcal{G}\circ A$. This can be done graph theoretically. $G_A(A_{rA}^T v_{n_A})$ only involves scaling the components of this vector using the (diagonal) entries of G_A. Treating this as a branch current vector $A_{rA}[G_AA_{rA}^T v_{n_A}]$ yields the net current leaving each node. This again can be done graph theoretically.

Next consider $(B_BR_BB_B^T)i_{1_B}$. B_B is the fundamental circuit matrix of $\mathcal{G}\times B$ with respect to a tree t_B and a co tree L_B. Then $B_B^T i_{1_B}$ is the branch current vector corresponding to co tree current vector i_{1_B}. As before, multiplying by R_B amounts to scaling the vector. Computing $B_B[R_BB_B^T i_{1_B}] = B_BY_B$ say. This can also be given a graph theoretical interpretation. If $B_B = [I\ B_{12}]$ and

$Y_B = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$ then $B_B Y_B = [I\ B_{12}] \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \mathcal{E}_1$ say, i.e $[I\ B_{12}] \begin{bmatrix} Y_1 - \mathcal{E}_1 \\ Y_2 \end{bmatrix} = 0$. The vector $Y_1 - \mathcal{E}_1$ is the co tree voltage corresponding to the tree voltage Y_2 in $\mathcal{G} \times B$. Since Y_1 is already known $\mathcal{E}_1$ can be computed.

5 Approach to solve a specific set of linear equations

To solve the equation 3 using modified CG we experimented with Block diagonal preconditioning. Here in each iteration of MCG, it is necessary to solve

$$M \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix} \tag{4}$$

where

$$M = \begin{bmatrix} A_{rA} G_A A_{rA}^T & 0 \\ 0 & B_B R_B B_B^T \end{bmatrix}$$

As explained earlier, in order to solve the equation 4 we have to solve two decoupled networks. They are

1. An RJ circuit ($\mathcal{N}_{AL}$): The system of equations to be solved here is

$$A_{rA} G_A A_{rA}^T z_1 = r_1 \tag{5}$$

Solving the equation 5 is equivalent to solving the RJ circuit obtained from $\mathcal{N}_{AL}$ network after open circuiting L branches and treating r_1 as current sources. To illustrate it, let $\mathcal{N}_{AL}$ graph be as shown in Figure 5.

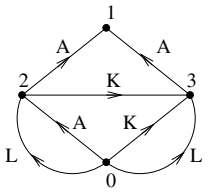


Figure 5. $\mathcal{N}_{AL}$ graph

With respect to $\mathcal{N}_{AL}$ graph, we have to solve

$$A_{rA} G_A A_{rA}^T z_1 = \begin{bmatrix} r_{11} \\ r_{12} \\ r_{13} \end{bmatrix} \tag{6}$$

To solve the equation 6 we solve the equivalent RJ network shown in Figure 6. For the circuits shown in Figures 2 and 6, node 0 is taken as datum node.

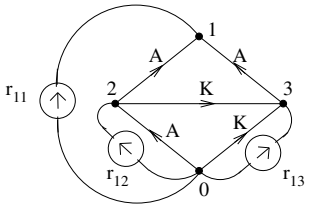


Figure 6. RJ circuit equivalent to the equation 6

2. An RV circuit ($\mathcal{N}_{BK}$): The system of equations to be solved here is

$$B_B R_B B_B^T z_2 = r_2 \tag{7}$$

We do not solve the equation 7 directly since the coefficient matrix is dense. Instead, we form an equivalent electrical network and solve it through nodal analysis. Solving the equation 7 is equivalent to solving the RV circuit obtained from $\mathcal{N}_{BK}$ network after short circuiting the K branches and treating r_2 as voltage sources in series with co tree branches.

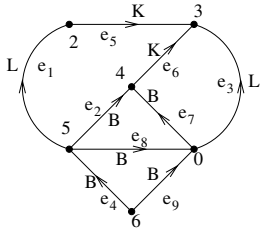


Figure 7. $\mathcal{N}_{BK}$ graph

To illustrate it, let $\mathcal{N}_{BK}$ graph be as shown in Figure 7. Figure 3 has been renumbered for convenience. Let the branches e_5, e_6, e_7, e_8 and e_9 form a tree for the network $\mathcal{N}_{BK}$. The branches e_1, e_2, e_3 and e_4 form the corresponding co tree. Now with respect to the circuit shown in Figure 7 we will have to solve

$$B_B R_B B_B^T \begin{bmatrix} z_{21} \\ z_{22} \\ z_{23} \\ z_{24} \end{bmatrix} = \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \\ r_{24} \end{bmatrix} \tag{8}$$

Where $z_{2k} = i_{e_k}$ for k=1,2,3 and 4.

Solution of the equation 8 can be obtained by solving the RV network shown in Figure 8.

Now a resistance in series with a voltage source can be converted into a resistance in parallel with a current source without changing the terminal behavior.

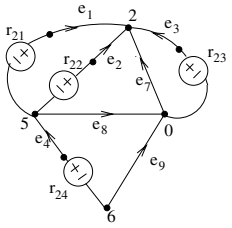


Figure 8. RV circuit equivalent to the equation 8

By applying this concept, the RV circuit shown in Figure 8 can be converted into RJ circuit which appears as shown in Figure 9.

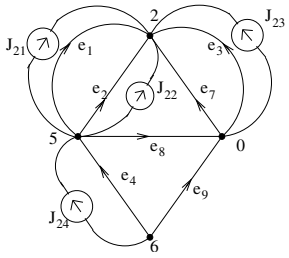


Figure 9. RJ circuit equivalent to the RV circuit shown in figure 8

Now we can perform nodal analysis for the circuit shown in Figure 9 and subsequently get the solution of circuit in Figure 8.

6 Experimental Results

In this section, we present simulation times for randomly generated simple non planar circuits. The number of nodes and the number of edges for every circuit are in the ratio of 1:3. For every circuit, the number of voltage sources and current sources each are 20% of the number of edges respectively. Nodes of $\mathcal{G}$ is the number of Nodes in the generated non planar circuit. In Table 1, N_{Iter}^{diagCG} refers to the number of iterations taken by diagonal matrix based pre conditioner CG method when the entire circuit is solved by nodal analysis and t_{dcAna}^{diagCG}, the total time taken for solving the circuit. In Tables 2, 3 and 4, N_{Iter}^{MCG} refers to the number of iterations taken by MCG method in hybrid analysis approach and t_{dcAna}^{MCG}, the total time taken to solve the circuit.

Experiments in all the tables are performed on 3GHZ Pentium IV processor with 2GB RAM. The solution for a network will be said to be obtained if the following constraints are met.

Nodes of $\mathcal{G}$	N_{Iter}^{diagCG}	t_{dcAna}^{diagCG}
10K	413	0.5
30K	1676	3.13
50K	2467	7.38
80K	3480	17.1
100K	5290	34.81
200K	8401	123.61

Table 1. For non planar resistive circuits, timing results of PCG-DC Analyzer when the resistance range is 1 Ω to 10^8 Ω

1. KCE at each node is satisfied within a tolerance termed as *toleranceforKCL*. The value of *toleranceforKCL* for the tables 2, 3 and 4 where hybrid analysis is used, is 10^{-4} times the maximum value of current source i.e if i_j is the net leaving current at node j then $i_j <$ *toleranceforKCL*.
2. KVE is exactly satisfied for all the tables.
3. A device characteristic is said to be verified if $\Delta <$ *tolForDevChar*, where *tolForDevChar* $= 0.01$. Δ is relative error in voltage across or current in the diode depending on reverse biasing or forward biasing.

The circuit with resistance range 1 Ω to 10^8 Ω has been generated as follows. First generate a non planar circuit randomly with resistances picked in the range from 1 Ω to 10^2 Ω. Then 30% of the resistive edges are reassigned to 1Ω and 30% to 10^8 Ω.

We can draw the following conclusion from the results given in Table 1. Whenever there is a large range of conductances in a non planar circuit, Preconditioned CG based DC Analyzer exhibits poor performance. This problem arises while solving non linear circuits like diode circuits and combinatorial optimization problems as explained earlier.

It is already mentioned that the performance of Conjugate Gradient method has a continuing deterioration with the increase in the range of conductances. But interestingly, from Tables 2 and 3 we can observe that hybrid analysis based approach does not exhibit such deterioration.

In Table 4, the circuits contain practical diodes (with exponential characteristics). The number of diodes is 10% of the total edges in the circuit. The index Total-NR iter refers to the number of Newton-Raphson iterations for linearizing non linear elements. The exponential characteristic of a diode can be represented by the equation

$$i = I_s(\exp \frac{v}{V_T} - 1)$$

Nodes of $\mathcal{G}$	$1-10^8\Omega$	
	N_{Iter}^{MCG}	t_{dcAna}^{MCG}
100K	7	10.35
200K	6	22.4
300K	6	42.4
400K	6	58.01
500K	6	73.02
800K	6	155.62
1000K	6	200.99

Table 2. For non planar resistive circuits, timing results of hybrid analysis based DC Analyzer when the resistance range is $1-10^8\Omega$

Nodes of $\mathcal{G}$	$1-10^{10}\Omega$	
	N_{Iter}^{MCG}	t_{dcAna}^{MCG}
100K	5	9.11
200K	5	21.31
300K	5	39.25
400K	5	55.23
500K	5	73.03
800K	5	144.34
1000K	5	181.37

Table 3. For non planar resistive circuits, timing results of hybrid analysis based DC Analyzer when the resistance range is $1-10^{10}\Omega$

where i and v are current through and voltage across the diode branch. We have chosen $I_s = 10^{-10}A$ and $V_T = 0.0259V$. While solving the diode circuits, depending on the range of conductances either hybrid analysis technique or Nodal Analysis technique is used in our DC analyzer. From Tables 2, 3 and 4 we can observe that the problem of large conductance range can be tackled by hybrid analysis approach and modified CG. The reason for the near quadratic (time) behavior from $100K$ node circuit to $1000K$ node circuit is the varying performance of modified CG with varying distribution of conductance values across NR iterations.

7 Summary and Conclusions

The main difficulty is handling practical diode circuits where the range of conductances is very large because then the convergence rate of diagonal preconditioned CG method deteriorates adversely. This problem was faced when we tried to solve a non planar min cost flow problem using electrical techniques. We observed that this problem can be overcome by hybrid analysis approach along with MCG method. This technique has strengthened our belief that we can tackle even very large size non planar min cost flow problems by electrical networks.

Nodes of $\mathcal{G}$	Total-NR iter	t_{dcAna}^{MCG}
100K	22	185.23
300K	24	1150.12
500K	25	2728.96
700K	30	5126.64
1000K	27	8044.54

Table 4. For non planar diode circuits, timing results of hybrid analysis based dc Analyzer

References

[1] H. Narayanan, "Mathematical Programming and Electrical Network Analysis II: Computational Linear Algebra through Network analysis," *International Symposium on Mathematical Programming for Decision Making: Theory and Applications (ISMPDM07), ISI Delhi*, January 10-11, 2007.

[2] H. Narayanan,"A theorem on graphs and its application to network analysis," *Proceedings of IEEE International Symposium on Circuits and Systems*, pp. 1008-1011, 1979.

[3] H. Narayanan, *Submodular Functions and Electrical Networks, Annals of Discrete Mathematics*, vol. 54. North Holland, Amsterdam, The Netherlands, 1997.

[4] S. H. Batterywala and H. Narayanan, "Efficient DC Analysis of RVJ Circuits for Moment and Derivative Computations of Interconnect Networks," *12th International Conference on VLSI Design*, pp. 169-174, 1999.

[5] G. Trivedi, M. P. Desai and H. Narayanan, "Fast dc analysis and its application to combinatorial optimization problems," *International Conference on VLSI Design, India*, January 2006.

[6] G. Trivedi, Sumit Punglia and H. Narayanan, "Application of DC Analyzer to combinatorial optimization problems," *20th International Conference on VLSI Design*, 2007.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, USA, 1990.

Session 3C

Test Generation

2009 22nd International Conference on VLSI Design

The Effect of Filling the Unspecified Values of a Test Set on the Test Set Quality

Irith Pomeranz[1]
School of Electrical & Computer Eng.
Purdue University
W. Lafayette, IN 47907
pomeranz@ecn.purdue.edu

and

Sudhakar M. Reddy[2]
Electrical & Computer Eng. Dept.
University of Iowa
Iowa City, IA 52242
reddy@engineering.uiowa.edu

Abstract

Test generation and test data compression processes create test vectors with unspecified input values that need to be filled. We study the extent to which filling the unspecified input values affects the untargeted fault coverage of a test set. To make the study independent of any particular test generation or test data compression scheme, we consider test sets for stuck-at faults that are obtained by first unspecifying as many values as possible without losing stuck-at fault coverage, and then filling the unspecified values randomly. The results indicate that there are significant differences in the untargeted fault coverage between different test sets. The differences in the average number of detections of stuck-at faults are less noticeable. We also show that adding a small fraction of untargeted faults to the set of faults considered during the unspecifying process improves significantly the untargeted fault coverage after filling of unspecified values.

1. Introduction

Test generation processes [1]-[6] typically leave unspecified input values that need to be filled. Compressed test data also contains unspecified values that are filled by test data decompression logic [7]-[8]. Values may be filled randomly, or based on considerations such as power dissipation [9]-[12]. Certain decompression schemes impose additional constraints on the filled values, for example, in broadcast scan [13], all the scan chains that are driven from the same scan input are filled with the same values.

The quality of a test set after filling its unspecified values can be measured by considering the numbers of detections of stuck-at faults [14]. This is motivated by the following observations. It was shown earlier that when a stuck-at fault is detected multiple times, the likelihood of detecting defects associated with the fault site increases [15]-[20]. Based on these results, the number of detections of stuck-at faults can be used to measure the quality of a test set. However, it was also shown that different tests for a fault may be different in ways that are not relevant to the detection of the fault or the defects associated with its site [21]. For example, consider a fault f in the input cone of an output z, which is detected on z. Inputs that are not in the input cone of z may be changed to create multiple tests for f, without creating different conditions around the site of f. In this case, the tests may not detect different defects. As a result, it is possible to reach a high number of detections for a given stuck-at fault without detecting the fault in fundamentally different ways. Thus, the number of detections may not be sufficient for evaluating the quality of a test set.

In this work we study the quality of a test set after its unspecified values are filled. To make the study independent of any particular test generation or test data compression and decompression scheme, we use an experiment that proceeds as follows. We start from a fully-specified test set T for single stuck-at faults. We use a procedure similar to the ones in [22]-[24] to unspecify as many values as possible in T. We repeat this procedure 10 times to obtain 10 incompletely-specified test sets $T_0, T_1, \cdots, T_9$ that detect the same set of single stuck-at faults as T. We then fill the unspecified values of each test set T_i 10 times randomly to obtain 100 fully-specified test sets $T_{i,0}, T_{i,1}, \cdots, T_{i,9}$, for $0 \le i \le 9$. For every $T_{i,j}$, we compute the average number of detections of single stuck-at faults, and the coverage of a set of faults that were not targeted during the test generation, unspecifying or filling process of T. The untargeted faults we consider are bridging faults. In this experiment, $T_{i,j}$ is a test set that may be obtained by test generation after filling unspecified values, or by compressing and then decompressing a test set T.

The fault sets used in this experiment are described in Section 2. The processes of unspecifying a test set and filling its unspecified values are described in Section 3. The experiment and its results are described in Section 4. The results indicate that there are significant differences in the untargeted fault coverage between the various test sets $T_{i,j}$, for $0 \le i \le 9$ and $0 \le j \le 9$. The differences are less noticeable from the average number of detections of stuck-at faults.

A way to address these issues is to expand the set of faults targeted, directly or indirectly, during the test generation or compression process, where unspecified values are created, or when the unspecified values are filled. In

1. Research supported in part by SRC Grant No. 2007-TJ-1643.
2. Research supported in part by SRC Grant No. 2007-TJ-1642.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.11

Section 5 we study the effects of adding bridging faults to the set of target faults of the unspecifying process. This study shows that including a small fraction of bridging faults as targets reduces the effects of random filling considerably. This is demonstrated by the increase in the coverage of untargeted bridging faults after unspecified values are filled. To reduce the effect of random filling on the test quality it is also possible to consider bridging faults indirectly, as done, for example, in [25].

2. Fault sets

We consider two sets of target faults. F_{sa} is the set of collapsed single stuck-at faults. F_{br} consists of four-way bridging faults [26]-[27], and it is defined as follows.

A four-way bridging fault is associated with two lines, g_1 and g_2, and a value a. The fault is denoted by (g_1=a ,g_2=a'). The fault is activated on g_1 when $g_1 = a$ and $g_2 = a'$. A test for the fault detects the fault g_1 stuck-at a' while setting $g_2 = a'$.

The number of four-way bridging faults in a circuit may be very high, and many of the faults may be easy-to-detect [28]. In addition, some of the faults may be undetectable. Therefore, we include in F_{br} a subset of the non-feedback four-way bridging faults. The subset is selected as follows to limit its size while ensuring that the faults it contains are detectable but not easy-to-detect.

Initially, $F_{br} = \phi$. For every line g_1 which is not a fanout branch and for every value $a \in \{0,1\}$, we find the set of lines G_2 that consists of every line g_2 such that there is no path in the circuit between g_1 and g_2 (in either direction). If the size of G_2 is larger than 100, we remove lines from G_2 randomly until its size reaches 100. For every line g_2 that remains in G_2, we include in F_{br} the fault (g_1=a ,g_2=a').

We then reduce the size of F_{br} by fault simulation as follows. Let T be a test set for F_{sa}. We perform fault simulation of F_{br} under T. We remove from F_{br} faults that are not detected by T. The faults left in F_{br} are detectable (they are detected by T).

To remove easy-to-detect faults from F_{br}, we consider up to 2000 random input vectors. For every random input vector r, we perform fault simulation of F_{br} under r with fault dropping. We continue to simulate random input vectors only as long as the size of F_{br} is higher than $2L$, where L is the number of circuit lines.

We are left in F_{br} with faults that are detected by T, and are not easy-to-detect. If the number of faults in F_{br} is higher than $2L$, we select $2L$ faults randomly to keep in F_{br}, and we remove the remaining faults.

3. Unspecifying and filling processes

In this section we outline the processes we use for unspecifying a test set and for filling its unspecified values.

We consider a test set T and a set of target faults F. We denote the number of tests in T by m, and the number of circuit inputs by n. Test i is denoted by t_i. The value of input j under test t_i is denoted by $t_i(j)$.

The unspecifying process we use is the one from [24]. We first perform fault simulation of F under T with fault dropping. We denote by F_i the set of faults detected by t_i during this process, for $0 \leq i < m$.

During the unspecifying process we consider every value $t_i(j)$ of T separately, for $0 \leq i < m$ and $0 \leq j < n$. When $t_i(j)$ is considered, we first unspecify it by setting $t_i(j) = x$. We then simulate all the faults in F_i. If all the faults in F_i continue to be detected by t_i we accept the change. Otherwise we restore the original value of $t_i(j)$.

Only the detection of faults included in F_i may be affected by the change in $t_i(j)$ (all the other faults continue to be detected by the other test vectors that have not been changed). Therefore, we only simulate the faults in F_i. We simulate them only under t_i, although some of these faults may also be detected by other test vectors. This is done to restrict the simulation effort.

To obtain different incompletely-specified test sets from a given test set T, we reorder the test vectors of T in different ways. As a result, the test vectors are simulated in different orders, and the sets of detected faults F_i are different. This results in different unspecified values at the end of the procedure.

To reorder a test set T, we swap every test vector $t_i \in T$ with a randomly selected vector $t_j \in T$. Specifically, for every vector t_i where $i = 0,1,\cdots,m-1$, we select a vector t_j randomly, where $0 \leq j < m$. We then swap t_i and t_j.

To fill an incompletely-specified test set T, we select a random value for every unspecified value of T. We select different sets of random values in order to obtain different fully-specified test sets based on T.

4. Effects of unspecifying and filling processes

In the experiment described in this section, we apply the procedure of unspecifying a test set T for stuck-at faults 10 times, each time reordering the test vectors differently, and using F_{sa} as the set of target faults. We obtain 10 incompletely-specified test sets denoted by $T_0,T_1,\cdots,T_9$. For every test set T_i, we denote by p_i the percentage of unspecified values in T_i. Every test set T_i detects all the detectable faults in F_{sa}.

For every test set T_i, we fill the unspecified values of T_i randomly 10 times. We denote the test sets obtained after specifying the unspecified values of T_i by $T_{i,0},T_{i,1},\cdots,T_{i,9}$. We simulate the set of bridging faults F_{br} under every test set $T_{i,j}$, for $0 \leq i \leq 9$ and $0 \leq j \leq 9$. We denote the bridging fault coverage of $T_{i,j}$ by $fc_{i,j}$.

We also perform 10-detection fault simulation of stuck-at faults under $T_{i,j}$, for $0 \leq i \leq 9$ and $0 \leq j \leq 9$. In

this process, a stuck-at fault is dropped after it is detected 10 times. We denote by $nd_{i,j}(f)$ the number of times a stuck-at fault $f \in F_{sa}$ is detected. We compute the average number of detections of stuck-at faults under $T_{i,j}$ as $\overline{nd}_{i,j} = \sum\{nd_{i,j}(f): f \in F_{sa}\}/|F_{sa}|$.

Considering $T_0, T_1, \cdots, T_9$, we find the minimum and maximum percentage of unspecified values, $p_{\min} = \min\{p_i: 0 \le i \le 9\}$ and $p_{\max} = \max\{p_i: 0 \le i \le 9\}$, respectively.

Considering the test sets $T_{0,0}$, $T_{0,1}$, $\cdots$, $T_{0,9}$, $T_{1,0}$, $\cdots$, $T_{9,9}$, we find the minimum and maximum bridging fault coverage $fc_{\min} = \min\{fc_{i,j}: 0 \le i \le 9, 0 \le j \le 9\}$ and $fc_{\max} = \max\{fc_{i,j}: 0 \le i \le 9, 0 \le j \le 9\}$, respectively. We also find the minimum and maximum average number of detections of stuck-at faults,

$$\overline{nd}_{\min} = \min\{\overline{nd}_{i,j}: 0 \le i \le 9, 0 \le j \le 9\} \text{ and}$$
$$\overline{nd}_{\max} = \max\{\overline{nd}_{i,j}: 0 \le i \le 9, 0 \le j \le 9\}.$$

Considering the original test set T, we define parameters p_{orig}, fc_{orig} and $\overline{nd}_{orig}$, which are the percentage of unspecified values, the bridging fault coverage, and the average number of detections of stuck-at faults for T. We note the following. Since T is fully-specified, $p_{orig} = 0$. Due to the selection of F_{br} as a subset of bridging faults detected by T, fc_{orig} is guaranteed to be 100%. We perform 10-detection fault simulation of stuck-at faults under T to compute the average number of detections $\overline{nd}_{orig}$.

In Table 1 we show for every circuit considered the following parameters. Under column *%unspec* we show the values of $p_{\min}$ and $p_{\max}$. Under column *bridg f.c.* we show the values of $fc_{\min}$ and $fc_{\max}$. Under column *ave nd* we show the values of $\overline{nd}_{orig}$, $\overline{nd}_{\min}$ and $\overline{nd}_{\max}$.

From Table 1 it can be seen that the 10 incompletely-specified test sets obtained for every circuit have similar percentages of unspecified values (for $s298$, between 37.50% and 39.95%). Nevertheless, there are large differences in the fault coverage of bridging faults depending on the incompletely-specified test set and the way its unspecified values are filled (for $s298$, the minimum is 71.28% and the maximum is 93.33%). The test sets obtained after filling unspecified values always detect fewer than 100% of the bridging faults detected by the original test set. This indicates that the unspecified values and their fill can have a significant effect on the untargeted fault coverage of a test set.

Considering the average number of detections of stuck-at faults, the differences between the minimum and maximum values are much less significant than indicated by the bridging fault coverage (for $s298$, the minimum is 4.60 and the maximum is 4.78). Moreover, the maximum is sometimes higher than the average number of detections of the original test set, even though the bridging fault coverage is always lower than 100%. Thus, the number of detections of stuck-at faults does not predict the coverage of untargeted faults.

Table 1: Effects of unspecifying and filling

	%unspec		bridg f.c.		ave nd		
circuit	min	max	min	max	orig	min	max
s298	37.50	39.95	71.28	93.33	4.81	4.60	4.78
s344	41.94	44.72	74.91	94.64	4.06	3.92	4.05
s382	45.83	48.33	66.44	86.24	4.73	4.42	4.85
s420	50.56	52.56	90.48	94.88	5.61	5.00	5.34
s510	67.11	67.93	82.62	90.67	5.45	5.27	5.38
s526	41.17	42.58	84.28	91.99	5.44	5.29	5.45
s641	47.73	50.84	81.09	90.94	4.97	4.91	5.13
s820	50.88	51.53	84.98	91.49	3.54	3.46	3.53
s953	71.11	71.67	93.44	96.17	5.58	5.50	5.58
s1196	57.81	58.63	91.47	96.53	6.11	6.03	6.16
s1423	42.81	46.11	82.92	91.34	5.31	5.45	5.57
s5378	71.39	72.31	73.25	77.37	7.60	7.69	7.77
s9234	67.11	67.50	86.43	88.17	5.71	5.90	5.99
s13207	92.98	93.15	72.63	75.11	6.76	7.42	7.48
s15850	79.65	80.19	80.67	83.02	6.66	7.30	7.37
s38417	73.77	73.99	90.39	91.32	7.31	7.77	7.81

5. Reducing the effects of the filling process

It is possible to reduce the effects of the filling process on the untargeted fault coverage by extending the set of faults considered during the unspecifying process, or by considering faults explicitly during the filling process. In this section we study the option of extending the set of target faults of the unspecifying process. In the context of test generation this implies using unspecified values to target additional faults during test generation. In the context of test data compression this implies ensuring that additional faults will be detected after decompression. We perform the following experiment.

We apply the unspecifying procedure using a set of target faults that consists of F_{sa} and a subset of F_{br}. The subset of F_{br} consists of a fraction s of the faults in F_{br}. The faults are selected randomly. We consider $s = 0.0$, 0.1, 0.2, $\cdots$, 1.0. We denote by F_s the set of target faults that consists of F_{sa} and s of the faults in F_{br}. We denote by T_s the test set obtained by applying the unspecifying procedure to T and F_s. After obtaining T_s, we fill its unspecified values randomly 10 times to obtain test sets $T_{s,0}, T_{s,1}, \cdots, T_{s,9}$.

For every test set $T_{s,j}$, we compute the bridging fault coverage with respect to F_{br}, denoted by $fc_{s,j}$. For every s, we find the minimum and the maximum bridging fault coverage, $fc_{s,min} = \min\{fc_{s,j}: 0 \le j \le 9\}$ and $fc_{s,max} = \max\{fc_{s,j}: 0 \le j \le 9\}$.

As s is increased, we expect $fc_{s,min}$ and $fc_{s,max}$ to become closer to each other and closer to 100%. For $s = 1.0$, we expect $fc_{1.0,min} = fc_{1.0,max} = 100\%$.

The results of this experiment are shown in Tables 2-17. For every value of s we show the percentage of unspecified values in the test set T_s. We then show the minimum and maximum bridging fault coverages $fc_{s,min}$ and $fc_{s,max}$.

From Tables 2-17 it can be seen that as s is increased, the percentage of unspecified values decreases. The minimum and maximum bridging fault coverages

increase significantly with s and approach 100%. The fault coverage reaches 100% around $s = 0.8$ or 0.9.

A significant increase in the bridging fault coverage occurs around $s = 0.1$ or 0.2 for most of the circuits considered. These values of s can be used to limit the number of target faults of the unspecifying process.

6. Concluding remarks

We considered test sets for stuck-at faults that were obtained by first unspecifying values in a given stuck-at test set T without losing stuck-at fault coverage, and then filling the unspecified values randomly. In this discussion, the unspecifying process represented a test generation or test set compression process. We obtained 100 different test sets by unspecifying T in 10 different ways, and filling the unspecified values of each test set also in 10 different ways. The results indicated that there are significant differences in the bridging fault coverage between the various test sets. The differences in the average number of detections of stuck-at faults were less noticeable. This is due to the fact that multiple tests for a stuck-at fault, which are obtained by unspecifying and filling processes, may not be sufficiently different from each other. We also showed that adding a small fraction of bridging faults to the set of faults considered during the unspecifying process improves significantly the bridging fault coverage after filling of unspecified values.

References

[1] P. Goel and B. C. Rosales, "Test Generation and Dynamic Compaction of Tests", in Proc. Test Conf., 1979 pp. 189-192.

[2] I. Pomeranz, L. N. Reddy and S. M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits", in Proc. Intl. Test Conf., 1991, pp. 194-203.

[3] J.-S. Chang and C.-S. Lin, "Test Set Compaction for Combinational Circuits", in Proc. Asian Test Symp., 1992, pp. 20-25.

[4] Y. Matsunaga, "MINT -An Exact Algorithm for Finding Minimum Test Sets", IEICE Trans. Fundamentals., vol. E76-A, No. 10, Oct. 1993, pp. 1652-1658.

[5] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits", IEEE Trans. on Computer-Aided Design, Dec. 1995, pp. 1496-1504.

[6] I. Hamazaoglu and J. H. Patel, "Test Set Compaction Algorithms for Combinational Circuits", in Proc. Intl. Conf. on Computer-Aided Design, 1998, pp. 283-289.

[7] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, B. Keller and B. Koenemann, "OPMISR: The Foundation for Compressed ATPG Vectors", in Proc. Intl. Test Conf., 2001, pp. 748-757.

[8] J. Rajski, J. Tyszer, M. Kassab, N. Kukherjee, R. Thompson, K.-H. Tsai, A. Hertwig, N. Tamarapalli, G. Mrugalski, G. Eide and J. Qian, "Embedded Deterministic Test for Low Cost Manufacturing Test", in Proc. Intl. Test Conf., 2002, pp. 301-310.

[9] A. Chandra and K. Chakrabarty, "Reduction of SOC Test Data Volume, Scan Power and Testing Time Using Alternating Run-length Codes", in Proc. Design Autom. Conf., 2002, pp 673-678.

[10] P. M. Rosinger, P. T. Gonciari, B. M. Al-Hashimi and N. Nicolici, "Analysing Trade-offs in Scan Power and Test Data Compression for Systems-on-a-Chip", IEE Proceedings - Computers and Digital Techniques, July 2002, pp. 188-196.

[11] J. Lee and N. A. Touba, "Low Power Test Data Compression Based on LFSR Reseeding", in Proc. Intl. Conf. on Computer Design, 2004, pp. 180-185.

[12] N. Badereddine, K. Chakrabarty, P. Girard, C. Landrault, S. Pravossoudovitch and A. Virazel, "Power-Aware Test Data Compression for Embedded IP Cores", in Proc. Asian Test Symp., 2006, pp. 5-10.

[13] K. Lee, J. Chen and C. Huang, "Using a Single Input to Support Multiple Scan Chains", in Proc. Intl. Conf. Computer-Aided Design, 1998, pp. 74-78.

[14] A. Chandra and M. Chakrabarty, "Test Resource Partitioning and Reduced Pin-Count Testing Based on Test Data Compression" in Proc. Design, Automation and Test in Europe Conf., 2002, pp. 598-603.

[15] S. C. Ma, P. Franco and E. J. McCluskey, "An Experimental Chip to Evaluate Test Techniques Experiment Results", in Proc. Intl. Test Conf., 1995, pp. 663-672.

[16] S. M. Reddy, I. Pomeranz and S. Kajihara, "Compact Test Sets for High Defect Coverage", IEEE Trans. on Computer-Aided Design, Aug. 1997, pp. 923-930.

[17] J. T.-Y. Chang, C.-W. Tseng, C.-M. J. Li, M. Purtell and E. J. McCluskey, "Analysis of Pattern-Dependent and Timing-Dependent Failures in an Experimental Test Chip", in Proc. Intl. Test Conf., 1998, pp. 184-193.

[18] M. R. Grimaila, S. Lee, J. Dworak, K. M. Butler, B. Stewart, H. Balachandran, B. Houchins, V. Mathur, J. Park, L.-C. Wang and M. R. Mercer, "REDO - Random Excitation and Deterministic Observation - First Commercial Experiment", in Proc. VLSI Test Symp., 1999, pp. 268-274.

[19] B. Benware, C. Schuermyer, N. Tamarapalli, K.-H. Tsai, S. Ranganathan, R. Madge, J. Rajski and P. Krishnamurthy, "Impact of multiple-detect test patterns on product quality", in Proc. Intl. Test Conf., 2003, pp. 1031-1040.

[20] S. Venkataraman, S. Sivaraj, E. Amyeen, S. Lee, A Ojha and R. Guo, "An Experimental Study of n-Detect Scan ATPG Patterns on a Processor", in Proc. VLSI Test Symp., 2004, pp. 23-28.

[21] I. Pomeranz and S. M. Reddy, "Definitions of the Numbers of Detections of Target Faults and their Effectiveness in Guiding Test Generation for High Defect Coverage", in Proc. Conf. on Design Autom. and Test in Europe, 2001, pp. 504-508.

[22] S. Kajihara and K. Miyase, "On Identifying Don't Care Inputs of Test Patterns for Combinational Circuits", in Proc. Intl. Conf. on Computer-Aided Design, 2001, pp. 364-369.

[23] A. El-Maleh and A. Al-Suwaiyan, "An Efficient Test Relaxation Technique for Combinational & Full-Scan Sequential Circuits", in Proc. VLSI Test Symp., 2002, pp. 53-59.

[24] I. Pomeranz and S. M. Reddy, "Reducing the Number of Specified Values Per Test Vector by Increasing the Test Set Size", IEE Proceedings - Computers & Digital Techniques, Jan. 2006, pp. 39-46.

[25] H. Tang, G. Chen, S. M. Reddy, C. Wang, J. Rajski and I. Pomeranz, "Defect Aware Test Patterns", in Proc. Design Autom. and Test in Europe Conf., 2005, pp. 450-455.

[26] S. Sengupta, S. Kundu, S. Chakravarty, P. Parvathala, R. Galivanche, G. Kosonocky, M. Rodgers and T. M. Mak, "Defect-Based Tests: A Key Enabler for Successful Migration to Structural Test", Intel Technology Journal, Q.1, 1999.

[27] V. Krishnaswamy, A. B. Ma, P. Vishakantaiah, "A Study of Bridging Defect Probabilities on a Pentium (TM) 4 CPU", in Proc. Intl. Test Conf., 2001, pp. 688-695.

[28] I. Pomeranz, S. M. Reddy and S. Kundu, "On the Characterization of Hard-to-Detect Bridging Faults", in Proc. Design Autom. and Test in Europe Conf., 2003, pp. 1012-1017.

Table 2: Unspecifying with target bridging faults *s* 298

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s298	0.0	38.48	83.93	93.33
s298	0.1	35.05	91.28	97.26
s298	0.2	32.11	94.53	98.29
s298	0.3	30.88	96.41	98.97
s298	0.4	29.66	97.78	99.32
s298	0.5	28.68	99.32	99.83
s298	0.6	28.43	99.32	100.00
s298	0.7	28.19	99.66	100.00
s298	0.8	28.19	99.66	100.00
s298	0.9	27.45	100.00	100.00
s298	1.0	27.45	100.00	100.00

Table 3: Unspecifying with target bridging faults *s* 344

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s344	0.0	43.33	81.83	93.77
s344	0.1	38.06	92.73	97.40
s344	0.2	36.94	95.33	97.92
s344	0.3	34.72	96.89	98.96
s344	0.4	33.61	97.40	98.96
s344	0.5	33.33	98.10	99.31
s344	0.6	31.94	98.96	99.65
s344	0.7	29.44	99.48	100.00
s344	0.8	28.89	100.00	100.00
s344	0.9	28.61	100.00	100.00
s344	1.0	28.61	100.00	100.00

Table 4: Unspecifying with target bridging faults *s* 382

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s382	0.0	47.67	66.44	76.68
s382	0.1	41.17	85.91	92.79
s382	0.2	39.67	91.28	95.30
s382	0.3	37.50	93.79	96.14
s382	0.4	36.17	96.31	98.83
s382	0.5	35.50	96.98	99.33
s382	0.6	34.50	97.65	99.50
s382	0.7	33.83	98.49	99.66
s382	0.8	33.33	98.83	100.00
s382	0.9	32.67	99.66	100.00
s382	1.0	32.17	100.00	100.00

Table 5: Unspecifying with target bridging faults *s* 420

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s420	0.0	51.23	91.19	94.76
s420	0.1	50.23	93.33	95.71
s420	0.2	49.17	94.52	96.55
s420	0.3	48.04	95.71	97.14
s420	0.4	47.38	96.43	97.86
s420	0.5	46.38	97.38	98.45
s420	0.6	45.58	97.86	99.05
s420	0.7	45.12	98.21	99.17
s420	0.8	44.19	99.05	99.88
s420	0.9	43.52	99.52	100.00
s420	1.0	43.19	100.00	100.00

Table 6: Unspecifying with target bridging faults *s* 510

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s510	0.0	67.70	85.10	89.08
s510	0.1	65.70	88.88	92.35
s510	0.2	64.30	90.96	95.13
s510	0.3	63.26	92.15	96.23
s510	0.4	62.15	93.94	97.12
s510	0.5	61.11	95.23	97.52
s510	0.6	59.85	96.92	98.31
s510	0.7	58.89	98.31	99.11
s510	0.8	57.70	99.30	99.70
s510	0.9	57.19	99.60	99.80
s510	1.0	56.59	100.00	100.00

Table 7: Unspecifying with target bridging faults *s* 526

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s526	0.0	41.25	85.64	90.33
s526	0.1	38.33	91.70	94.34
s526	0.2	36.58	95.12	96.48
s526	0.3	35.17	96.00	97.46
s526	0.4	34.25	96.78	98.34
s526	0.5	32.25	97.66	99.02
s526	0.6	31.58	98.14	99.22
s526	0.7	30.50	99.41	99.80
s526	0.8	30.08	99.61	99.80
s526	0.9	29.75	99.71	99.80
s526	1.0	29.25	100.00	100.00

Table 8: Unspecifying with target bridging faults *s* 641

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s641	0.0	50.76	85.74	89.91
s641	0.1	47.73	89.52	93.30
s641	0.2	45.12	93.70	95.43
s641	0.3	43.43	94.96	96.38
s641	0.4	41.67	96.22	96.85
s641	0.5	40.24	97.16	97.79
s641	0.6	38.72	98.66	99.13
s641	0.7	38.22	98.98	99.37
s641	0.8	37.46	99.13	99.61
s641	0.9	36.62	99.68	99.84
s641	1.0	36.03	100.00	100.00

Table 9: Unspecifying with target bridging faults *s* 820

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s820	0.0	51.20	86.59	90.59
s820	0.1	49.26	91.75	94.00
s820	0.2	47.92	93.36	96.20
s820	0.3	46.81	95.23	96.84
s820	0.4	45.33	97.36	98.26
s820	0.5	44.59	97.61	98.65
s820	0.6	43.62	98.45	99.03
s820	0.7	43.11	98.84	99.10
s820	0.8	42.37	99.29	99.48
s820	0.9	41.63	99.74	99.87
s820	1.0	41.21	100.00	100.00

Table 10: Unspecifying with target bridging faults *s* 953

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s953	0.0	71.46	94.28	96.01
s953	0.1	70.29	95.59	97.32
s953	0.2	69.53	96.59	97.95
s953	0.3	68.83	97.17	98.32
s953	0.4	68.10	98.01	98.74
s953	0.5	67.57	98.48	99.00
s953	0.6	66.96	98.79	99.32
s953	0.7	66.40	99.21	99.58
s953	0.8	65.99	99.42	99.74
s953	0.9	65.47	99.69	99.90
s953	1.0	64.97	100.00	100.00

Table 11: Unspecifying with target bridging faults *s* 1196

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s1196	0.0	58.17	92.60	94.48
s1196	0.1	57.40	95.36	96.70
s1196	0.2	56.84	96.86	97.45
s1196	0.3	56.43	97.41	98.33
s1196	0.4	56.05	97.99	98.58
s1196	0.5	55.30	98.66	99.12
s1196	0.6	54.87	98.95	99.54
s1196	0.7	54.35	99.33	99.75
s1196	0.8	54.10	99.41	99.83
s1196	0.9	53.51	99.71	100.00
s1196	1.0	53.12	100.00	100.00

Table 12: Unspecifying with target bridging faults *s* 1423

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s1423	0.0	46.11	87.43	90.18
s1423	0.1	41.34	93.24	95.42
s1423	0.2	39.01	95.04	96.65
s1423	0.3	37.66	96.37	97.32
s1423	0.4	36.26	97.78	98.31
s1423	0.5	35.33	98.10	98.80
s1423	0.6	34.62	98.49	99.05
s1423	0.7	33.31	98.98	99.51
s1423	0.8	32.50	99.33	99.65
s1423	0.9	31.61	99.68	99.86
s1423	1.0	30.77	100.00	100.00

Table 13: Unspecifying with target bridging faults *s* 5378

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s5378	0.0	71.58	73.25	74.78
s5378	0.1	64.45	86.03	86.83
s5378	0.2	60.37	90.36	91.47
s5378	0.3	57.16	93.16	93.81
s5378	0.4	54.51	94.97	95.58
s5378	0.5	52.69	96.30	96.81
s5378	0.6	50.88	97.37	97.83
s5378	0.7	49.60	98.09	98.45
s5378	0.8	47.85	98.98	99.15
s5378	0.9	46.63	99.49	99.68
s5378	1.0	45.73	100.00	100.00

Table 14: Unspecifying with target bridging faults *s* 9234

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s9234	0.0	67.47	86.90	87.51
s9234	0.1	61.02	93.14	93.76
s9234	0.2	57.51	95.08	95.46
s9234	0.3	54.71	96.41	96.68
s9234	0.4	52.41	97.21	97.55
s9234	0.5	50.44	97.95	98.15
s9234	0.6	48.54	98.45	98.60
s9234	0.7	47.18	98.87	99.01
s9234	0.8	45.72	99.30	99.39
s9234	0.9	44.43	99.70	99.76
s9234	1.0	43.26	100.00	100.00

Table 15: Unspecifying with target bridging faults *s* 13207

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s13207	0.0	93.15	73.07	74.29
s13207	0.1	89.92	84.39	85.20
s13207	0.2	87.80	88.56	89.15
s13207	0.3	86.08	91.36	91.73
s13207	0.4	84.52	93.49	93.80
s13207	0.5	83.35	94.89	95.24
s13207	0.6	82.34	96.25	96.55
s13207	0.7	81.44	97.39	97.66
s13207	0.8	80.61	98.35	98.51
s13207	0.9	79.85	99.20	99.30
s13207	1.0	79.08	100.00	100.00

Table 16: Unspecifying with target bridging faults *s* 15850

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s15850	0.0	79.91	80.93	82.24
s15850	0.1	73.85	91.30	91.86
s15850	0.2	70.35	93.43	93.79
s15850	0.3	67.77	94.72	95.04
s15850	0.4	65.51	95.81	96.11
s15850	0.5	63.44	96.96	97.15
s15850	0.6	61.72	97.72	97.86
s15850	0.7	60.21	98.36	98.51
s15850	0.8	58.93	99.00	99.08
s15850	0.9	57.64	99.55	99.61
s15850	1.0	56.58	100.00	100.00

Table 17: Unspecifying with target bridging faults *s* 38417

circuit	s	%unspec	bridg f.c. min	bridg f.c. max
s38417	0.0	73.92	90.54	90.97
s38417	0.1	69.91	94.23	94.67
s38417	0.2	67.44	95.69	96.05
s38417	0.3	65.58	96.65	96.94
s38417	0.4	63.84	97.43	97.68
s38417	0.5	62.46	98.01	98.20
s38417	0.6	61.12	98.56	98.71
s38417	0.7	59.89	98.96	99.09
s38417	0.8	58.88	99.33	99.39
s38417	0.9	57.95	99.67	99.71
s38417	1.0	57.00	100.00	100.00

2009 22nd International Conference on VLSI Design

New Techniques for Accelerating Small Delay ATPG and Generating Compact Test Sets*

Boxue Yin[1] , Dong Xiang[1] , Zhen Chen[2]
Key Laboratory for Information System Security, Ministry of Education
Tsinghua National Laboratory for Information Science and Technology
1.School of software, 2.Dept. of Com. Sci. and Tech., Tsinghua University, Beijing, China
ybx06@mails.tsinghua.edu.cn

Abstract — The small delay defects testing has two challenges. One is that the longest testable path selection for every target fault in ATPG consumes much CPU time. The other is the test data volume are very large. In this paper, we propose two strategies to resolve these two problems. A new path selection in advance scheme is proposed to accelerate ATPG. It aims to find fewer paths and cover more faults in advance, which is different from the previous works. To reduce the test data volume, we propose a novel scan-based test scheme. We partition the scan flip-flops into some scan chains. The first scan flip-flop of every scan chain works in enhanced scan mode. And other scan flip-flops work in broad-side mode. This can significantly increase the don't care bits of every test pattern and provide more room for test compaction. Then the test pattern count can be reduced significantly. Experimental results show the efficiency of these techniques.

Keywords — Small delay defect, the longest testable path selection, broad-side scan testing.

I. Introduction

As semiconductor technologies develop, the probability of small timing defects has significantly increased. It is shown that there are much more small delay defects than large delay defects in manufacturing [6]. Three types of fault models are widely used in traditional delay testing: the gate delay fault model (transition fault model) [12] [1], the path delay fault model [13] and the segment delay fault model [3]. Although the path delay model is accurate, the huge number of path delay faults makes it complex and impractical for large circuits. For transition fault model, whenever a fault occurs, it is assumed that the fault effect is large enough to cause a timing failure. Therefore, to detect a transition fault, the length of the path along which the fault is activated and propagated is not considered. The transition fault test generation often detects faults through shorter paths. Thus many small delay defects remain undetected. The segment delay fault model is a trade-off between the transition fault model and path delay fault model.

When considering the fault size, the path to activate and propagate the fault is very important. As we know, if the delay fault size is greater than the slack of the shortest functional path, S_1, the circuit will malfunction regardless of the location of the fault. This fault can be detected by using the transition fault model. If the delay fault size is smaller than the slack of the longest functional path, S_2, the circuit can work correctly and the fault is redundant. When the fault size is between S_1 and S_2, the fault will affect the normal operation of the circuit if the sum of the longest function path passing through the fault and the fault size is greater than the system clock period. This kind of faults should be activated and propagated along the longest testable path passing through them. When we use critical path in the path delay test, some small delay defects can be detected. But critical paths are limited in number, the small delay defects which can't affect the critical path need to be detected through their least slack path. When the frequency of the circuits increase, these faults will affect the function of circuits.

A. Previous works

In recent delay test studies, the detection of small delay defects has become more and more important. In [2], a new transition fault model ALAPTF is proposed, which aims to detect small delay defects. And the corresponding ATPG algorithm is developed. To generate the tests for the longest path through each gate, some techniques are used [8]. A list of long paths passing through each gate is generated before ATPG and then the test generation for the paths in this list is performed to detect the small delay faults. In [5], the transition faults with small delay are activated and propagated only along implicitly kept sensitizable critical paths for compact quality tests. Two algorithms, which are activation-first and propagation-first, are proposed in [4].

On the other hand, the ATPG for small delay defects which activate and propagate the transition faults with small delay along the longest path may generate much more patterns than the traditional transition fault generation. Even in broad-side test approach, the test set is still large. In [7], a new criterion is employed to identify a subset of small delay faults that are detected along the longest path. Others are treated as traditional transition faults. This method may reduce the overall small delay test quality.

B. Motivation

In the process of test generation for small delay defects, selecting the longest testable path for every fault gate or fault line is a time-consuming task. For some faults on the same longest testable path, the conventional ATPG tool will find the same path repeatedly. In this paper, we propose a new method that selects small set of paths which can cover

*This work was partially supported by the National Science Foundation of China under grants 60373009 and 60425203.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.64

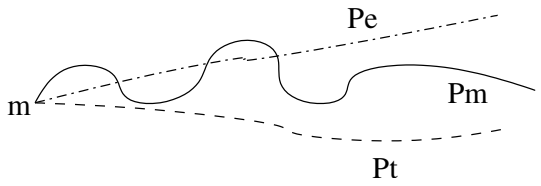


Figure 1: Example for **l-path**.

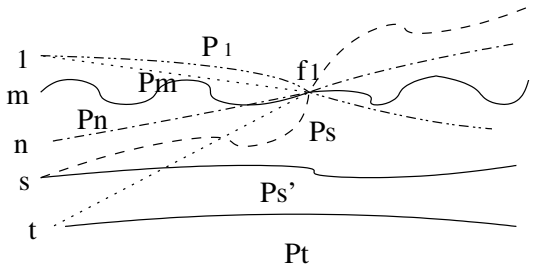


Figure 2: The proof of Lemma 1.

many faults. The testable path means functional sensitizable path. The effort to find the longest paths passing through these faults in ATPG can be avoided. We know that if a path S is the longest path passing through the fault f_1, S may also be the longest path passing through other faults $f_2...f_n$. In this condition, the conventional ATPG isn't sure whether S is the longest path for $f_2...f_n$ before these test patterns are generated. So it should find S for $f_2...f_n$ repeatedly. In this paper, we propose the new path selection scheme to resolve this problem. The longest testable path beginning from each primary input and pseudo-primary input is selected in advance. And these selected paths are recorded. Then a principle is given, and this principle can guide the ATPG to avoid finding the same longest path for some faults repeatedly. The motivation to find the longest testable path beginning from each primary input and pseudo-primary input is to find fewer paths in advance and cover more faults. This work is different from the previous work that selects the critical path. The critical path is the longest testable path in the circuit. So the number critical paths are small and the faults which they cover are also less.

The broad-side test approach is widely used for small delay defects testing. The test sets are smaller than the sets for other test approach of the same fault coverage. But its test set is still too large. We modify the standard broad-side scan test to increase don't care bits in test patterns. Then the test pattern count will be significantly reduced after test pattern compaction.

The rest of the paper is organized as follows. Section 2 presents the new path selection scheme. The process of accelerated test generation for small delay defects is described in Section 3. The modified broad-side scan test approach is proposed in Section 4. Section 5 gives the experiment results and Section 6 concludes the paper.

II. The New Path Selection Scheme

In the circuit under test, there are several paths which begin from the primary input (PI) or pseudo-primary input (PPI) and end at the primary output (PO) or pseudo-primary output(PPO). Among these paths, some can only propagate the rising transitions, while some can only propagate the falling transitions. And others can propagate both. In this paper, we find the longest path beginning from each PI and PPI that can propagate the rising transition and also find the longest path beginning from each PI and PPI that can propagate the falling transition. These two paths from the same input can be the same. In the following parts of this paper, for ease of description, we ignore the difference between these two kinds of paths. Usually, the sensitization conditions used to propagate the stuck-at faults are adopted to detect small delay defects. So the testable path that we find means functional sensitizable path. To find these longest testable paths beginning from PIs and PPIs, the redundant identification strategies proposed in [10] [16]are used. These strategies can identify the untestable paths fast. After we get the testable paths, we can easily find the longest path that begins from each PI or PPI of the circuit. Also the method in [11] can be used to obtain these longest testable paths. But in this paper, we don't find the longest path passing through each gate in circuit and only find the longest path passing through each PI and PPI. This is different from the scheme in [11].

Firstly, we find the longest testable path from each input. Then the path for each input and its length is recorded. Secondly, for each gate in the circuit, we use the testability measure proposed in Section 4 to find the PIs and PPIs which can reach the gate. Then we use these information to accelerate the ATPG program for small delay defects.

Definition 1 *The longest testable path beginning from a primary input or pseudo-primary input and ending at a primary output or pseudo-primary output is called* ***l-path****. And* ***LS*** *is the set which contains all l-paths for all primary inputs and pseudo-primary inputs in a circuit.*

As shown in Fig.1, P_e,P_m,P_t all begin from the input m and P_m is the longest one of them. So P_m is the l-path for input m.

Lemma 1 *For a fault f_1, the l-paths which pass through f_1 are P_1, ... P_n. Among these paths, the longest one is P_m. And the inputs of the circuit that can reach the fault gate are 1 ... t. The l-paths $P_1...P_t$ begin from the inputs 1 ... t. { $P_1...P_t$ } includes {P_1, ... P_n}. If P_m is also the longest one in $P_1...P_t$, then the longest testable path passing through f_1 is P_m.*

Proof: we would like to prove the lemma via contradiction. As shown in Fig. 2, suppose that there exists another testable path P_s passing through f_1 which is longer than P_m. According to the condition that P_m is the longest one of the l-paths passing through f_1, P_s is not a l-path. P_s should begin from one primary input or pseudo-primary input that is noted as s and the l-path of s is P_s'. Since P_s passes through f_1, s can reach the fault gate and the set {1 ... t}includes s. Since P_m is also the longest one of the l-paths {$P_1...P_t$} which includes P_s', P_m is longer than P_s'. And since P_s is longer than P_m, P_s is longer than P_s'. This contradicts with Definition 1 that P_s' is the l-path of s (P_s'is the longest testable path beginning from s).

The condition that P_m is also the longest one in $P_1...P_t$ is necessary for Lemma1. If P_m is not the longest one in

$\{P_1...P_t\}$, there is P_r in $\{P_1...P_t\}$ that is longer than P_m and P_r begins from input r. Then there may be a testable path beginning from r passing through the fault. This path is shorter than P_r, but is longer than P_m. Then P_m is not the longest one passing through the fault.

We would like to introduce a new test pattern generation scheme for small delay faults. According to the proposal above, we have found the l-path beginning from each PI and PPI. These l-paths compose the set LS. And the length of each l-path is recorded for each input. After this, we modify the small delay ATPG. For every target transition fault with small delay, our method will check whether it is on some l-paths. If this fault is on some l-paths in LS, the longest one (P_m) of these paths is selected. And the set SP is composed of the inputs of the circuit which can reach the fault gate. We will check whether P_m is also the longest one of the l-paths which begin from the inputs in SP. If P_m is the longest one, according to the Lemma 1, P_m is longest testable path passing through the fault gate and P_m can be selected to activate and propagate the fault. And the faults satisfying the above conditions are detected along the true longest testable path. The effort to find the longest path to propagate the fault can be avoided, because the l-paths in LS have been already found. And the process of checking whether a l-path is the longest testable path passing through a gate is very fast. This technique can significantly accelerate the ATPG. Suppose that there are n transition faults with small delay on an l-path and the l-path is the longest path passing through these faults. In the conventional ATPG process, the program has to find the l-path n times. Because though the l-path is found as the longest path for one fault, the conventional ATPG isn't sure the l-path is also the longest path for other faults. In our method, the l-path has been found in advance. During the ATPG process, the Lemma1 can guide the ATPG and the effort of finding the longest path for the n faults can be avoided.

III. The Process of Accelerated Test Generation

The process of the accelerated test generation for small delay defects is described as follows. The new path selection scheme is applied in the accelerated ATPG. And this scheme can be used in any test approach such as broad-side, skew-load and enhanced scan test approach.

Accelerated small delay test generation()
{

1. Generate the l-paths that begin from the PIs and PPIs. These l-paths compose the set LS .
2. While the fault list is not empty, do 3,4,5.
3. Select a target fault f and check if it is on any l-path in LS. If it is on some l-paths do 4. Otherwise, do 5.
4. The l-paths $P_1...P_n$ are passing through the fault f.
   (a) Find the longest one P_m of $P_1...P_n$.
   (b) Get the set of inputs $i_1...i_t$ that can reach the fault gate and get the l-paths $P_1...P_t$ beginning from $i_1...i_t$.
   (c) Check whether P_m is the longest one of $P_1...P_t$.
      i. If P_m is the longest one of $P_1...P_t$, the activation and propagation of the target fault can directly along the path of P_m. After the test pattern is generated for the target fault, do 3 select a new target fault.
      ii. If it P_m is not the longest one, do 5.
5. Traverse all the paths passing through the target fault and find the longest testable one to activate and propagate the target fault.

}

IV. Generate the Compact Test Sets

A. A novel small delay test approach

The cost of test application is directly determined by the size of test pattern set. Broad-side test approach is widely used in small delay testing. In broad-side approach, the second pattern is obtained from the response to the first pattern. So the test sets are smaller than that of other test approaches. But they are still large. Due to large test volume required to achieve satisfactory coverage, small delay defect coverage is often compromised for acceptable test volume. To reduce the test data volume, we will propose a novel scan-based test approach.

A two time-frame circuit model is widely used in broad-side scan testing for small delay defects. Fig. 3 shows the two time-frame model. Each frame of the model is a copy of the circuit under test. And the PPOs in the first frame is connected to the PPIs in the second frame. In the second time-frame, the fault effect is propagated to the PO or PPO and some PIs or PPIs should be assigned specified values. The PPOs in the first time-fame, corresponding to the PPIs which have the specified values, should backtrace to the PPIs or PIs in the first time-frame.

As shown in Fig. 3 , if PPI s_i is set to 1 to propagate the fault effect in the second frame, a large number of PPIs and PIs will be assigned specified values in the first frame. On the other hand, if the scan flip-flop s_i works in enhanced scan mode, the value 1 can be loaded from scan-in directly. And no backtrace is required in the first time-frame to assign s_i to 1. Hence, the don't care bits of the test pattern can increase. Then the space for test compaction will be much larger. And the test patterns can be reduced. By this motivation, we select some scan flip-flops working in enhanced mode. We partition the scan flip-flops into several scan chains. And the number of enhanced scan flip-flops is the same as the number of the scan chains. Then we place one enhanced scan flip-flop in each scan chain and set the enhanced scan flip-flop to be the first place in the scan chain. And these scan flip-flops at the first of the scan chains work in enhanced scan mode. The rest work in broad-side mode. Note that the number of scan chains (enhanced scan flip-flops) is limited by the number of scan-ins.

In standard scan test design, there is an input E_1 to control the scan enable signal for all the scan flip-flops. In our scan test design, an extra scan control input E_2 is needed. E_1 is used to control the scan enable signal for the broadside scan flip-flops. E_2 is used to control the scan enable signal for the enhanced scan flip-flops. When E_1=E_2=1, the circuit works in shift mode and the first test vector is shifted in. When E_1=0, E_2=1, the broadside scan flip-flops work in function mode and the enhanced scan flip-flops work in shift mode. Then the test data, which the first scan flip-flops of the scan chains need in the second time-frame, are scanned in the first scan flip-flops through the scan-ins for the scan

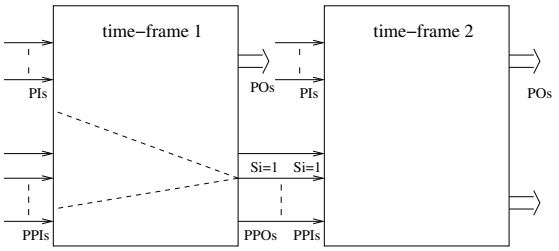


Figure 3: The two time-frame circuit model.

chains. And the broadside scan flip-flops capture the response to the first pattern simultaneously. Then E_1=E_2=0, the circuit is under function mode. Because the test data for the enhanced scan flip-flops in the second time-frame are one bit for each scan cell and the number of the enhanced scan flip-flops is small, so the test data increasing by enhanced scan test are very small. Since we only drive few scan flip-flops by E_2, the signal for E_2 can be easily designed to be fast changing between launch clock and capture clock. This technology increases the don't care bits of the test patterns. Therefore it can make the test compaction much more powerful. And it doesn't reduce the quality of the test sets .

From the description of the test application process above, the scan flip-flops which work in enhanced mode can arbitrarily receive the corresponding test data of two test vector. And each scan chain has only one enhanced scan flip-flop at the beginning of the chain. These enhanced scan flip-flops can directly receive the second test data after receiving the first test data after applying the first test data. So the enhanced scan flip-flops don't need hold-scan design. The test application process is described above.

When generating the test patterns for the modified broad-side test approach, the backtrace schemes of the values assigned to the PPIs in the second frame are different. If the PPIs working in enhanced scan mode are assigned specific values, the values are recorded directly without backtracing. And if the PPIs working in broad-side mode are assigned specific values, they should be backtraced in the first frame.

Note that this test approach is different from the one in [9]. Ours aims to provide the larger room for compaction, while [9] aims to improve the fault coverage. And the strategy of selecting the first scan flip-flops is different. Also the proposed approach is different from the one in [15]. The method of [15] combines advantages of the skew-load and broad-side approaches. The scan architectures of the proposed method and the one in [15] are different.

B. The enhanced scan flip-flops selection

As illustrated above, the first scan flip-flops of the scan chains work in enhanced scan mode that can increase the don't care bits in the test patterns. Which scan flip-flop in scan chain should be placed first to achieve the best compaction effect?

As shown in two time-frame circuit model, a PPI is assigned a specific value in the second frame. Then the corresponding PPO in the first frame should backtrace to the PPIs or PIs. If the number of these PPIs or PIs that the PPO affects is large, we choose this PPO as an enhanced scan flip-flop, then it can provide more don't care bits and it can achieve better compaction effect. So we want to choose some scan flip-flops, that their corresponding PPO in the first frame affect more PPIs or PIs, as the enhanced scan flip-flops.

A testability measure and a function are proposed in this section. The testability measure can more exactly calculate the PIs and PPIs that a PPO can affect. And the function based on the testability measure is used to choose the enhanced scan flip-flops.

The testability measure can calculate the PIs or PPIs that each gate can affect. To introduce the testability measure, we first explain some definitions. $RC_i(l)$ presents the minimum set of primary inputs (or pseudo-primary inputs) which have to be assigned a specified value(0 or 1)in order to set line l to value i. $C_i(l)$ presents the size of $RC_i(l)$. For a input l, we have:

$$RC_1(l) = RC_0(l) = \{l\}, \quad (1)$$
$$C_1(l) = C_0(l) = 1. \quad (2)$$

For an AND gate l with inputs A and B, we have:

$$RC_1(l) = RC_1(A) \cup RC_1(B), \quad (3)$$
$$C_1(l) = |RC_1(l)|. \quad (4)$$

where $|RC_1(l)|$ is the size of the set $RC_1(l)$.

$$RC_0(l) = \begin{cases} RC_0(A) & \text{if } |RC_0(A)| \le |RC_0(B)|, \\ RC_0(B) & \text{if } |RC_0(A)| > |RC_0(B)|. \end{cases} \quad (5)$$

$$C_0(l) = |RC_0(l)|. \quad (6)$$

For a OR gate l with inputs A and B, we have:

$$RC_1(l) = \begin{cases} RC_1(A) & \text{if } |RC_1(A)| \le |RC_1(B)|, \\ RC_1(B) & \text{if } |RC_1(A)| > |RC_1(B)|. \end{cases} \quad (7)$$

$$C_1(l) = |RC_1(l)|. \quad (8)$$

$$RC_0(l) = RC_0(A) \cup RC_0(B), \quad (9)$$
$$C_0(l) = |RC_0(l)|. \quad (10)$$

For a fanout s with branches B_1, B_2, ..., B_k, for $i \in \{0,1\}$, and $j \in \{1,2,\ldots,k\}$, we have:

$$C_i(B_j) = C_i(s). \quad (11)$$

Other kinds of gate are similar. After we have the testability of each gate, we propose a function to find the enhanced scan flip-flops which can significantly increase the don't care bits in generated patterns. The function is:

$$G = \sum_{ppo \in P} \sum_{i \in \{0,1\}} C_i(ppo), \quad (12)$$

where P is the set that includes all PPOs in the first frame of the two time-frame model. This function can reflect the sum of the controlability of all PPOs.

To select the enhanced scan flip-flops, we do as following. Suppose there are n scan chains. Firstly, we calculate G in the original circuit according to the function (12). Secondly, for each scan flip-flop in the circuit, we attempt to set it to be the enhanced scan cell. If a scan flip-flop is considered as the enhanced scan cell, the corresponding PPI in the second frame doesn't connect to the PPO in the first frame. This PPI is considered as a PI. Then calculate the G'. We choose the n scan flip-flops which can mostly minimize G' comparing G. That is to say we choose the n scan flip-flops working in enhanced scan mode to reduce G most. By using this method, the selected enhanced scan flip-flops can most reduce the PIs or PPIs that may be assigned specific value. So it can significantly increase the don't care bits of the generated patterns. This can increase the effectiveness of the test compaction.

V. Experimental Results

We implemented the ATPG algorithm and the proposed techniques on DELL workstation (Precision490) using C language. We develop a conventional ATPG program for small delay defects. It is based on ATALANTA. And we modify the conventional ATPG with the strategies proposed in this paper. Experimental results are shown in Table 1 and Table 2. In Table 1, the SDQL is the statistical delay quality level that is the probability of detecting small delay defects [14]. In calculating SDQL, we use the method in [14]and assume that the delay of every gate is 0.2 ns. We modify the method in [10] to find the testable paths for broad-side testing and select the longest true path passing through each gate for calculating SDQL.

Table 1 shows the results of the the proposed new path selection scheme applied to accelerate the ATPG for broad-side small delay defects testing. The selected l-paths are testable for broad-side testing and these paths are obtained by using the modified methods in [10]. Table 1 also shows the comparison between the ATPG using the proposed technique and the conventional small delay ATPG. The column "Faults" represents the fault number of the circuits. The 3rd to 5th columns show the fault coverage, CPU time and SDQL of the conventional small delay ATPG. And the 6th to 9th columns show the fault coverage, CPU time, Time ratio and the SDQL of the ATPG using the accelerating technique proposed in this paper. When we calculate the CPU time of the proposed method, the time of selecting longest testable path from each PIs and PPIs in advance is included. The "Time ratio" column shows the ratio of the CPU time of proposed method to that of conventional method. From table 1 we can see that the CPU time reduces significantly by using this accelerating technique, especially for s13207, s15850, s38417 and s38584. This shows that the proposed techniques can effectively accelerate the ATPG for small delay testing. Also the SDQL does not increase by using the proposed technique. And we know that the lower the SDQL value is, the higher the delay test quality is. The SDQLs of six circuits are smaller than that of the conventional method. Because in conventional ATPG, very few found paths for the target fault are not the longest one with the limited backtrack number. And the selected paths by using the proposed scheme are the longest ones for many faults. So the SDQL may reduce.

Table 2 shows the effectiveness of our scan test approach for test compaction. The 2nd column represents the count of the test patterns for standard broad-side small delay test approach. The 3rd to 6th columns represent the count of the test patterns for new method. And we use the proposed strategy in Section 4 to select the enhanced scan flip-flops, comparing with randomly selecting the enhanced scan flip-flops. The 3rd and 4th columns show the pattern count that the enhanced scan flip-flops are randomly selected, while the 5th and 6th show the pattern count of using the proposed selecting strategy. The "n=10","n=20" means that there are 10,20 scan flip-flops work in enhanced mode. And in the 3rd to 6th columns, the numbers on the left are the test pattern counts and the numbers on the right are the ratio of the pattern count of new method to the pattern count of standard broad-side scan test. From Table 2 we can see the test patterns are reduced significantly with the proposed approach. This approach can provide large room for test compaction. Note that the test compactor used in this experiment is simple. Also we can see the proposed testability measure and the selecting strategy are effective, comparing the results of 3rd,4th columns to those of 5th, 6th columns. And the quality of the test patterns generated by the new method does not decrease and the SDQL is the same as that of Table 1. However, the fault coverage may increase by using our test approach, we only consider the effectiveness of compaction and our selecting method is more effective for compaction.

VI. Conclusions

In this paper we propose a new technique to accelerate the small delay ATPG. The path selection scheme aims to select the fewer paths covering more faults. Also we propose a novel scan test approach. Some scan flip-flops work in enhanced scan mode, while the rest work in broad-side mode. In the two time-frame model, the enhanced scan flip-flops assigned the specific value in the second time-frame needn't backtrace to the first time-frame. This can significantly increase the don't care bits in the test patterns that provide larger room for test compaction. A selecting method is proposed. Experimental results show that these techniques are effective.

References

[1] K.-T. Cheng, "Transition Fault Testing for Sequential Circuits," IEEE Trans. Computer-Aided Design Integrate Circuits and System, vol. 12, no. 12, pp. 1971-1983, Dec. 1993.

[2] P. Gupta and M. S. Hsiao, "ALAPTF: A New Transition Fault Model and the ATPG Algorithm," in Proc. of IEEE Int. Test Conference, pp. 1053-1060, 2004.

[3] K. Heragu, J. H. Patel, and V. D. Agrawal, " Segment Delay Faults: A New Fault Model," in Proc. of 14th IEEE VLSI Test Symposium, pp. 32-39, 1996

[4] S. Kajihara, S. Morishima, A. Takuma, X. Wen, T. Maeda, S. Hamada, and Y. Sato, "A Framework of High-quality Transtition Fault ATPG for Scan Circuits," in Proc. of IEEE Int. Test Conferance, paper 2.1, 2006.

[5] M. M. V. Kumar and S. Tragoudas, "High-Quality Transition Fault ATPG for Small Delay Defects," IEEE Trans. on Computer-Aided Design of Integrated

Table 1: Comparison with the ordinary small delay testing method

Circuit	Faults	The conventional small delay test method			The proposed small delay test method			
		Coverage(%)	Time(s)	SDQL(ppm)	Coverage(%)	Time(s)	Time Ratio(%)	SDQL(ppm)
s1423	2512	87.34	2.16	0.21	87.34	1.57	72.7	0.21
s5378	6988	89.55	10.02	0.10	89.56	5.63	56.2	0.10
s9234	11328	80.12	301.25	0.95	80.12	167.05	55.5	0.92
s13207	15602	81.57	79.85	0.28	81.59	23.93	30.0	0.27
s15850	19046	78.12	119.50	1.62	78.12	33.05	27.7	1.57
s35932	62798	84.27	501.11	3.65	84.27	270.06	53.9	3.57
s38417	49738	96.24	965.60	0.17	96.34	309.31	32.0	0.16
s38584	61254	89.19	1115.93	0.47	89.19	436.73	39.1	0.46

Table 2: The effect of test compaction using the novel test approach

Circuit	Patterns of standard broad-side test approach	Patterns of proposed test approach			
		random select method		the proposed select method	
		n=10	n=20	n=10	n=20
s1423	139	95(68.3%)	78(56.1%)	80 (57.6%)	69 (49.6%)
s5378	454	286(63.0%)	223(49.1%)	210 (46.3%)	179 (39.4%)
s9234	789	466(59.1%)	401(50.8%)	346 (43.9%)	319 (40.4%)
s13207	738	501(67.9%)	493(66.8%)	410 (55.6%)	383 (51.9%)
s15850	624	357(57.2%)	283(45.4%)	270 (43.3%)	253 (40.5%)
s35932	143	129(90.2%)	114(79.7%)	122 (85.3%)	100 (69.9%)
s38417	2123	991(46.7%)	901(42.4%)	713(33.6%)	673 (31.7%)
s38584	1566	1135(72.5%)	996(63.6%)	895(57.2%)	817 (52.2%)

Circuits and Systems, vol. 26, no. 5, pp. 983-989, May 2007.

[6] R. Kapur, J. Zejda and T. W. Williams, "Fundamentals of Timing Information for Test: How Simple Can We Get ? " in Proc. of IEEE Int. Test Conference, paper 17.2, 2007.

[7] X. Lin, M. Kassab and J. Rajski, "Test Generation for Timing-Critical Transition Faults," in Proc. of 16th IEEE Asian Test Symposium, pp. 487-492, 2007.

[8] A. K. Majhi, V. D. Agrawal, J. Jacob, and L. M. Patnaik,"Line Coverage of Path Delay Faults," IEEE Trans. on Very Large Scale Integration(VLSI) Systems, vol. 8, no. 5, pp. 610-614, Oct. 2000

[9] I. Pomeranz, S. M. Reddy, "Enhanced Broadside Testing for Improved Transition Fault Coverage,"in Proc. of 16th IEEE Asian Test Symposium, pp. 473-478, 2007

[10] S. Padmanaban and S. Tragoudas, "Efficient Identification of (critical) Testable Path Delay Faults Using Decision Diagrams," IEEE Trans. on Computer-Aided Design, vol. 24, no. 1, pp. 77-87, Jan. 2005.

[11] M. Sharma and J. H. Patel, "Finding a Small Set of Longest Transition Paths That Cover Every Gate, "in Proc. of IEEE Int. Test Conference, paper 34.1, 2002.

[12] J. Savir, "Broad-side Delay Test," IEEE Trans. on Very Large Scale Integration Systems, vol. 2, no. 3, pp. 368-372, Sept. 1994.

[13] G. L. Smith, "Model for Delay faults upon paths," in Proc. of IEEE Int. Test Conferance, pp. 342-349, 1985.

[14] Y. Sato, S. Hamada, T. Maeda, A. Takatori, Y. Nozuyama and S. Kajihara, "Invisible Delay Quality - SDQM Model Lingts Up What Could Not Be Seen," in Proc. of IEEE Int. Test Conference, paper 47.1, 2005.

[15] S. Wang, X. Liu, S. T. Chakradhar, "Hybrid Delay Scan: A Low Hardware Overhead Scan-based Delay Test Technique for High Fault Coverage and Compact Test Sets," in Proc. of Design Autom. and Test in Europe Conference, pp. 1296-1301, 2004

[16] D. Xiang, K. Li, H. Fujiwara, and J. Sun, "Generating Compact Robust and Non-robust Tests for Complete Coverage of Path Delay Faults Based on Stack-at Tests," in Proc. of 24th IEEE Int. Conf. Computer Design, pp. 446-451, 2006

TIGUAN: Thread-parallel Integrated test pattern Generator Utilizing satisfiability ANalysis*

Alejandro Czutro*, Ilia Polian*, Matthew Lewis*, Piet Engelke*, Sudhakar M. Reddy**, Bernd Becker*

*Institute for Computer Science
Albert-Ludwigs-University
D-79110 Freiburg i. Br., Germany

**ECE Department
University of Iowa
Iowa City, IA 52242, USA

Abstract

We present the automatic test pattern generator TIGUAN based on a thread-parallel SAT solver. Due to a tight integration of the SAT engine into the ATPG algorithm and a carefully chosen mix of various optimization techniques, multi-million-gate industrial circuits are handled without aborts. TIGUAN supports both conventional single-stuck-at faults and sophisticated conditional multiple stuck-at faults which allows to generate patterns for non-standard fault models.

1 Introduction

Traditional deterministic automatic test pattern generation (ATPG) algorithms work directly on the circuit structure [1–4], possibly in conjunction with additional data structures such as implication graphs [5] or advanced techniques to prune the solution space [6, 7]. It has long been known that an ATPG problem can be reduced to a Boolean satisfiability (SAT) instance and solved using a SAT solver [8, 9]. However, this approach has not become widely adopted as the structural approaches tended to exhibit better performance.

It has recently been shown that SAT-based ATPG outperforms structural approaches for several classes of faults [10]. One such class consists of redundant faults. SAT solvers are routinely used to prove unsatisfiability in applications such as equivalence checking, and a number of techniques have been developed to quickly prune large parts of the solution space. In contrast, structural ATPG methods may need to traverse almost the complete solution space to make sure that no test pattern for a fault exists. It has also been reported that there are testable faults for which structural ATPG performs a large number of backtracks to find a pattern while SAT-based ATPG swiftly finds a solution [10].

The ability to handle redundant faults is becoming more important for two reasons. First, defects in nanoscale manufacturing technologies may not be described adequately by stuck-at faults [11]. Non-standard fault models such as resistive bridging faults [12, 13] or interconnect opens [14, 15] may impose very specific conditions on the lines in the circuit, which are, in many cases, impossible to satisfy, so the fault is undetectable.

*Parts of this work have been supported by the German Research Council under project BE 1176/14-1 and by the Alexander-von-Humboldt Foundation. We are thankful to Juergen Schloeffel of NXP Hamburg for providing industrial circuits and Tobias Schubert of University of Freiburg for fruitful discussions on SAT solving.

Second, redundant structures are increasingly used to enhance circuit reliability and yield [16, 17]. A significant fraction of faults in these structures are not detectable. To accurately estimate the defect coverage, the proof that the fault in question is undetectable (rather than aborted) is essential.

State of the art in SAT-based ATPG is currently given by the tool PASSAT [10] which is integrated into NXP's structural ATPG framework AMSAL. Performance enhancement of SAT-based ATPG by utilizing learning techiques has been discussed in [18].

In this paper we present the ATPG tool TIGUAN (Thread-parallel Integrated test pattern Generator Utilizing satisfiability ANalysis) which is based on the in-house SAT solver MiraXT [19]. MiraXT is a state-of-the-art SAT solver which incorporates various optimization techniques developed in the last few years. Moreover, it supports thread parallelism, thus fully utilizing the performance of multi-processor systems or multi-core processors. In contrast to PASSAT, TIGUAN is tightly coupled with the SAT engine and can dynamically control its internal parameters such as preprocessing steps to be performed or number of threads to be used. Moreover, we present a two-phase method which allows to utilize MiraXT's inherent parallelism in a meaningful way.

Another feature of TIGUAN is the support of the general conditional multiple-stuck-at (CMS@) fault model. The model allows faulty effects to be present on multiple circuit lines (victims) simultaneously if a number of conditions on other lines (aggressors) are satisfied. Many static non-standard fault models can be mapped to conditional multiple stuck-at faults, making TIGUAN a flexible tool to handle various defect classes.

Experiments demonstrate that TIGUAN can generate complete stuck-at test sets for large industrial circuits with up to several million gates without aborts. For two classes of non-standard fault models (represented by CMS@ fault lists) TIGUAN completely classifies all ISCAS and ITC benchmarks and most industrial circuits. TIGUAN also outperforms comparable SAT-based ATPG tools.

The remainder of the paper is organized as follows. The CMS@ fault model and the mapping of other fault models to the CMS@ fault model is introduced in the next section. Section 3 gives the overall flow of TIGUAN. Experimental results for stuck-at faults as well as more complex faults mapped to CMS@ faults are reported in Section 4. Section 5 concludes the paper.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.20

2 CMS@ Fault Model

TIGUAN considers the *conditional multiple-stuck-at* (CMS@) fault model which includes the standard single-stuck-at fault model and is related to generic fault modeling approaches such as fault tuples [20] or the Generalized Fault Model [21]. A CMS@ fault with r *aggressor lines* and s *victim lines* consists of a list $\{a_1/a_1^{val}, \ldots, a_r/a_r^{val}\}$ and a list $\{v_1/v_1^{val}, \ldots, v_s/v_s^{val}\}$, where each a_i and each v_j denotes a signal line and all a_i^{val} and v_j^{val} stand for a logical value (0 or 1). A circuit under a CMS@ fault exhibits faulty behavior under any input vector which sets every aggressor line a_i to a_i^{val}. In this case, the value on each victim line v_j changes to v_j^{val}.

A single-stuck-at-fault is represented by a CMS@ fault with an empty aggressor list and a victim list consisting of one entry. In the following, we explain the mapping of other fault models to CMS@ faults.

2.1 Gate-exhaustive testing

Gate-exhaustive testing requires that every single-stuck-at fault at the output of a gate is detected using all valid value combinations on the inputs of that gate [22]. A stuck-at-1 fault at the output of an AND2 gate would be tested independently by three patterns, one justifying 00 at the gate's inputs, one justifying 01 and one justifying 10. Gate-exhaustive testing was demonstrated to be effective in identifying hard-to-detect defects on actual manufactured silicon [22].

Generally, $2^n - 1$ test patterns must be generated for a stuck-at-1 fault at the output of an n-input AND or NOR gate and for a stuck-at-0 fault at the output of a NAND or OR gate. One pattern must be generated for the opposite stuck-at fault, respectively. 2^{n-1} patterns must be generated for a stuck-at-1 or a stuck-at-0 fault at the output of an XOR or XNOR gate.

Gate-exhaustive testing is easily mapped to the CMS@ fault model. For the example of the stuck-at-1 fault at the output c of an AND2 gate with inputs a and b, three CMS@ faults are injected. All three faults have the same victim list $\{c/1\}$; the aggressor lists of the three faults are $\{a/0, b/0\}$, $\{a/0, b/1\}$ and $\{a/1, b/0\}$, respectively. Similar transformations are performed for other gate types.

2.2 Resistive bridging faults

Bridging faults with non-zero bridge resistance may impact the behavior of a digital circuit in a non-trivial way [12, 13]. In general, a short defect with resistance R_{sh} between interconnect a driven by gate A and interconnect b driven by gate B imposes intermediate voltages V_a and V_b between 0 and V_{DD} on the affected interconnects. These voltages are interpreted as logic values by the gates driven by a and b, depending on the logic thresholds of the gates. The voltages V_a and V_b are determined by the electrical parameters of transistors within gates A and B and the number of transistors which are activated. The latter parameter depends on the logic values on the inputs of gates A and B. Hence, to detect a resistive short defect with a given resistance, specific values on the gates driving the shorted interconnects may be required, and the fault effect may be visible on one or more gates driven by the shorted interconnects.

The detection conditions may differ for short defects which involve the same pair of interconnects but have different resistances R_{sh}. It has been shown in [23, 24] that there exists a partition of the continuous space of R_{sh} values into $m + 1$ *sections* $[R_0, R_1]$, $[R_1, R_2]$, ..., $[R_{m-1}, R_m]$, $[R_m, \infty]$, where $R_0 := 0 < R_1 < \cdots < R_m < \infty$, such that the logical behavior of the circuit is identical for R_{sh} values within one section. This means that a test pattern generated for a short defect with a fixed resistance detects all defects between the same interconnects with a resistance from the same section. In other words, to fully cover all possible resistive short defects between interconnects a and b, it is sufficient to generate a test set which detects $m + 1$ representative defects (one from each section).

To demonstrate test generation for resistive bridging faults, we first generated resistive bridging fault lists by selecting, for each circuit, 10,000 pairs of interconnects randomly. For every pair of interconnects, we calculated the section information using the tool flow from [24] and assuming the same technology parameters as in [24]. For every section, we generated one conditional multiple-stuck-at fault. The aggressor list consisted of the conditions on the inputs of the gates driving the shorted interconnects. The victim list included all inputs of the gates driven by the shorted interconnects on which an erroneous value was interpreted.

3 TIGUAN

Given a circuit, a CMS@ fault list and a set of parameters which includes a timeout value, TIGUAN generates a test set which detects all faults for which a test pattern could be found within the time budget. All faults in the lists are classified as either detected, undetectable, or aborted (not classified within the time budget).

3.1 Test generation procedure

TIGUAN selects a fault from the fault list and attempts to generate a pattern for this fault by formulating a SAT instance in conjunctive normal form (CNF) and handing it to the MiraXT engine. The generation of the CNF is described in detail in [8, 9]. We apply the usual speed-up techniques such as D-chains [9]. The MiraXT engine incorporates several methods to accelerate SAT solving, which are described in [19] along with details on the multi-threading solving mechanism. We tuned the performance of MiraXT by adjusting several solver-internal control variables to values appropriate for ATPG instances. Based on extensive empirical data, we decided not to reuse parts of a CNF generated for one fault when considering other faults.

If MiraXT finds a model (i.e., a satisfying variable assignment) of the SAT instance, the test pattern is derived from the solution. If MiraXT reports that the instance is unsatisfiable, the fault is proven to be undetectable. TIGUAN can be started in the fault dropping mode; all yet-undetected faults in the fault list are simulated with generated patterns and covered faults are marked detected and excluded from further processing. We employ an in-house 32-bit pattern parallel fault simulator, so fault dropping is invoked after 32 new patterns have been accumulated. The ATPG process is continued until all faults have been classified.

Table 1: ATPG for stuck-at faults with fault dropping for NXP circuits, timeout 20 seconds per fault

Circuit	# gates	# faults	# detected	# redundant	# aborts	# patterns	Time per fault [s]			Total time
							CNF gen.	SAT	FSIM	T [s]
p35k	48927	67733	66721	1012	0	11536	0.033	0.0278	0.0007	1364
p45k	46075	68760	68564	196	0	3604	0.005	0.0017	0.0008	47
p77k	75033	120348	113049	7299	0	5318	0.029	0.3455	0.0510	5454
p78k	80875	163310	163310	0	0	468	0.005	0.0006	0.0061	7
p81k	96722	204174	202981	1193	0	7529	0.010	0.0017	0.0015	162
p89k (*)	92706	150538	148604	1934	0	9868	0.007	0.0015	0.0018	154
p100k	102443	162129	161404	725	0	5142	0.006	0.0032	0.0028	91
p141k (*)	185360	282428	279189	3239	0	8893	0.050	0.0337	0.0024	1706
p267k	296404	366871	365423	1448	0	11579	0.020	0.0031	0.0037	447
p269k (*)	297497	369055	367607	1448	0	11633	0.018	0.0031	0.0046	436
p286k (*)	373221	650368	640103	10264	1	20243	0.041	0.0490	0.0062	3456
p295k (*)	311901	472022	468174	3847	1	22786	0.024	0.0053	0.0042	1159
p330k	365492	540758	535070	5656	32	23392	0.038	0.0388	0.0048	3208
p378k	404367	816534	816534	0	0	1107	0.022	0.0007	0.0145	44
p388k (*)	506034	881417	876750	4665	2	11975	0.029	0.0078	0.0065	830
p469k	49771	142751	140869	1762	120	578	0.094	4.4455	1.7238	13139
p951k (*)	1147491	1557914	1542633	15281	0	20899	0.060	0.0011	0.0119	2668
p1522k (*)	1193824	1697662	1681874	15788	0	63549	0.073	0.0099	0.0173	9324
p2927k	2539052	3527607	3412613	114907	87	39842	0.156	0.0308	0.0602	33758

TIGUAN also provides a mode in which the percentage of don't cares (Xes) in the generated patterns is maximized. This property is essential for static as well as dynamic compaction [25] and test compression [26]. The injection of Xes is performed by the SAT engine; on top of that, an input-output-cone analysis similar to [27] is performed to identify further Xes. We are currently integrating more elaborate methods of test set relaxation [28, 29] into TIGUAN to achieve very high don't care densities comparable to percentages obtained by structural ATPG approaches.

3.2 Multi-threaded solving

Parallel test pattern generation requires an intelligent partitioning of the problem being solved into smaller sub-problems and distribution of these sub-problems to individual threads. This must be complemented by an appropriate representation of the data shared among the threads and an efficient mechanism to access this data. In the following, we first describe the data organization and then provide an overview of how TIGUAN partitions the test generation problem being solved.

3.2.1 Distributed data organization

MiraXT and thus TIGUAN implement parallelism based on the shared memory paradigm (rather than message passing). A common clause data base contains pointers to clauses of the original SAT instance as well as conflict clauses produced during solving. Note that clauses representing the fault-free circuit, clauses representing the circuit with the fault injected and auxilliary clauses from D-chain are treated equally. Every thread keeps a local list which contains two selected literals of each clause, called watch literals. If a thread requires the complete clause information, it must access the global data base, which may require inter-processor communication. State-of-the-art multi-processor and multi-core systems include mechanisms such as AMD's HyperTransport Bus which accelerate this kind of communication.

Utilizing a common clause database allows threads to share clauses. This concept is called knowledge sharing and basically allows the solver threads to learn from each others' mistakes (conflicts). Before inserting a clause into the common database, the thread analyzes whether clauses recently inserted by other threads are more effective, i.e., prune larger parts of the search space. An optimized fine-grained lock management is implemented. It has been shown to reduce the performance overhead due to lock conflicts to fractions of a per cent. Clause deletion is implemented by a two-stage garbage collection strategy which almost eliminates the need for locks.

3.2.2 Problem partitioning and solving

The solving and the partitioning of the instance is managed by the *master control object* (MCO) of very limited complexity. MCO essentially forwards messages between threads and does not intervene with a thread's computation process. MCO also manages running and idle threads which are waiting for new sub-problems.

After CNF generation, several preprocessing steps are performed to simplify the instance. The multithreaded solver starts by giving the complete decision tree to one of the threads, and it begins the solving process. All other threads communicate to the MCO that they are idle. Idle threads are put into sleep mode in which they do not poll and consequently do not cause communication overhead. Running threads poll the MCO periodically whether any global events have occurred.

Possible global events are 'instance has been solved by another thread', 'timeout has been exceeded', and 'idle threads exist'. In the latter case, the running thread divides its sub-problem into two parts, wakes one of the sleeping threads and transfers control of one part to this thread. If a thread's sub-problem is unsatisfiable, it inserts the required conflict clauses into the data base and enters the idle state. The problem is unsatisfiable if all threads become idle.

Table 2: ATPG without fault dropping for ISCAS, ITC and NXP circuits for stuck at faults and comparison with [18]

Circuit	Gates	Faults	TIGUAN				PASSAT	
			Det.	Red.	Ab.	T [s]	Ab.	T [s]
c0432	203	524	520	4	0	0.5	0	2.6
c0499	275	758	750	8	0	1.0	0	21.0
c1355	619	1574	1566	8	0	4.5	0	32.5
c1908	938	1879	1870	9	0	4.6	0	14.4
c3540	1741	3428	3291	137	0	14.0	0	47.9
c7552	3827	7550	7419	131	0	19.4	0	106.5
s01494	686	1506	1494	12	0	0.6	0	2.7
s05378	3221	4603	4563	40	0	4.1	0	14.3
s15850	11067	11725	11336	389	0	47.8	0	121.3
s38417	25585	31180	31015	165	0	89.7	0	191.3
b10	197	486	486	0	0	0.1	0	0.3
b11	579	1436	1434	2	0	1.0	0	4.8
b12	1127	2827	2826	1	0	1.5	0	5.6
b14	5923	16167	16137	30	0	122.1	0	1426.8
b15	8026	21282	20545	737	0	378.8	0	2673.6
p81k	96722	204174	202981	1193	0	4429	0	12116
p89k	92706	150538	148604	1934	0	2544	0	5755
p100k	102443	162129	161404	725	0	2102	19	15397
p141k	185360	282428	279189	3239	0	29938	236	95452
p951k	1147491	1557914	1542633	15281	0	158875	132	166791

3.3 Two-stage method

It has been noted, e.g. in [4], that sophisticated performance enhancements are effective for relatively few hard-to-detect faults while slowing down the processing of easy-to-detect faults. We observed that, with average SAT solving time per fault below 0.1 second for most circuits, various optimizations do not result in a net run time gain. This is also true for thread parallelism: the overhead to initialize the threads and set up the communication infrastructure does not appear to be justified for most faults.

Consequently, we implemented a two-stage ATPG strategy. In the first stage, TIGUAN is run in the single-thread mode with an aggressive time limit. In the second stage, TIGUAN is applied to the remaining hard-to-detect faults employing thread parallelism.

4 Experimental Results

TIGUAN was applied to ISCAS 85 circuits and combinational cores of ISCAS 89 circuits, ITC 99 circuits and industrial circuits provided by NXP. The measurements for stuck-at faults (Tables 1 – 4) were performed on a 2.8 GHz AMD Opteron computer with 16 GB RAM, and the measurements for non-standard fault models (Tables 5 and 6) were performed on a 2.3 GHz machine with 4 GB RAM.

4.1 Single-threaded single-stuck-at ATPG

Table 1 reports ATPG results for industrial circuits using fault dropping and 20 seconds timeout per fault (a fault was classified as aborted if no pattern was found within 20 seconds). The name of the circuit, the number of gates and collapsed faults and the distribution of the faults into classes detected, provably redundant and aborted is shown in columns 1 through 6. Column 7 contains the number of generated patterns.

Table 3: Comparison of number of aborts (Ab.) and run time for TIGUAN and PASSAT [10] with fault dropping

ITC-99 circuits					NXP circuits				
Circ.	PASSAT		TIGUAN		Circ.	PASSAT		TIGUAN	
	Ab.	T [s]	Ab.	T [s]		Ab.	T [s]	Ab.	T [s]
b14	0	19.0	0	13.2	p35k	0	1561.0	0	1364.0
b15	0	24.0	0	44.0	p81k	0	583.0	0	162.0
b17	0	142.0	0	123.6	p89k	0	573.0	0	154.0
b18	0	1350.0	0	341.8	p100k	0	410.0	0	91.0
b20	0	56.0	0	29.4	p141k	0	4740.0	0	1706.0
b21	0	59.0	0	33.3	p469k	77	6180.0	120	13139.0
b22	0	95.0	0	36.0	p951k	1	18300.0	0	2668.0

The time (in seconds) per fault for CNF generation, SAT solving and fault simulation (fault dropping) can be found in columns 8 through 10, the total time T [s] in column 11. No thread parallelism of the MiraXT engine was employed.

Circuits marked by asterisk (*) contain tristate elements. TIGUAN replaces bufif1 gates by AND gates and notif1 gates by NAND gates which retains the circuit's functionality. To prevent bus contention, an additional clause which ensures that at most one driver is active at the same time can be generated. We did not generate such a clause in our experiments.

TIGUAN can handle multi-million-gate designs with very few aborts and in limited time. The number of patterns is rather large, however we point out that no compaction techniques such as reverse-order simulation were employed. The option to maximize don't cares was not used.

Tables 2 and 3 compare the performance of TIGUAN (without thread parallelism) with the best published results by PASSAT available to us [10, 18] (only results for circuits quoted in [10, 18] are reported in Tables 2 and 3).[1] Results in Table 2 have been generated with fault dropping switched off and timeout of 20 seconds (as in [18]). We quote the best numbers achieved by PASSAT among different learning techniques presented in [18]. Table 3 compares results obtained using fault dropping and timeout of 20 seconds with columns 4 and 5 in Table VI in [10] (run times were converted into seconds). Although the same industrial circuits were used in [10, 18], some of them were named differently: circuits p44k, p49k, p80k, p88k, p99k, p177k and p1330k in [10, 18] correspond to circuits p35k, p469k, p81k, p89k, p100k, p141k and p951 in Tables 2 and 3, respectively.

TIGUAN outperforms PASSAT both with respect to aborts and run time. For circuits p89k, p141k and p951k, part of the run time advantage is due to the simplified encoding of tristate elements for the three circuits mentioned above (PASSAT switches to multi-valued logic which includes the high-impedance value if a circuit contains tristate elements). All other circuits are purely Boolean and do not require multi-valued logic.

4.2 Multithreaded performance

We ran TIGUAN in the two-stage mode described in Section 3.3. The limits for the first and the second stage were 1 and 20 seconds, respectively. Table 4 summarizes the results for circuits with at least one abort during the first stage. Column

[1] An AMD Athlon with 2.2 GHz and 1 GB RAM was used in [18]. A dual-dual-core Xeon with 3 GHz and 32 GB RAM was used in [10].

Table 4: Performance of thread-parallel two-stage approach for single-stuck-at faults

Circuit	Two-stage approach												One-stage approach (Timeout 20 s)		No timeout (no aborts)
	First stage (Timeout 1 s)		Second stage (Timeout 20 s)												
	T [s]	Faults	1 thread			2 threads			4 threads				From table 1		
		left	aborts	T [s]	tot.time	aborts	T [s]	tot.time	aborts	T [s]	tot.time		Aborts	T [s]	T [s]
p77k	4545	1322	0	2940	7485	0	1354	5899	0	1003	5548		0	**5454**	5454
p286k	2115	126	1	1459	3574	1	1232	**3347**	1	1609	3724		1	3456	3497
p295k	1062	3	1	45	**1107**	1	62	1124	1	66	1128		1	1159	1228
p330k	2376	70	31	806	3182	17	616	2992	16	491	**2867**		32	3208	23475
p388k	800	2	2	40	840	2	41	841	2	40	840		2	**830**	1263
p469k	17929	2680	141	10434	28363	28	3343	21272	3	2152	20081		120	**13139**	30815
p1522k	9295	22	0	63	9358	0	15	**9310**	0	19	9314		0	9324	9324
p2927k	25856	666	92	3929	29785	80	3298	29154	73	3120	**28976**		87	33758	50812

2 gives the run time of the first stage. The number of faults aborted during the first stage and targeted by the second stage can be found in column 3. The second stage was run for 1, 2 and 4 parallel threads with a timeout of 20 seconds. For each scenario, the number of aborts during the second stage, its run time and the cumulative run time of the first and the second stage are given in columns 4 through 12.

Columns 13 and 14 give the number of aborts and the run time of the one-stage method from columns 6 and 11 of Table 1, respectively. Note that the timeout for the one-stage method was 20 seconds. The minimal run time of columns 6, 9, 12 and 13 is marked bold. This indicates the minimal time which is required for the complete ATPG process by either two-stage or one-stage approach. The two-stage method with multithreading always yields less aborts than the one-stage approach and reduces the ATPG time for more than half of the circuits. 2-thread parallelism often yields lower run times while using 4 threads helps to reduce aborts.

For reference, the final column of Table 4 reports the time which TIGUAN consumes when started without a time limit (all faults are classified without aborts). Note that all circuits not included in Table 4 have already been classified without aborts using a timeout of one second. Hence, TIGUAN completely classified all faults in the industrial circuits (as it did for ISCAS and ITC circuits not included in Tables 1 and 4).

4.3 Non-standard fault models

Table 5 reports the application of TIGUAN to generate gate-exhaustive test sets for larger ISCAS, ITC and NXP circuits. The number of faults (column 3) significantly exceeds the number of gates (column 2). A significant fraction of the generated faults are redundant (column 5). There are little aborts (column 6). The run times exceed those for stuck-at faults but are generally reasonable (column 8).

Table 6 summarizes the performance of TIGUAN for the resistive bridging fault list generated as explained in Section 2.2. The format of the table is similar to Table 5. The number of CMS@ faults equals 10,000 multiplied by the average number m of sections per resistive bridging fault. This num-

Table 5: Results for gate-exhaustive testing with fault dropping, timeout 20 seconds per fault

Circuit	Gates	Faults	Distribution			Pats.	Run time [s]	
			Det.	Red.	Ab.		per flt.	total
c5315	2608	12084	10194	1890	0	1069	0.0004	4
c6288	2480	9664	7934	1730	0	439	0.0019	18
c7552	3827	15050	12345	2705	0	1227	0.0006	9
cs13207	9441	26004	22950	3054	0	1381	0.0006	15
cs15850	11067	29922	26703	3219	0	1213	0.0009	26
cs35932	19876	60064	46484	13580	0	128	0.0004	23
cs38417	25585	70236	66228	4008	0	2425	0.0003	20
cs38584	22447	75278	64629	10649	0	1549	0.0003	24
b17	25719	138230	97826	40404	0	6041	0.0040	554
b18	76513	396886	292165	104721	0	16084	0.0058	2313
b20	12991	66444	52049	14395	0	5048	0.0028	187
b21	13168	66420	52444	13976	0	5597	0.0029	192
b22	18789	94022	73540	20482	0	5522	0.0026	244
p330k	365492	1166046	1037130	128843	73	36401	0.0102	11934
p378k	404367	1370984	1191909	179075	0	1980	0.0037	5117
p388k	506034	1663442	1463686	199754	2	17317	0.0049	8220
p469k	49771	312784	241562	70844	378	652	0.1618	50603
p951k	1147491	3250198	2884773	365425	0	28050	0.0089	28863
p1522k	1193824	3708692	3350769	357923	0	80404	0.0140	52036
p2927k	2539052	7048378	6253392	794723	263	51340	0.0241	169859

Table 6: Results for resistive bridging faults with fault dropping, timeout 20 seconds per fault

Circuit	Faults	Distribution			Patterns	Run time [s]	
		Det.	Red.	Aborts		per fault	total
c5315	28214	19594	8620	0	1661	0.0008	23.50
c6288	33603	20086	13517	0	1320	0.0037	125.28
c7552	32028	19024	13004	0	1224	0.0013	41.94
cs13207	20366	15107	5259	0	1115	0.0007	14.42
cs15850	20061	14803	5258	0	1090	0.0014	28.18
cs35932	27160	9332	17828	0	133	0.0015	41.97
cs38417	25976	20174	5802	0	1619	0.0011	27.34
cs38584	26602	17207	9395	0	1486	0.0012	32.43
b17	41651	7966	33685	0	2925	0.0142	591.01
b18	42881	8753	34128	0	3926	0.0250	1070.18
b20	44378	8073	36305	0	2285	0.0104	461.74
b21	44915	8027	36888	0	2293	0.0104	467.61
b22	44824	8551	36273	0	2170	0.0108	482.63
p330k	23716	20991	2725	0	4428	0.0216	511.33
p378k	27898	23659	4239	0	529	0.0060	166.41
p388k	24637	21495	3142	0	2139	0.0112	274.79
p469k	45528	13444	31837	247	774	0.4523	20594.07
p951k	21967	20106	1861	0	1958	0.0149	326.58
p1522k	22731	19167	3564	0	5731	0.0522	1186.51
p2927k	22638	19351	3286	1	3761	0.0634	1434.36

ber ranges between 14,489 for b13 and 45,528 for p469k. There are again no aborts for almost all circuits while the run times are reasonable. We also applied the two-stage method, observing results similar to the case of stuck-at faults: the number of aborts was reduced, and the run time went down for circuits with the largest SAT solving time. We are not aware of comparable results by PASSAT or any other SAT-based tool.

5 Conclusions

TIGUAN currently can completely classify all single-stuck-at faults in both large industrial circuits and structurally complex ISCAS circuits without aborts. It is also an effective and flexible tool to generate tests for non-standard fault models for which no adequate dedicated ATPG tool is available. This is achieved by providing a mapping between the non-standard model and conditional multiple stuck-at fault model which TIGUAN supports. The two-stage approach allows to identify hard-to-detect faults for which sophisticated optimization strategies of the SAT engine and thread parallelism are effective.

One research direction for the future is the incorporation of state-of-the-art static and dynamic compaction [25,30–33] and test set relaxation techniques [28, 29] to reduce the pattern count. We also plan to extend the CMS@ concept to dynamic fault models such as delay faults [34], and power droop [35]. Moreover, we investigate the theoretical findings on fault vs. search parallelism [36] to better utilize novel multi-processor and multi-core architectures with ultra-fast interprocessor communication.

6 References

[1] J.P. Roth. Diagnosis of automata failures: A calculus and a method. *IBM J. Res. Dev.*, 10:278–281, 1966.

[2] P. Goel. An implicit enumeration algorithm to generate test for combinational logic. *IEEE Trans. on Comp.*, 30:215–222, 1981.

[3] H. Fujiwara. FAN: A Fanout-Oriented Test Pattern Generation Algorithm. In *IEEE International Symposium on Circuits and Systems*, pages 671–674, 1985.

[4] I. Hamzaoglu and J.H. Patel. New techniques for deterministic test pattern generation. *Jour. of Electronic Testing: Theory and Applications*, 15:63–73, 1999.

[5] P. Tafertshofer and A. Ganz. SAT based ATPG using fast justification and propagation in the implication graph. In *Int'l Conf. on CAD*, pages 139–146, 1999.

[6] E. Gizdarski and H. Fujiwara. SPIRIT: A highly robust combinational test generation algorithm. *IEEE Trans. on CAD*, 21(12):1446–1458, 12 2002.

[7] C. Wang, S.M. Reddy, I. Pomeranz, X. Lin, and J. Rajski. Conflict driven techniques for improving deterministic test pattern generation. In *Int'l Conf. on CAD*, 2002.

[8] T. Larrabee. Efficient Generation of Test Patterns Using Boolean Difference. In *Int'l. Test Conference*, pages 795–801, 1989.

[9] P. Stephan, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. Combinational test generation using satisfiability. *IEEE Transactions on CAD*, 15(9):1167–1176, September 1996.

[10] R. Drechsler, S. Eggersglüß, G. Fey, A. Glowatz, F. Hapke, J. Schloeffel, and D. Tille. On acceleration of SAT-based ATPG for industrial designs. *IEEE Trans. on CAD*, 27(7):1329–1333, 2008.

[11] R. Aitken. New defect behavior at 130nm and beyond. In *European Test Symposium (Emerging Ideas Contribution)*, pages 279–284, 2004.

[12] M. Renovell, F. Azaïs, and Y. Bertrand. Detection of defects using fault model oriented test sequences. *Jour. of Electronic Testing: Theory and Applications*, 14:13–22, 1999.

[13] P. Engelke, I. Polian, M. Renovell, and B. Becker. Simulating resistive bridging and stuck-at faults. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2181–2192, Oct. 2006.

[14] Y. Sato, I. Yamazaki, H. Yamanaka, T. Ikeda, and M. Takakura. A persistent diagnostic technique for unstable defects. In *Int'l Test Conf.*, pages 242–249, 2002.

[15] S. Hillebrecht, I. Polian, P. Engelke, B. Becker, M. Keim, and W.-T. Cheng. Extraction, simulation and test generation for interconnect open defects based on enhanced aggressor-victim model. In *Int'l Test Conf.*, 2008. In press.

[16] D.P. Siewiorek and R.S. Swarz. *Reliable Computer Systems – Design and Evaluation*. Digital Press, 1992.

[17] M. Zhang, S. Mitra, T.M. Mak, N. Seifert, N.J. Wang, Q. Shi, K.S. Kim, N.R. Shanbhag, and S.J. Patel. Sequential element design with built-in soft error resilience. *IEEE Trans. on VLSI*, 14(12):1368–1378, 2006.

[18] G. Fey, T. Warode, and R. Drechsler. Reusing learned information in SAT-based ATPG. In *VLSI Design*, pages 69–76, 2007.

[19] M. Lewis, T. Schubert, and B. Becker. Multithreaded SAT solving. In *ASPDAC 2007*, Yokohama, Japan, January 2007. 12th Asia and South Pacific Design Automation Conference.

[20] R. Desineni, K.N. Dwarkanath, and R.D. Blanton. Universal test generation using fault tuples. In *Int'l Test Conf.*, pages 812–819, 2000.

[21] S. Kundu, S.T. Zachariah, S.-Y. Chang, and C. Tirumurti. On modeling crosstalk faults. *IEEE Trans. on CAD*, 24(12):1909–1915, 2005.

[22] K.Y. Cho, S. Mitra, and E.J. McCluskey. Gate exhaustive testing. In *Int'l Test Conf.*, 2005.

[23] T. Shinogi, T. Kanbayashi, T. Yoshikawa, S. Tsuruoka, and T. Hayashi. Faulty resistance sectioning technique for resistive bridging fault ATPG systems. In *Asian Test Symp.*, pages 76–81, 2001.

[24] P. Engelke, B. Braitling, I. Polian, M. Renovell, and B. Becker. SUPERB: Simulator utilizing parallel evaluation of resistive bridges. In *Asian Test Symp.*, pages 433–438, 2007.

[25] I. Pomeranz, L.N. Reddy, and S.M. Reddy. COMPACTEST: A method to generate compact test sets for combinational circuits. In *Int'l Test Conf.*, pages 194–203, 1991.

[26] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee. Embedded deterministic test. *IEEE Trans. on CAD*, 23(5):776–792, 5 2004.

[27] S. Eggersglüß and R. Drechsler. Improving test pattern compactness in SAT-based ATPG. In *Asian Test Symp.*, pages 445–452, 2007.

[28] S. Kajihara and K. Miyase. On identifying don't care inputs of test patterns for combinational circuits. In *Int'l Conf. on CAD*, pages 364–369, 2001.

[29] A.H. El-Maleh and K. Al-Utaibi. An efficient test relaxation technique for synchronous sequential circuits. *IEEE Trans. on CAD*, 23(6):933–940, 2004.

[30] S. Kajihara, I. Pomeranz, K. Kinoshita, and S.M. Reddy. Cost-effective generation of minimal test sets for stuck-at faults in combinational logic circuits. *IEEE Trans. on CAD*, 14(12):1496–1504, 1995.

[31] B. Ayari and B. Kaminska. A new dynamic test vector compaction for automatic test pattern generation. *IEEE Trans. on CAD*, 13(3):353–358, 1994.

[32] I. Hamzaoglu and J. Patel. Test set compaction algorithms for combinational circuits. *IEEE Trans. on CAD*, 19(8):957–963, 2000.

[33] E.M. Rudnick and J. Patel. Efficient techniques for dynamic test sequence compaction. *IEEE Trans. on Computers*, 48(3):323–330, 1999.

[34] G.L. Smith. Model for Delay Faults Based upon Paths. In *Int'l Test Conf.*, pages 342–349, 1985.

[35] I. Polian, A. Czutro, S. Kundu, and B. Becker. Power droop testing. *IEEE Design & Test Magazine*, 2007.

[36] T. Fujiwara, H.; Inoue. Optimal granularity of test generation in a distributed system. *IEEE Trans. on CAD*, 9(8):885–892, 1990.

An ILP Based ATPG Technique for Multiple Aggressor Crosstalk Faults Considering the Effects of Gate Delays

Kunal Ganeshpure, Sandip Kundu
University of Massachusetts, Amherst
{kganeshp, kundu}@ecs.umass.edu

Abstract - **Crosstalk faults have emerged as a significant mechanism for circuit failure. Long signal nets are of particular concern because they tend to have a higher coupling capacitance to overall capacitance ratio. A typical long net also has multiple aggressors. In generating patterns to create maximal crosstalk noise on a net, it may not be possible to activate all aggressors logically or simultaneously. Therefore, pattern generation must focus on activating a maximal subset of aggressors switching on or about the same time the victim net switches, while propagating the fault effect to a primary output. This is a well-known problem. In this paper, we present a solution which uses 0-1 Integer Linear Programming (ILP) in conjunction with circuit transformation to model gate delays. A major contribution of this paper is modeling multi-path fault propagation as a linear programming problem. The proposed technique was applied to ISCAS 85 benchmark circuits. Results indicate that percentage of total capacitance that can be switched varies from 20-80%. Patterns generated by this technique are useful for both manufacturing test application as well as signal integrity verification.**

Keywords:-Crosstalk noise, Multiple Aggressors, Integer Linear Programming, Max-satisfiability, Delay fault, Circuit Transformation

I. INTRODUCTION

Increase in circuit density and switching speed has led to an increasing number of signal integrity related failures in VLSI circuits [1]. Capacitive crosstalk is one of the major sources of such failures. Crosstalk fault results from parasitic coupling between adjacent signal nets and is more common in nets that have weaker drivers relative to their adjacent peers. Current trends in integrated circuit design indicate that interconnect sidewall coupling capacitances can be significant, thus increasing the parasitic coupling.

Crosstalk fault effects can be classified into two types: crosstalk induced pulse and crosstalk induced delay. In the first case, the victim line remains in a static state, while one or more aggressor lines are switching. The amplitude and the width of the pulse depends, among other factors, on relative switching time of the aggressors, the amount of coupling capacitance and the relative transition times of the aggressor and victim nets. In the second case of crosstalk faults, both the aggressor(s) and victim lines have simultaneous or near simultaneous transitions.

If it were not for stringent area and performance requirements, an error due to crosstalk observed during validation could be eliminated by resizing drivers, re-routing signals, shielding interconnect lines with power distribution lines and other such redesign techniques. However, redesign may be very expensive in terms of design effort and its effectiveness may be offset by process variation. Thus, these problems need to be tested during manufacturing [2].

Crosstalk faults are observed more frequently for long nets. A long net may have multiple fanouts and may be routed through multiple levels of interconnect metals. Thus, a typical long net is capacitively coupled with multiple aggressors. Due to sharing of logic, it may not be possible to excite all aggressors while simultaneously sensitizing a victim net. Moreover, even if all the aggressor nets are excited, it may not be possible to do so in close temporal proximity to the victim net due to gate delays. From an ATPG point of view, the next best solution is to switch a set of aggressors in close temporal proximity to the victim net so as to maximize the switching of the total coupling capacitance.

In this paper, we present a novel ATPG technique to generate patterns that will excite the worst case delay at the victim by switching maximal set of aggressors and propagate the fault effect to a primary output in the presence of gate delays.

The rest of the paper is organized as follows: in section II we review previous work. Section III describes the problem statement. In section IV, the proposed Crosstalk ATPG algorithm is explained. This is followed by results for ISCAS85 benchmark circuits in section V. We conclude and propose future work in section VI.

II. PREVIOUS WORK

Crosstalk noise induced errors are a significant source of signal integrity problems in deep submicron technology. Bai, Dey and Krstic proposed a heuristic solution for multiple aggressor crosstalk ATPG problem [3]. In their approach, an implication graph is constructed to determine a feasible set of aggressors (a set of aggressors that could be switched to cause maximum crosstalk given the Boolean constraints of the circuit) and then a modified version of Path Oriented DEcision Making (PODEM) algorithm is used to determine a pattern pair that satisfies both feasible aggressor set excitation and fault propagation. Lee, Nordquist and Abraham presented an ATPG technique for crosstalk induced glitches but did not

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.10

consider crosstalk induced delay [4]. A mixed signal test generator was proposed by Chen, Gupta and Breuer in [5], which not only considered static signal values but also dynamic signals like transitions and glitches as the possible input signals. Timed test pattern generation for CMOS domino circuits has been proposed by Kundu and Blanton in [6]. Both [6] and [7] consider multiple aggressors but employ computationally expensive circuit level timing simulations. Kundu et al. proposed generalized fault model for multiple aggressor crosstalk faults in [1] but the ATPG aspect was not considered. Manich and Figueras proposed a circuit transformation approach to address gate delays for maximizing switching activity [8]. In their approach, signal activity in temporal domain of a target circuit is mapped to signal activity in the spatial domain of a zero-delay transformed circuit, thus obviating the need for explicit time domain analysis.

In [9] and [10], heuristic ATPG solution for multiple aggressor crosstalk faults considering zero delay and unit delay models respectively, have been proposed. These solutions are based on a heuristic combination of Integer Linear Programming (ILP) and stuck-at-fault ATPG. The authors do not claim the heuristic solutions to be optimal; they only claim them to be practical. In this paper, we present a unified ILP formulation that is not only optimal but also practical for the benchmark circuits and would converge to an exact solution given enough time. Moreover, we consider gate delays not only for maximal aggressor excitation but also for fault propagation thus we obtain much better quality patterns than [9] and [10].

III. PROBLEM STATEMENT

The problem of generating pattern that results in maximal crosstalk noise has two aspects:

Switching aggressors to cause maximal delay at victim: As the victim net is coupled with multiple aggressors, we have to find the subset of aggressors that create largest delay at the victim node. This is a known max-satisfiability problem [9]

Propagation of fault effect to the output: In addition to maximal noise creation, the pattern must also propagate the fault effect at the victim net to an observable output.

TABLE I INPUT PATTERNS APPLIED TO THE CIRCUIT IN FIGURE 1

Input pattern a,b,c,d,e,f	Aggressor and victim switching	Weight switched	Fault propagation	Comment
001↓↓11	$\{A_2,A_3V\}=\{\uparrow\uparrow\downarrow\}$	100+20 =120	Through gates G_5 or G_6 by setting g = 1	Greedy approach switching highly controllable nodes
↓11↓↓11	$\{A_1A_4A_3V\}=\{\uparrow\uparrow\uparrow\downarrow\}$	130	Through G_6 if it does not have large slack	Better than Greedy approach but bad for fault propagation
101↓↓11	$\{A_2A_3V\}=\{\uparrow\uparrow\downarrow\}$	120	Through G_5 or G_6	Optimal pattern

It may not always be possible to switch all the aggressors in a desired fashion and at the same time to propagate the fault effect to an output. So we seek to find the best that can be achieved. The following example illustrates the above problems.

Example: In the circuit shown in Figure 1, the gate G0 drives victim net V while the coupled aggressor lines A_1, A_2, A_3 and A_4 are driven by gates G1, G2, G3 and G4 respectively. The numbers in the box associated with the aggressors indicate the coupling weight. Coupling weight represents the magnitude of coupling capacitance between various aggressors and the victim. Total delay introduced is proportional to the sum of the coupling weights of all the aggressors switching in opposite direction of the victim (desired direction of switching).

Table I shows the effect of various approaches for maximal aggressor excitation and fault effect propagation for circuit given in Figure 1.

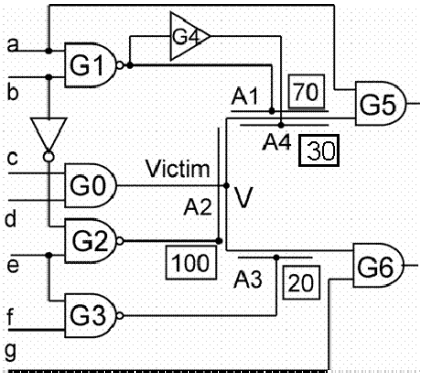


Figure 1. Example circuit showing aggressors and victims

The example above illustrates that the max-satisfiability problem of switching maximal aggressor weight is also connected to the propagation problem. The problem becomes more constrained due to the presence of gate delays as now the aggressors need to be switched in a close temporal proximity to the switching time of the victim net. This can be seen in the example below.

Example: In the circuit shown in Figure 2, the gate G5 drives victim net V while the coupled aggressor lines A_1, A_2 and A_3 are driven by gates G1, G6 and G4 respectively. The numbers in the box associated with the aggressors indicate the coupling weight if the aggressor and victim are switching in the same timeslot. The numbers below the logic gates represent the integer gate delays associated with the gates. Thus a net can only switch at integer time slots.

Consider the pattern pair $\{\uparrow,\uparrow,1,\downarrow,\uparrow\}$ at input nodes $\{a_1,a_2,a_3,a_4,a_5\}$. This will switch the victim V from 1 to 0 while the aggressors A_1 and A_2 switch in the opposite direction. Under Zero delay assumption, the total coupling weight switched is 100+30=130. But in the actual scenario, due to delay of the gates G2 and G5, the victim V switches at timeslot 6 while the aggressors A_1 and A_2 switch at timeslot 3, hence diminishing the impact of the aggressors on the victim. A pattern that generates high aggressor

switching in zero delay model, may not do as well under variable delay model.

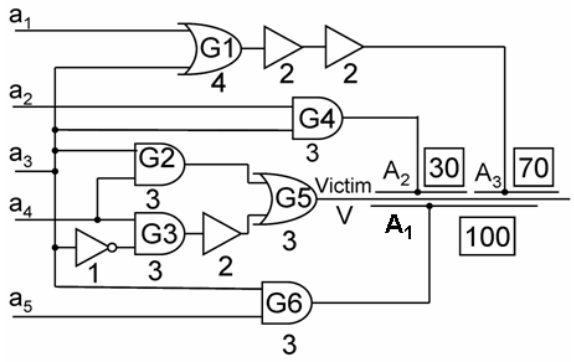


Figure 2. Aggressors and victim with gate delays for a combinational circuit

Now for pattern $\{\uparrow,\uparrow,0,\downarrow,\uparrow\} = \{a_1,a_2,a_3,a_4,a_5\}$ victim V switches from 1 to 0 and only one aggressor A_3 switches in the opposite direction. Under zero delay model, this will be considered a sub-optimal pattern as it switches a weight of 70. But the switching event at aggressor A_3 and victim V occur at the same time slot (8) and therefore has a greater impact with full coupling weight of 70.

This example shows the importance of gate delays in generating patterns for Crosstalk maximization.

IV. Proposed Solution

Max-satisfiability is known to be an intractable problem. In this paper we present a complete solution to our problem by mapping it to an ILP formulation. Thus provided enough time, we will be able to obtain input pattern that leads to absolute worst case crosstalk induced delay on the victim net.

Given a list of aggressors coupled with a victim and the corresponding coupling weight, the following steps are followed.

1: Circuit transformation

a. For gate delays

Unit delay model is incorporated by doing a time domain transformation of the circuit. Here the original circuit is transformed such that there is a one to one correspondence between transitions in the original circuit G and the XOR outputs of the transformed circuit [8] .

b. For Fault Effect Propagation

We perform circuit transformation in the output logic cone of the victim net in order to generate conditions for fault effect propagation. In this step the output logic cone including the victim is duplicated. The original logic cone represents the good machine while the duplicated logic cone represents the faulty machine. In addition, a *D* value is generated for each gate in the fault propagation cone by *XORing* the corresponding gate outputs of the two logic cones. A *D* value represents the case where the faulty value and good value are different i.e. the fault effect is propagating. ILP formulation is done to propagate the *D* value from victim net to the primary outputs.

2: ILP formulation for maximal noise generation and fault effect propagation

In this step, ILP equations are formulated to represent Boolean functions of logic gates [11], and to specify the fault propagation conditions. An objective function that maximizes the total crosstalk at the victim is generated. Thus, both the logic circuit and the maximal aggressor weight equations are represented as ILP equations. Next, we use an ILP solver to solve simultaneous logic constraints for maximal aggressor switching requirement. This represents the final test vector pair. In our solution we assume that the delay introduced by the pattern pair will be large enough to always cause an error at the output and that it does not get subsumed in timing slacks.

A. Circuit Transformation

a) Time domain expansion:

In this phase, the original circuit is expanded to represent internal switching activities due to unit gate delays [8]. For c17 benchmark circuit as shown in Figure 6, the numbers at the gate outputs represent the possible signal arrival times corresponding to the delays of all the possible paths in the input logic cone of the gate. As shown in the Figure 7, gates are replicated as many times as the number of possible arrival times in the original circuit. For example, gate number 23 is replicated three times corresponding to three arrival times in timeslots 0t, 2t and 3t. Moreover, the inputs to each of the replicas of the gate 23 are connected to the replicas of the gates 16 and 19 in previous time slot.

It should be noted that, time domain expansion can be generalized for arbitrary integer delays by adding unit delay buffers to the original circuit. Moreover, any floating point delay can be scaled and approximated as integer delays without any loss of generality of the solution.

The transition of aggressors and victim nets is indicated by XORing the corresponding outputs at various time slots. In our case Xor_V_1 is high when the victim transitions between timeslots t0 and t1.

b) Fault effect propagation:

In this phase the output logic cone of the victim net, including the victim, is duplicated for fault effect propagation. In Figure 7 the output logic cone of the victim net a11_1 (where a11 is gate number and 1 is the timeslot of the gate) is represented using broken line. The duplicated gates are renamed by replacing the prefix '*a*' with '*b*'. Inputs to the duplicated gates which are not a part of the output logic cone of the victim net are supplied from the corresponding gates in original circuit. For example, for the gate b22_3 in the duplicated circuit, the input (represented by a continuous line) which is not a part of the output logic cone of the victim comes from the gate a10_1 of the original circuit. Fault effect propagation is indicated by XORing the corresponding outputs of the original and the duplicate circuits to generate *D* value. For example, the nets a16_2 and b16_2 are XORed to obtain *D*

value of d16_2. ILP formulation is done using *D* values for fault propagation.

B. ILP formulation

a) For Logic gates:

In order to obtain the worst case crosstalk condition in the circuit, ILP formulation is done for the circuit by writing the ILP equations for the logic gates [11] which are formed using clausal description of the function of the gates [12].

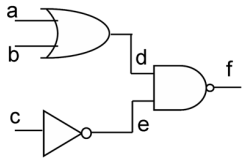


Figure 3. Combinational circuit with internal nodes names

For the circuit shown in Figure 3 the complete set of ILP equations are shown in Figure 4.

The magnitude of delay induced at the victim depends on the amount of current flowing through the coupling capacitance between the aggressor and the victim line. Figure 5 shows aggressor and victim nets coupled through a capacitance $C_{coupling}$.

$$\left.\begin{array}{l} d+f\geq 1 \\ e+f\geq 1 \\ (1-d)+(1-e)+(1-f)\geq 1 \end{array}\right\} \text{For NAND gate}$$

$$c+e=1 \quad \text{For NOT gate}$$

$$\left.\begin{array}{l} (1-a)+d\geq 1 \\ (1-b)+d\geq 1 \\ b+a+(1-d)\geq 1 \end{array}\right\} \text{For OR gate}$$

$$a,b,c,d,e,f\in[0,1]$$

Figure 4. ILP equations for circuit in the Figure 3

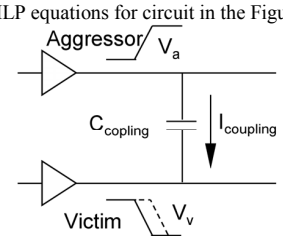


Figure 5. Capacitive coupling causing crosstalk induced delay

Here $I_{coupling}$ is the current flowing through the coupling capacitor and is dependent on the slopes and the direction of the transition at the aggressor and victim nets.

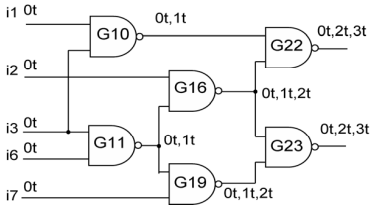


Figure 6. C17 benchmark with various switching times

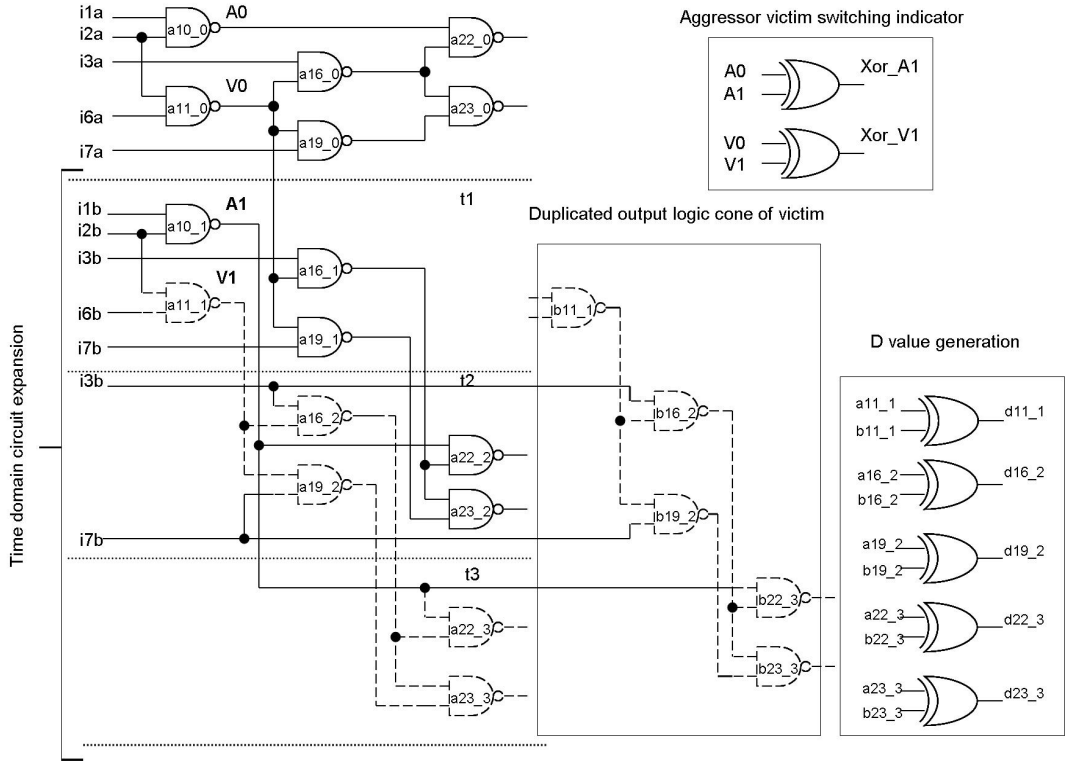


Figure 7. Circuit transformation

$$I_{coupling} = \frac{dQ}{dt} = \frac{d}{dt}(C_{coupling}(V_a - V_v)) = C_{coupling}(1-k)\frac{dV_a}{dt}$$

Where: $$k = \frac{dV_v}{dt} \Big/ \frac{dV_a}{dt}$$

TABLE II DEPENDENCE OF k ON VARIOUS SCENARIOS OF AGGRESSORS AND VICTIM TRANSITIONS

Aggressor	Victim	k
↑	↑	1
↑	↓	-1
No transition	↕	0
↕	No Transition	0

The value of k varies with the aggressor and victim transitions as shown in Table II.

Moreover, the k is scaled in a manner proportional to the time difference between the aggressor and victim transitions as shown in Table III. Thus there is crosstalk only when the aggressor and victim switch less than 3 time slot away. This defines the window size of 3.

TABLE III SCALING OF THE *K* FACTOR

Distance between transitions in Unit delay	k
0	1
1	0.66
2	0.33
3	0

The objective of the ILP formulation is to obtain a pattern pair that causes aggressor transitions so as to maximize the value of *k*.

The following section explains the ILP formulation for a victim switching at a time slot p. Here $MaxTimeSlot$ indicates the last possible time when the victim could ever switch. For example, for net $a16\_1$ in Figure 7, current timeslot $p = 1$ and $MaxTimeSlot = 2$.

b) Fault effect Propagation

Consider the pair of duplicated output logic cones of the victim net as shown in Figure 7. For a gate k in timeslot i, ak_i, bk_i and dk_i represent the gate in the original, duplicated circuit and the output of the D propagation XOR gate respectively. Thus we have

$$dk\_i = (ak\_i \oplus bk\_i)$$

To ensure fault propagation, at least one of the dk_i values at primary outputs (dk_i_{po}) must be 1.

$$\sum dk\_i_{po} \geq 1$$

Now a D value at a gate output implies that at least one of the gate inputs in the output logic cone of the victim net has a D value. Therefore for a gate k_i with inputs $k_1\_i_1$ and $k_2\_i_2$ the following implication is obtained: $dk\_i \Rightarrow dk_1\_i_1 \vee dk_2\_i_2$.

Therefore $$(1 - dk\_i) + dk_1\_i_1 + dk_2\_i_2 \geq 1$$

Finally, in order to initiate fault effect generation at the victim net, a D value has to be enforced at all the copies of the victim net v_j starting from the current time slot p to the $MaxTimeSlot$.

$$dv\_j = 1 \text{ where } j = p...MaxTimeSlot$$

Next, we explain the ILP formulation for maximal aggressor excitation.

c) Maximal aggressor excitation

Consider a pair of capacitively coupled nets having names *A* and *V*. Net *A* is the aggressor and *V* is the victim. After circuit modification, the nodes *A* and *V* are renamed to A_i and V_i, where $i = 0,1,2...MaxTimeSlot$ is the timeslot in which A and V belong in the expanded circuit. The aggressors and victim in the consecutive time-slots are XOR ed to represent a switching event between the time slots.

$$Xor\_a_i = (A_i \oplus A_{i-1}) \qquad Xor\_v_i = (V_i \oplus V_{i-1})$$

Let O_{ij} be true when both aggressor in time slot *i* and victim in timeslot *j* are switching in the opposite direction and S_{ij} be true when aggressor in timeslot *i* and the victim in timeslot *j* are switching in the same direction.

For make the aggressor in time slot *i* and victim in time slot *j*, Xor_a_i and Xor_v_j have to be true respectively. For them to switch in the opposite direction the aggressor value A_i at i^{th} timeslot and the victim value V_j at the j^{th} time slot should be opposite. Thus.

$$Xor\_a_i\_v_j = (A_i \oplus V_j)$$

Therefore $O_{ij} = Xor\_a_i \bullet Xor\_v_j \bullet Xor\_a_i\_v_j$

Similarly for the aggressor *i* and victim *j* to switch in the same direction Xor_a_i, Xor_v_j must be true and $Xor\_a_i\_v_j$ must be false.

$$S_{ij} = Xor\_a_i \bullet Xor\_v_j \bullet \overline{Xor\_a_i\_v_j}$$

For the aggressor victim pair $\{i, j\}$, $k_{ij} = (O_{ij} - S_{ij})$. For each k_{ij} there is an associated coupling capacitance C_{ij}. Thus the objective function for crosstalk noise maximization at the victim in i^{th} time-slot is given by

$$Obj = \sum_j k_{ij} * C_{ij}$$

where $j \in \{N : |i - j| < WindowSize\}$

Apart from the above constraints; it has to be made sure that the victim does not switch after current timeslot p. Thus extra set of constraints which set the victim value to a constant after switching are added as follows.

$$V_{j+1} = V_j \qquad \text{where } j \in p....MaxTimeSlot - 1$$

The above ILP formulation is solved for the victim net in all the time slots and the worst case aggressor excitation

is selected. If W_p is the solution for the victim switching in p^{th} timeslot, then the worst case switched weight is:

$$W = \max\{W_p : p = 1....MaxTimeSlot\}$$

V. Results

The crosstalk ATPG was run on all ISCAS 85 benchmark circuits to obtain the results described below. Crosstalk fault list consists of a victim net that is coupled with multiple aggressor nets. Ideally, the crosstalk fault list should be generated as a post-processing step after RC extraction from a physical layout. However, for the purpose of this paper, we created the fault list by random selection of nets as our primary goal is the ATPG process. The maximum number or aggressors per victim is limited to 7. Moreover, the coupling capacitance C_{ij} for each aggressor is also selected randomly between 1 and 0.

TABLE IV Results for ISCAS85 benchmarks circuits

Circuit name	Total Num. of Aggs. per victim	Total Agg. Coupling Cap. per victim	Aggressors switched in opposite direction from ATPG		% of total maximum aggressor weight switched	Time in (sec)
			Total Aggs. Swt.	Total Agg. Wt. Swt..		
C17	1	0.68	1	0.68	100.0	0.17
C432	6	2.61	4	0.82	31.42	4029.27
C499	7	3.23	1	0.88	27.24	3018.75
C880	7	4.64	5	3.17	68.32	4526.6
C1355	5	2.65	2	0.00	0.00	504.46
C1908	7	4.71	2	0.84	17.83	15098.61
C2670	7	2.36	4	1.45	61.44	1505.98
C3540	7	3.88	2	0.95	24.48	9123.39
C5315	7	3.61	4	2.38	65.93	2006.01
C6288	7	3.16	0	0.00	0.00	37827.36
C7552	7	4.98	6	4.52	90.76	5540.59

The ATPG results are presented in Table IV. The results are generated for a window size of 3 time units. Moreover, the ILP solver was run using a time out limit of 500 seconds. To illustrate the efficiency of the solution, Table IV shows the % of total maximum aggressor weight that is switched by the pattern obtained from our ATPG in the column 5. No improvement was obtained in the circuit's c1355 and c6288 as the solver timed-out. C6288, which is a multiplier, is a known nemesis for many SAT solvers [12]. C1355, where a parity tree is modeled by NAND gates is also known to cause problems for path oriented ATPG 0. For these problems, a heuristic solution is more appropriate than an exact solution. All results are validated using circuit simulation. It can be seen that, for most circuits the proposed solution is able to switch large fraction of the switched maximum weight as shown column 5 in Table IV.

The ILP problem was solved using GLPK, a GNU Linear Programming Kit [13]. The platform for these experiments is a Dell PowerEdge 2800 server with 2.8GHz Dual Core Intel Xeon Processor, 2MB L2 cache and 2GB RAM.

VI. Conclusion

In this paper we presented a novel ATPG technique to generate a two vector test for multiple aggressor crosstalk faults for integer delay model. The problem of maximizing the effect of the aggressor on the victim is a max-satisfiability problem which is known to be intractable. Here we solve the problem using ILP which would converge to an exact solution. We limit the ILP formulation only to the input and output cones of interest. Thus the proposed solution is both optimal and scalable. Arbitrary integer gate delays can be modeled by adding unit delay buffers, while floating delays may be scaled and approximated as integer delays without any loss of generality of the solution.

References

[1] S. T. Zachariah, Y. Chang, S. Kundu and C. Tirumurti, "On Modeling Crosstalk Faults," Design, Automation and Test in Europe, 2003, pp.10490

[2] W. Y. Chen, S. K. Gupta and M. A. Breuer, "Test generation in VLSI circuits for crosstalk noise," Proceedings of International Test Conference, 1998, pp. 641–650

[3] X. Bai, S. Dey and A. Krstic, "HyAC: A Hybrid Structural SAT Based ATPG for Crosstalk," International Test Conference, 2003, pp. 112-121

[4] K. T. Lee, C. Nordquist and J. A. Abraham, "Automatic test pattern generation for crosstalk glitches in digital circuits," Proceedings of VLSI Test Symposium, 1998, pp. 34-39

[5] W. Y. Chen, S. K. Gupta and M. A. Breuer, "Test generation for crosstalk-induced delay in integrated circuits," Proceedings of International Test Conference, 1999, pp. 191-200

[6] R. Kundu and R. D. Blanton, "Timed Test Generation Crosstalk Switch Failures in Domino CMOS Circuits," VLSI Test Symposium, 2002, pp. 379-388

[7] B. C. Paul and K. Roy, "Testing Crosstalk Induced Delay Faults in Static CMOS Circuits Through Dynamic Timing Analysis," International Test Conference, 2002, pp. 384-390

[8] S. Manich and J. Figueras, "Maximizing the Weighted Switching Activity in Combinational CMOS Circuits under the Variable Delay Model," In proceedings of European Design and Test Conference, 1997, pp. 597-602

[9] K. Ganeshpure, S. Kundu, "Automatic Test Pattern Generation for Maximal Circuit Noise in Multiple Aggressor Crosstalk Faults," Design Automation and Test in Europe, 2007

[10] K. Ganeshpure, S. Kundu, "On ATPG for Multiple Aggressor Crosstalk Faults in Presence of Gate Delays," ITC 2007

[11] R. Fortet, "Applications de l'algebre de Boole en recherche operationelle," Revue Francaise de Recherche Operationelle, 1960,Vol. 4, pp. 17-26

[12] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," IEEE Transactions on Computer-Aided Design, 1992, Vol. 11, No. 1, pp. 4-15

[13] http://www.gnu.org/software/glpk/

Goel, P., "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," Computers, IEEE Transactions on , vol.C-30, no.3, pp.215-222, March 1981

Session 4A

Advanced Device Modeling

Concept of "Crossover point" and its application on Threshold Voltage definition for Undoped-Body Transistors

Ratul Kumar Baruah[1] and Santanu Mahapatra[2]
Nano-Scale Device Research Laboratory, Centre for Electronics Design and Technology
Indian Institute of Science, Bangalore - 560012
[1]rkbarua@gmail.com & [2]santanu@cedt.iisc.ernet.in

Abstract

As the conventional MOSFET's scaling is approaching the limit imposed by short channel effects, Double Gate (DG) MOS transistors are appearing as the most feasible candidate in terms of technology in sub-45nm technology nodes. As the short channel effect in DG transistor is controlled by the device geometry, undoped or lightly doped body is used to sustain the channel. There exits a disparity in threshold voltage calculation criteria of undoped-body symmetric double gate transistors which uses two definitions, one is potential based and the another is charge based definition. In this paper, a novel concept of "crossover point"is introduced, which proves that the charge-based definition is more accurate than the potential based definition. The change in threshold voltage with body thickness variation for a fixed channel length is anomalous as predicted by potential based definition while it is monotonous for charge based definition. The threshold voltage is then extracted from drain currant versus gate voltage characteristics using linear extrapolation and "Third Derivative of Drain-Source Current" method or simply "TD" method. The trend of threshold voltage variation is found same in both the cases which support charge-based definition.

1 Introduction

In a continuous effort to increase current drive and better control short-channel effects, MOS transistors have evolved from classical bulk single-gate devices into multi-gate devices. As the dimensions of transistors are shrinking, the close proximity between the source and drain reduces the ability of the gate electrode to control the potential distribution and the flow of current in the channel region, and undesirable effects, called "short channel effects" starts plaguing MOSFETs. For all practical perposes, it seems impossible to scale the dimensions of classical "bulk" MOSFETs below 20nm. In a bulk device (fig. 1(a)), the electric field lines propagate through the depletion regions associated with the junctions. Their influence on the channel can be reduced by increasing the doping concentration in the channel region. In very small devices, unfortunately, the doping concentration becomes too high $(10^{19}cm^{-3})$ for proper device operation.

One solution to recover this problem is use of a fully depleted single gate SOI (FDSOI) device, where most of the field lines propagate through the burried oxide (BOX)before reaching the channel region (fig. 1(b)). SOI devices have better current driving capability and they consume less power than bulk devices. Short channel effects can be reduced in FDSOI devices by using a thin burried oxide and an underlying ground plan (silicon substrate). Most of the electric field lines from the source and drain terminate on the burried ground plan instead of the channel region. This approach, however, has increased junction capacitance and body effect.

A much more efficient device configuration is obtained by using the double-gate transistor structure (fig. 1(c)). In a double-gate device the electric field lines from source and drain underneath the device terminate on the bottom gate electrode and canot, therefore, reach the channel region. Only the field lines that propagate through the silicon film itself can encroach on the channel region and degrade short channel characteristics. This encroachment can be reduced by reducing the silicon film thickness.

Therefore, as the conventional single gate bulk MOSFET (Metal Oxide Scale Effect transistor) scaling is approaching the limit imposed by short channel effects, Double Gate(DG) MOSFET is becoming attractive candidate for future VLSI due to its better gate control over the channel [1]. In DG MOSFET the short channel effect is controlled by the device geometry and hence undoped (or, lightly doped) body is used to sustain the channel. Undoped body also helps to alleviate several other problems related to nano-scale CMOS e.g., mobility degradation, random dopant fluctuations, compatibility with mid-gap metal

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.41

gate etc. There is however a sharp distinction between the electrostatics of traditional bulk transistors and undoped body devices. In bulk transistor, where the substrate is sufficiently doped, the inversion charges are located close to the surface and hence the surface potential solely controls the electrostatic integrity of the device. However, in undoped body devices, gate electric field penetrates the body center, and inversion charge exists throughout the body. Therefore the definition of threshold voltage needs to be reconsidered for undoped body devices.

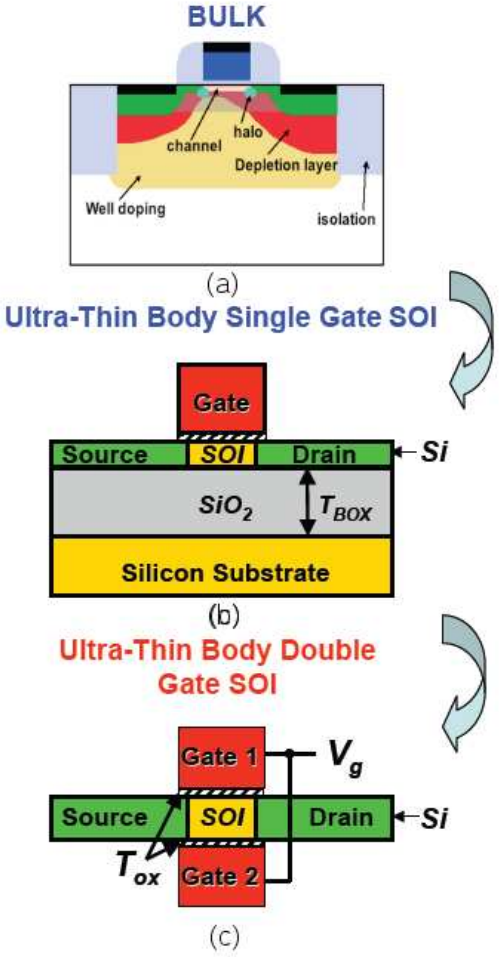


Figure 1. Evolution of Double Gate MOSFET.

Till now two definitions for threshold voltage calculation in short-channel undoped DG MOSFETs (Fig. 2) have been proposed. The first one is surface potential based [2], which is similar to the definition of the threshold voltage of bulk devices. The second one is based on the amount of charge

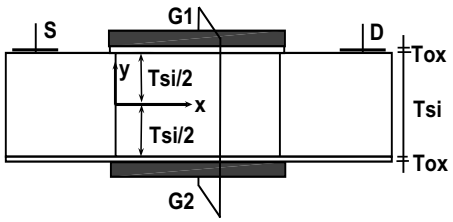


Figure 2. Cross-section of DG MOSFET adopted in device simulations.

per unit area of the body [3, 4]. It is argued that the charge based definition is more accurate than the surface potential based definition as in undoped body devices charge exits throughout the body [5]. However no proof is found behind this claim. In this paper we introduced the concept of "crossover point" to solve this dilema. We demontrate that the body potential versus gate voltage characteristics for DG MOSFETs having equal channel lengths but different body thickness pass through a single common point, which we term a "crossover point". Using the concept of "crossover point" it is shown that in case of surface potential based definition the threshold voltage changes anomalously with body thickness variation, where in case of charge based defination the threshold voltage increases monotonously with decreasing body thickness. It is also found that the threshold voltage actually increases monotonously with decreasing body thickness if it is extracted from $I_D - V_G$ characteristics using different methodologies (Linear extrapolation method, TD Method). We therefore justify that the charge based definition is more appropriate than surface potential for threshold voltage calculation of undoped body multi-gate transistors.

2 Results and Discussion

2.1 Calculation of Threshold Voltage

So far two definitions for threshold voltage (V_{TH}) calculation are used for short-channel undoped body multi-gate transistors: (i) surface potential based definition [2], which is similar to the threshold voltage definition of bulk devices

and (ii) charge based definition [3, 4], which is based on the amount of charge per unit area of the body. As in undoped body transistors charge exists throughout the body, the second definition is argued to be more accurate, where the threshold voltage is defined as the gate voltage at which the charge per unit area (Q) at the virtual cathode becomes equal to some critical threshold charge (Q_{TH}), and Q is defined as

$$Q = qn_i \int_0^{T_{si}/2} e^{\Psi(X_c,y)/U_T} dy \tag{1}$$

Here, q is the electronic charge, n_i is the intrinsic carrier concentration of body, $\Psi(x,y)$ is the body potential, T_{si} is the body thickness, L is the channel length, U_T is the thermal voltage, x and y are the directions parallel and perpendicular to the Si/SiO_2 interface and ($L/2$,0) denotes body center, X_c is the position of virtual cathode $\left(\left.\frac{d\Psi}{dx}\right|_{x=X_c} = 0\right)$, which is approximately equal to $L/2$ for low V_{ds}. Now, as $\Psi(X_c,y)$ is a very complicated function of y, the above integration cannot be evaluated analytically. So in common practice the integration is approximated as:

$$Q \approx qn_i \frac{T_{si}}{2} e^{\Psi(X_c,T_{si}/4)/U_T} \tag{2}$$

Therefore, the threshold criteria $Q = Q_{TH}$ can be expressed in terms of potential as

$$\Psi(X_c, T_{si}/4)_{TH} = U_T \ln\left(\frac{2Q_{TH}}{qn_i T_{si}}\right) \tag{3}$$

In surface potential based definition the threshold voltage is defined as the gate voltage when the surface potential Ψ_s at virtual cathode becomes equal to some constant critical value Ψ_{crit}. Taur [2] has taken the value of Ψ_{crit} as $E_g/2$, where E_g is silicon bandgap. Since V_{DD} for future technology nodes will take values less than 1 V [6], $\Psi_{crit} = E_g/2$ appears to be impractical as it results in threshold voltage of the order of 0.8 V. $\Psi_{crit} = E_g/2$ condition actually denotes the onset of strong inversion. As future devices will be operating in moderate inversion region, $\Psi_{crit} = E_g/2 - 4U_T = 0.45$ V appears to be more practical definition for threshold voltage.

2.2 Crossover Point and Pseudo Flatband condition

From numerical device simulation [7] it is observed that if the body potential of a undoped DG MOSFET is plotted against gate voltage at $y = 0$ (body center), $y = T_{si}/2$ (body surface), $T_{si}/4$ (Mid point of surface and body center) all characteristics pass through a common point for a particular gate voltage (V_{FB}) (Fig. 3). This implies that

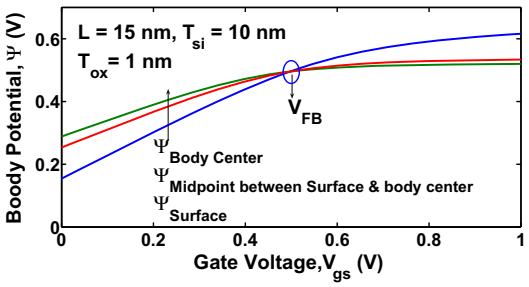


Figure 3. Potential vs. $V_{gs}@L/2$ curve for L=15nm, $T_{si} = 10nm$ and $T_{ox} = 1nm$ at body surface, center and midpoint between them showing V_{FB}.

at V_{FB} there is no potential drop along the radial direction from body center to the surface. This is precisely the flat band condition.

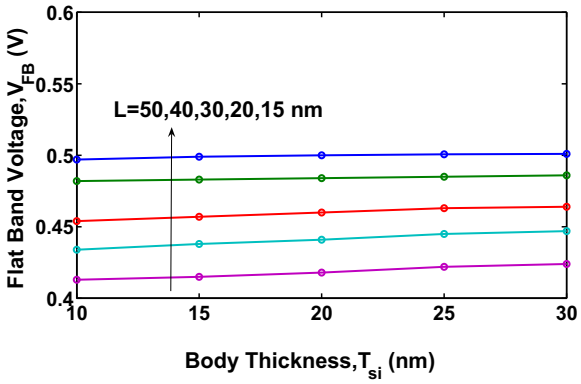
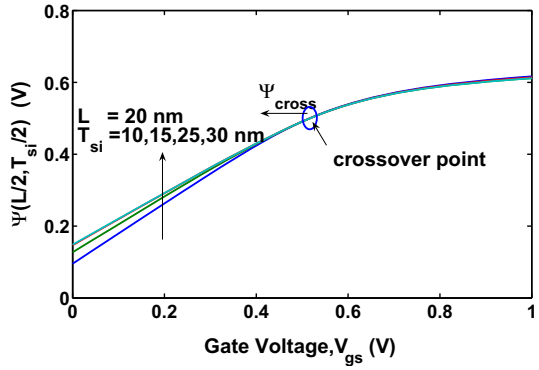


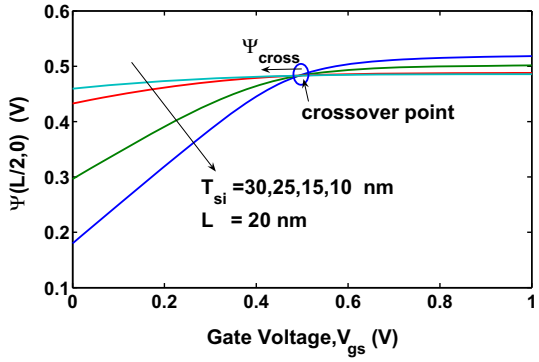
Figure 4. Curve showing constant V_{FB} at different body thickness for different L.

But we attribute this situation as "Pseudo Flat Band condition" as there exists potential variation along the lateral direction. Interestingly, it is also observed that V_{FB} is *almost* independent of the variation of body thickness when channel length is constant (Fig. 4), or in other words, devices having same L but different T_{si} hold same value of V_{FB}.

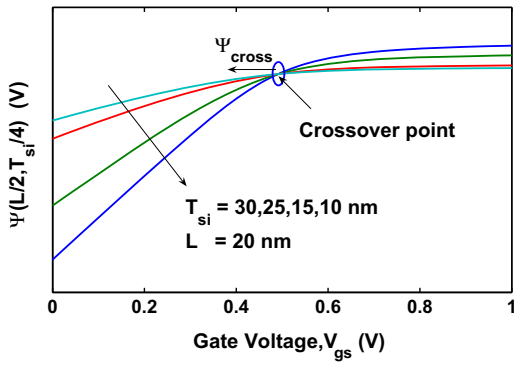
Hence, for a given L and different T_{si}'s, if we plot potential as a function of V_{gs} at a particular radial point $(X_C, T_{si}.m)$, where $m(< \frac{1}{2})$ is a constant, all the characteristics should pass through the common "flatband" point,



(a)



(b)



(c)

Figure 5. Body potential versus gate voltage characteristics for channel length L=20 nm at body surface $(L/2, T_{si}/2)$, center $(L/2, 0)$ and midpoint betwen them $(L/2, T_{si}/4)$.

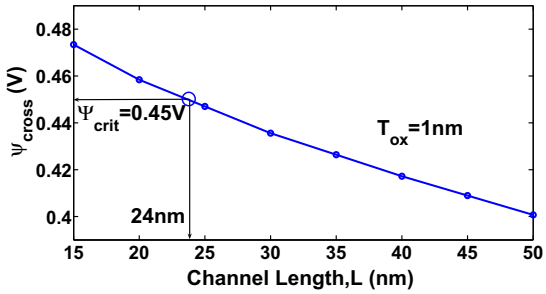


Figure 6. Variation of "crossover point" potential (Ψ_{cross}) as a function of channel length.

which we termed as "crossover point" (Fig. 5(a), 5(b) and 5(c)).

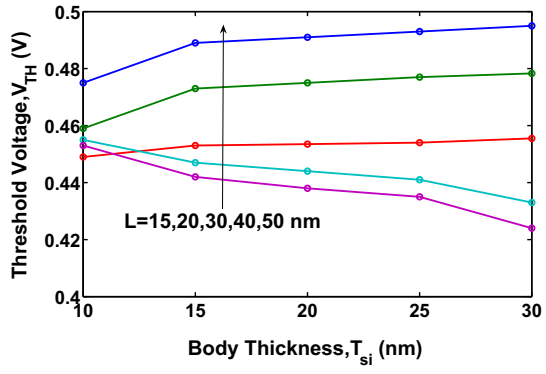
The body potential related to this point is denoted by Ψ_{cross}. Another observation made in this work is, the value of Ψ_{cross} increases with decreasing channel length as shown in Fig. 6. This is due to the fact that for long channel devices surface potential is always greater than body centre for positive gate voltages. Hence for long channel devices $V_{FB} \approx 0$ (or, more precisely equal to the difference between gate and body work function). However, for short channel devices, due to the lateral electric field from drain-to-source body center potential could be higher than surface (Fig. 3) for positive V_{GS}. Hence, in order to bring the surface potential equal to body center one needs higher gate voltage. As a result Ψ_{cross} increases with decreasing L. This phenomena is used to justify the definition of threshold voltage as discussed in the next section.

2.3 Effect of Body Thickness on Threshold Voltage

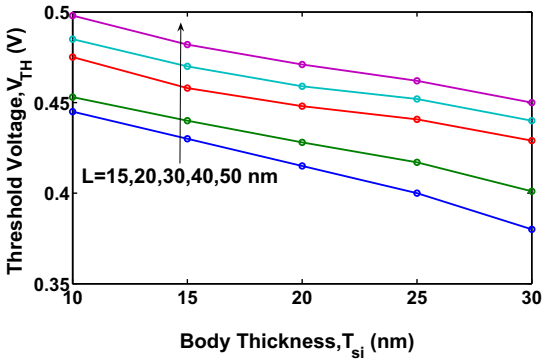
The relative value between Ψ_{crit} and Ψ_{cross} dictates how the V_{TH} will change with T_{si} for a given channel length if one uses surface potential based definition. If $\Psi_{crit} > \Psi_{cross}$, the threshold voltage will decrease with T_{si}. However, the opposite trend is observed for devices having $\Psi_{crit} < \Psi_{cross}$. As Ψ_{cross} increases with decreasing L (Fig. 6), it is expected that for small channel lengths $(< 24nm)$, V_{TH} should increase with T_{si} and for large L $(> 24nm)$ it should exhibit the opposite trend.

Therefore, surface potential based definition results in anomalous change in threshold voltage for body thickness variation for any given channel length as shown in Fig.7(a).

Using Charged based definition, the threshold voltage

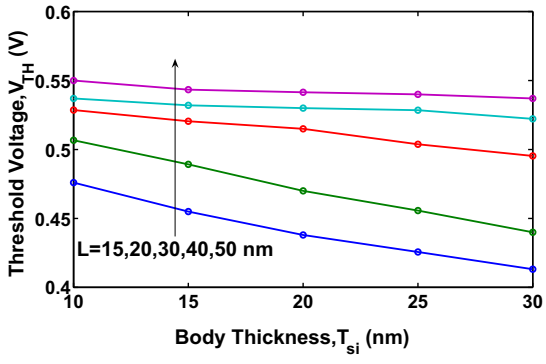


(a)

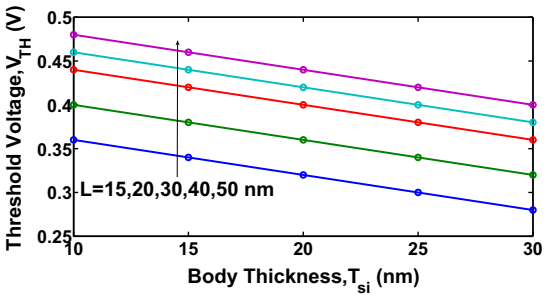


(b)

Figure 7. (a) V_{TH} calculated from surface potential based definition (b) V_{TH} calculated from charge based definition.



(a)



(b)

Figure 8. V_{TH} versus T_{si} characteristics as extracted from (a) Linear extrapolation method (b) TD method.

variation with body thickness is found to follow monotonus trend for any given channel length (Fig. 7(b)). In this work we have used potential at $(X_c, T_{si}/4)$ and $Q_{TH} = 8 \times 10^{-4} C/m^2$ to compute threshold voltage from Eqn (3). The trend of V_{TH} vs T_{si} characteristics remain unchanged if other values of Q_{TH} are used.

In order to justify which definition is correct, we have extracted threshold voltage from $I_D - V_G$ characteristics using two different methods. First, we use Linear Extrapolation method. Here, V_{TH} is extracted by linearly extrapolating the I_{ds} versus V_{gs} characteristics at low drain voltage from the point of maximum g_m (= dI_d/dV_{gs}). Second, we use TD method, where V_{TH} is extracted the point of maximum $\partial^2 g_m/\partial^2 V_{gs}$. The extracted V_{TH} are plotted as a function of body thickness in Fig. 8(a), 8(b).

For both the cases monotonous trend is observed which is similar to charged based definition (Fig. 7(b)). Therefore, we conclude that for undoped multi-gate devices charge based model for threshold voltage calculation is more accurate than surface potential model.

It is worth nothing that similar argument is equally valid for Gate-All-Around (GAA) Cylindrical transistors, as "crossover points" are also observed in those devices [5].

3 Conclusion

In this work by using a novel concept called "crossover point", it is demonstrated the change in threshold voltage with body thickness variation for a fixed channel length is anomalous as predicted by potential based definition while it is monotonous for charge based definition. The trend of threshold voltage variation extracted from drain current versus gate voltage characteristics using linear extrapolation and TD methods support the charge-based definition. Therefore, it is concluded that charge based definition is more accurate than potential based definition.

4 Acknowledgement

This work is funded by Department of Science and Technology (DST), India, under grant number: SR/FTP/ETA-05/2006.

References

[1] J.T. Park and J. P. Colinge, "Multiple gate SOI MOSFETs: Device design guidelines", *IEEE Trans. Electron Devices*, vol. 49, no. 12, pp. 2222-2229, Dec. 2002.

[2] Y. Taur, "Analytic solutions of charge and capacitance in symmetric and asymmetric double-gate MOSFETs", *IEEE Trans. Electron Devices*, vol. 48, no. 12, pp. 2861-2869, Dec. 2001.

[3] Hamdy Abd El Hamid, Benjamin Iniguez, and Jaume Roig Guitart, "Analytical Model of the Threshold Voltage and Subthreshold Swing of Undoped Cylindrical Gate-All-Around-Based MOSFETs" *IEEE Trans. Electron Devices*, vol. 54, no. 3, pp. 572-579, March 2007.

[4] Q.Chen,Â E.M. Harrell, II, and J.D. Meindl,"A physical short-channel threshold voltage model for undoped symmetric double-gate MOSFETs" *IEEE Trans. Electron Devices*, vol. 50, no. 7, pp. 1631-1637, July, 2003.

[5] Biswajit Ray and Santanu Mahapatra, "Modeling and Analysis of Body Potential of Cylindrical Gate-All-Around Nanowire Transistor", *IEEE Trans. Electron Devices*,Vol 55, No. 9, 2008.

[6] The International Technology Roadmap for Semiconductors, Semiconductor Industry Association, San Jose, CA, 2006.

[7] ATLAS User's Manual ,Version 5.10.R,Dec 2005, SILVACO INTERNATIONAL.

[8] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*, Cambrigde, U.K.: Cambridge Univ. Press, 1998.

2009 22nd International Conference on VLSI Design

Extended-Sakurai-Newton MOSFET Model for Ultra-Deep-Submicrometer CMOS Digital Design

Nishant Chandra, Apoorva Kumar Yati, A.B. Bhattacharyya
Jaypee Institute Of Information Technology University, Noida-201307, India
chandra.nishant2005@gmail.com, apoorva.yati@gmail.com, abbhattacharyya@yahoo.com

***Abstract*—In this paper an extension of the Sakurai and Newton's N^{th} power law model, namely Extended-Sakurai-Newton Model, is proposed. The proposed model (henceforth referred to as the ESN model) preserves the simplicity and accuracy of the Sakurai Newton model for the estimation of drain current in deep submicron CMOS devices and extends it for varying device widths. Although the Modified Sakurai-Newton Current Model (MSN Model) also provides an estimation of transistor drain current with varying transistor widths, it suffers from the drawback of being more error prone and computation intensive in parameter extraction. The proposed model matches with BSIM3v3 level 49 T-SPICE simulations to within an error of 1.8%(3.67% maximum), in 0.18μm and 0.25μm CMOS processes for a wide range of transistor widths and input rise/fall times. The proposed model is further used to improve the Elmore Delay prediction of CMOS inverter operated at low supply voltages. The centroid-of-current and power based delay metrics [1] are modified based on the proposed model. The new delay metric is able to accurately predict the delay of CMOS inverter operated at low supply voltages. The proposed ESN Model is also applied to predict the delay of two-input CMOS NAND gate. Hence the proposed model can be effectively used in the design of digital CMOS gates involving varying device widths and supply voltages in the deep submicron region.**

I. Introduction

In CMOS digital design, the drain current, having a decisive impact on the circuit speed, needs to be modeled in a compact manner so that the digital building blocks can be designed and evaluated analytically on a given technology node. VLSI designers require quick and accurate design methods for estimating the performance of CMOS circuits. These estimations are then followed by SPICE simulations for arriving at the final design. The Shockley Model for MOSFETs [2] is widely used for manual calculations of drain current for MOS transistors due to compactness of the equations involved. However, this model hardly reflects the impact of technological scaling on transistor drain current [3].

Sakurai and Newton [4] proposed an empirical compact MOSFET model known as Alpha-Power Law Model for the estimation of transistor drain current which takes care of scaling related effects and enables quick estimate of CMOS digital circuit performance. Such an approach is particularly necessary for the purpose of manual design and pedagogy. This model takes the velocity saturation effect into account through a parameter α, which has a value between 1 and 2 depending upon the channel length for a given technology under consideration. Since, the α-power law model was not sufficiently accurate, an improved model known as Nth power law model [3] (henceforth referred to as the SN model) was proposed later by Sakurai and Newton. Although the SN model is widely used for CMOS gate design, yet it fails to predict the drain current of narrow-width transistors for varying widths, which are usually encountered in such designs. A Modified Sakurai-Newton (MSN) model [5] has been proposed recently which attempts to alleviate the problem related to accuracy of the model with variable width by introducing a separate width-dependent process transconductance parameter in the drain current expression of the SN model but, it is found that the extraction of coefficients related to width-dependence parameter is computation intensive and prone to error.

The present communication proposes an Extended-Sakurai-Newton Compact MOSFET model (henceforth referred as the ESN Model), which improves the accuracy of the Sakurai-Newton model through an improved procedure for extraction of width dependent coefficients. We have observed that the constant process transconductance parameter 'B' in the SN model can itself be modified for varying transistor widths and hence eliminates the need to derive a separate transconductance parameter 'K' as in [5]. Since the method of parameter extraction of the ESN model is same as that of the SN model [3] and is done by solving single variable equation, the chances of error in computation of model parameters are minimized. The proposed model matches with the BSIM3v3 level 49 T-SPICE simulations to within an average error of 1.8%(3.67% maximum) for 0.18μm and 0.25μm CMOS processes over a wide range of transistor widths and input rise/fall times. Thus, the Sakurai-Newton model can be extended to predict the drain current for varying transistor widths.

It has been shown that the Elmore delay of CMOS gates, defined as the centroid of the drain current [1] during the switching process, is the Sakurai-Newton Delay ($t_{pHL,pLH}$) [4] for a step input. The performance of the ESN Model is evaluated by benchmarking it against SPICE simulation, particularly at low supply voltages (upto $V_{DD} = 2V_{TH}$) through the use of the centroid of current based delay metric presented in [1]. The delay metric failed to accurately predict the delay for varying supply voltage. The same centroid of current based delay metric, when modified by the ESN Model is able to accurately track the SPICE delay for supply voltage upto $3V_{TH}$ with a high correlation coefficient of 0.98.

It is worth mentioning that the Sakurai-Newton Model [3] is an engineering MOSFET model which is capable of providing a "working formula" for design of nanoscale CMOS gates

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.48

and has been shown to model new emerging devices such as DG MOSFETs and SiGe devices as well [7]. Though it is apparently empirical, the parameters have been shown to have physical basis [8] and finds support from extensive technology scaling experiments as well [9].

The paper is organized as follows. Section II presents the Sakurai Newton model [3]. In Section III we present the Extended Sakurai Newton (ESN) Model proposed by us. Application of the ESN Model to inverter delay, CMOS gate design and proposed modification to the centroid of current based delay metric for predicting Elmore delay for low supply voltage is given in Section IV, followed by conclusion in Section V.

II. Sakurai-Newton Model

The following are the drain current expressions given by the SN model [3]:

$$V_{TH} = V_{TO} + \gamma(\sqrt{2\phi_F - V_{BS}} - \sqrt{2\phi_F}) \quad (1)$$

$$V_{DSAT} = K(V_{GS} - V_{TH})^m \quad (2)$$

$$I_{DSAT} = \frac{W}{L_{EFF}} B(V_{GS} - V_{TH})^\alpha \quad (3)$$

$$I_D = I_{DSAT}(1 + \lambda V_{DS}) \quad \lambda = (\lambda_0 - \lambda_1 V_{BS}) \quad (4)$$

$$(V_{DS} \geq V_{DSAT} : saturated\, region)$$

$$I_D = I_{DSAT}(1 + \lambda V_{DS}) \left(2 - \frac{V_{DS}}{V_{DSAT}}\right) \frac{V_{DS}}{V_{DSAT}} \quad (5)$$

$$(V_{DS} > V_{DSAT} : linear\, region)$$

The parameter 'α' is the velocity saturation index where $1 \leq \alpha \leq 2$. B is the transconductance parameter, V_{GS}, V_{DS}, and V_{BS} are gate-source, drain-source and bulk-source voltage, respectively. V_{TH} is the threshold voltage, V_{DSAT} is the drain saturation voltage, L_{eff} is the effective channel length, V_{DD} is the supply voltage and I_{DSAT} is the drain saturation current. V_{T0}, γ and $2\phi_F$ describe the threshold voltage. K and m control the linear region characteristics while B and n determine the saturation region characteristics. λ_0, λ_1 and λ are related to finite drain conductance in the saturated region. The parameters appearing in the equations are derived using the I-V characteristics of a MOSFET of a given width [3].

Limitations of the SN Model: According to (4) and (5), the drain current given by the SN model is directly proportional to the transistor gate width as 'B' is constant. The SN model is inaccurate when compared with BSIM3v3 level 49 T-SPICE simulations as the transistor width increases from the minimum size. The SN model parameters were derived [3] for 0.18μm and 0.25μm process technologies for transistor widths of 0.22μm and 0.36μm respectively and used to predict the drain current for larger transistor widths. Fig. 1 is a representative illustration which demonstrates that the SN model becomes inaccurate as the transistor width increases

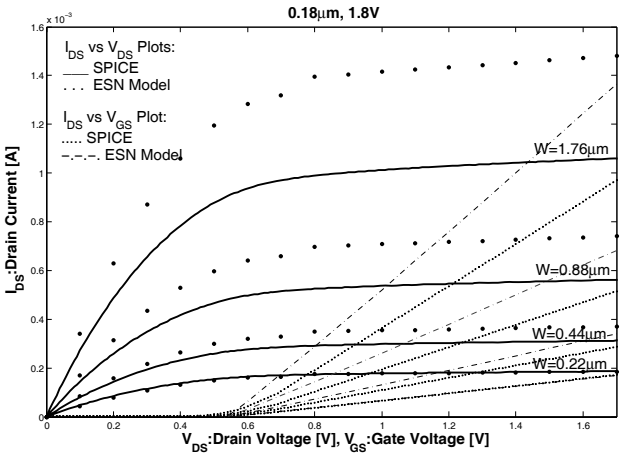


Fig. 1. SN Model I-V plots for different NMOS transistor widths(W); for $L_{eff} = 0.18\mu m$.

from the minimum value. This inaccuracy is due to the fact that the process transconductance parameter 'B' does not take into account the narrow-width effects. In order to get accurate drain current estimates for larger widths one needs to re-derive the model parameters for each transistor width, which is inconvenient and impractical. Thus there is a need for a model that takes into account the narrow-width effects and also preserves the simplicity and accuracy of the SN model.

III. The Extended Sakurai-Newton Model

We propose the following modification to the SN model. The drain current equations are as follows:

$$I_{DSAT} = B(V_{GS} - V_{TH})^\alpha \quad (6)$$

$$V_{DSAT} = K(V_{GS} - V_{TH})^m \quad (7)$$

$$I_D = I_{DSAT}(1 + \lambda V_{DS}) \quad \lambda = (\lambda_0 - \lambda_1 V_{BS}) \quad (8)$$

$$(V_{DS} \geq V_{DSAT} : saturated\, region)$$

$$I_D = I_{DSAT}(1 + \lambda V_{DS}) \left(2 - \frac{V_{DS}}{V_{DSAT}}\right) \frac{V_{DS}}{V_{DSAT}} \quad (9)$$

$$(V_{DS} > V_{DSAT} : linear\, region)$$

where, the parameter B is defined as:

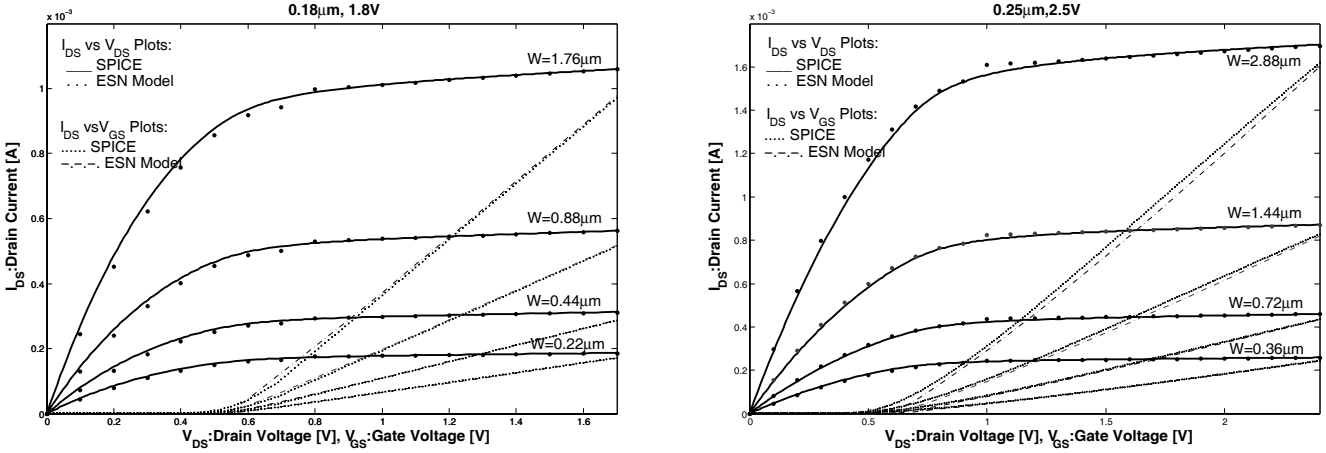
$$B = \beta_1 + \beta_2 W + \beta_3 W^2 \quad (10)$$

As is evident from (10), the transconductance parameter 'B' is a width-dependent quantity. It is also to be noted that the parameter 'B' is same as the transconductance parameter described in (3). Hence, its method of extraction is the same as described in [3] for any given value of transistor width. All the other model parameters are same as those of the SN model and hence are extracted in the same manner as given in [3].

For a given technology node, the values of all the SN model parameters are derived for the minimum transistor width as described in [3]. In order to extend the SN model to accurately predict transistor drain current values for higher widths, only the value of the parameter 'B' has to be re-derived for every

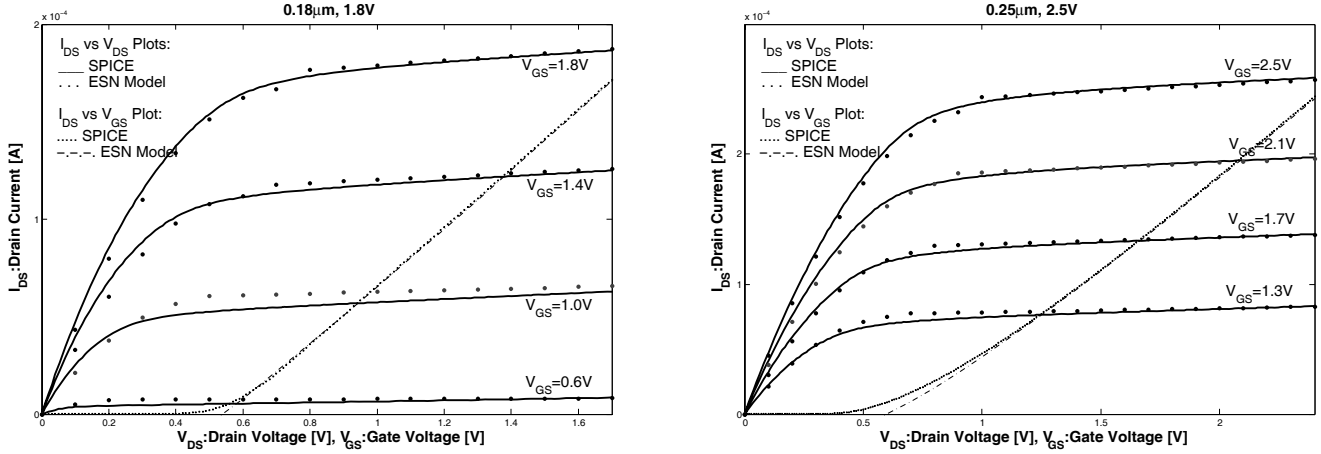
TABLE I
ESN MODEL PARAMETER VALUES

	0.25μm CMOS Process		0.18μm CMOS Process	
Parameters	NMOS	PMOS	NMOS	PMOS
α	1.1436	1.403	1.0448	1.6181
K	0.7509	1.112	0.66	0.7176
m	0.4182	0.7499	0.3785	1.4235
$\lambda(V^{-1})$	0.0421	0.0612	0.0731	0.051
$V_{T0}(V)$	0.5935	-0.5785	0.5421	-0.4373
$\gamma(V^{1/2})$	0.438	0.496	0.457	0.602
$2\phi_F(V)$	0.866	0.879	0.882	0.872
$V_{DSAT}(V)$	$0.7509(V_{GS}-V_{TH})^{0.4182}$	$1.112(V_{GS}-V_{TH})^{0.7499}$	$0.66(V_{GS}-V_{TH})^{0.3785}$	$0.7176(V_{GS}-V_{TH})^{1.4235}$
β_1	2.384×10^{-5}	6.304×10^{-6}	4.228×10^{-5}	1.116×10^{-5}
β_2	244.1	85.65	402.1	140.1
β_3	1.393×10^{6}	1.543×10^{5}	-2.091×10^{6}	2.167×10^{6}



(a) (b)

Fig. 2. ESN-model I-V plots for varying W; (a) 0.18μm (b) 0.25μm



(a) (b)

Fig. 3. ESN-model I-V plots for varying V_{GS}; (a) 0.18μm (b) 0.25μm

value of transistor width, keeping all the other parameters constant. The coefficients β_1, β_2 and β_3 are then determined by fitting a quadratic to the B vs. W plot as the transistor drain current is no longer linearly related to the width as the

TABLE II
T_{PHL} RESULTS FOR DIFFERENT INVERTER CIRCUIT CONFIGURATIONS

Technology	W_p [μm]	W_n [μm]	T_{IN} [ps]	C_L [fF]	T-SPICE [ps]	ESN Model [ps]	SN Model [ps]	Error ESN [%]	Error SN [%]
0.25	1.44	0.36	50	50	294.78	297.74	295.47	1.0	0.23
				100	541.65	539.85	535.68	0.33	1.1
			100	50	301.41	304.95	302.68	1.17	0.42
				100	548.4	547.06	542.89	0.24	1.0
		1.44	50	50	97.96	96.03	81.68	1.97	16.62
				100	171.27	167.65	141.73	2.11	17.25
			100	50	104.68	103.24	88.89	1.38	15.08
				100	177.92	174.87	148.94	1.71	16.29
	1.44	2.88	50	50	57.36	55.68	46.55	2.93	18.84
				100	94.89	91.96	76.57	3.08	19.31
			100	50	64.11	62.5	53.76	2.5	16.14
				100	101.56	99.18	83.78	2.34	17.51
		4.32	50	50	43.38	42.16	35.24	2.81	18.76
				100	68.59	66.68	55.25	2.78	19.45
			100	50	50.15	48.8	42.45	2.69	15.35
				100	75.26	73.3	62.47	2.6	17
0.18	0.88	0.22	45	45	251.25	253.97	256.53	1.08	2.10
				75	399.6	398.05	402.1	0.39	0.63
			75	45	255.82	258.72	261.28	1.13	2.13
				75	404.23	402.8	406.8	0.35	0.64
		0.88	45	45	92.16	91.33	70.68	0.9	23.31
				75	141.33	139.54	107.08	1.27	24.23
			75	45	97.03	96.08	75.43	0.98	22.26
				75	146.21	144.3	111.83	1.31	23.51
	0.88	1.76	45	45	55.19	54.16	40.48	1.87	26.65
				75	81.27	79.81	58.68	1.79	27.8
			75	45	60.16	58.9	45.23	2.09	24.82
				75	86.23	84.56	63.42	1.94	26.45
		2.64	45	45	41.85	40.44	30.17	3.37	27.91
				75	59.6	57.97	42.3	2.73	29.03
			75	45	46.9	45.18	34.92	3.67	25.54
				75	64.6	62.72	47.05	2.91	27.17
							Average Error	1.84	16.4

transistor width decreases.

Fig. 2 and 3 show the comparison between the ESN model I-V curves and BSIMv3v3 level 49 T-SPICE simulations for 0.25μ and 0.18μ CMOS processes. As shown in the plots, the proposed model matches with the T-SPICE simulations for varying widths and input voltages with a high degree of accuracy. Thus we can infer that by simply deriving the parameter 'B' for different widths, we can accurately predict the transistor drain current values for varying widths without the need of deriving an extra parameter 'K' as proposed in [5], thereby preserving the accuracy and simplicity of the SN model.

The model parameters derived for the ESN Model for NMOS and PMOS devices are listed in table I.

IV. Applications of the ESN Model

A. CMOS Inverter Delay

We now demonstrate the use of the ESN model in accurately predicting the propagation delay (t_{pHL}) of a CMOS inverter. Table II shows the comparison between simulated and calculated values of the propagation delay for the ESN and the SN model for varying device widths, output capacitances and input rise/fall times. We use the same expression for propagation delay as described in [4].

$$t_{pHL}, t_{pLH} = \left(\frac{1}{2} - \frac{1 - v_T}{1 + \alpha}\right) t_T + \frac{C_T V_{DD}}{2 I_{D0}} \qquad (11)$$

where, $v_T = \frac{V_{TH}}{V_{DD}}$, C_T is the total output capacitance discharged (C_L + parasitic capacitance), α is the velocity saturation index, t_T is the input rise/fall time and I_{D0} is the transistor drain current for V_{GS}=V_{DS}=V_{DD}, except that now it is calculated by the ESN model equation (8). The model parameters were derived for minimum-sized transistors for both the technologies.

As is evident from Table II, the SN model has an average error of 16.4% whereas, the ESN model has an average error of 1.84%. It is to be noted that for a given technology node, with increasing transistor widths, the SN model becomes increasingly inaccurate whereas the ESN model retains a high level of accuracy.

B. CMOS Gate Design

The propagation delay of a CMOS gate can be approximately estimated by converting it into an equivalent inverter. The size of the PMOS and NMOS used in the equivalent inverter represent the effective strengths of the actual pull-up and pull-down paths respectively.

For n series-connected transistors, the width-to-length ratio of the equivalent transistor is [10],

$$\left(\frac{W}{L}\right)_{equivalent} = \frac{1}{\sum_{n(on)} \frac{1}{\left(\frac{W}{L}\right)_n}} \tag{12}$$

Similarly for n transistors connected in parallel, the width-to-length ratio of the equivalent transistor is [10],

$$\left(\frac{W}{L}\right)_{equivalent} = \sum_{n(on)} \left(\frac{W}{L}\right)_n \tag{13}$$

Here, we consider a two-input CMOS NAND gate assuming that the circuit is driving a load capacitance of $100fF$ and needs to satisfy $t_{pHL,pLH} \leq 70ps$. Converting the gate into an equivalent inverter and substituting the parameters for 0.18μm technology in (8) we get,

$$I = B_{eq,n}(1.8-0.5421)^{1.0448}(1+0.0731\times 1.8) = 1.438B_{eq,n} \tag{14}$$

Considering the parasitics, the delay t_{pHL} is calculated using (11) with $t_T = 0$ we get,

$$t_{pHL} = \frac{117\times 10^{-15}\times 0.9}{1.438B_{eq,n}} \leq 70\times 10^{-12}s. \tag{15}$$

Hence, $B_{eq,n} \geq 1.046\times 10^{-3}A/V^{\alpha}$. From (10), the equivalent NMOS width comes out to be 2.5321μm and similarly the equivalent PMOS width comes out to be 5.5045μm. Using (12) and (13) for the NAND gate for worst case input, the width of the each NMOS comes out to be 5.0642μm and that of each PMOS comes out to be 2.7522μm. We get a delay of $68ps$ from T-SPICE simulations which meets the delay requirement of $t_{pHL} \leq 70ps$.

C. Elmore Delay of CMOS Inverter for Low Supply Voltages

The Sakurai-Newton (SN) delay approximation [4] is a widely used closed form delay metric for the CMOS gates because of simplicity and reasonable accuracy. In [1], it is shown that when the CMOS gate is modeled as an RC circuit, for a step input, the SN delay approximation is the exact Elmore delay of a CMOS gate, which can be viewed as the centroid of current. For a step input, the Elmore delay of a CMOS inverter can be expressed as [1],

$$t_{elmore} = \frac{\int_0^{\infty} tI(t)dt}{\int_0^{\infty} I(t)dt} \tag{16}$$

When $I(t)$ in the above equation is substituted by the drain current expression of the SN model, we get (17).

$$t_{elmore} = \frac{C_T V_{DD}}{k(V_{DD}-V_{TH})^{\alpha}} \approx t_{sn} \tag{17}$$

where t_{sn} is approximated by substituting I_{D0} with the SN model drain saturation current given by (3) and the input rise/fall time (t_T=0) in (11). Here $k = \frac{W}{L_{eff}}\mu_n C_{OX}$. The SN metric fails to accurately predict the delay of a CMOS inverter with varying supply voltages [6]. Thus it may not be applicable in many low power applications with voltage scaling.

In this section, we demonstrate how the centroid of current (17) can be modified according to the ESN model to accurately trace the SPICE delay values with varying supply voltage upto $V_{DD} = 3V_{TH}$. Shown below is the modified centroid of current based delay metric,

$$t_{ESN} = \frac{C_T V_{DD}}{2B(V_{DD}-V_{TH})^{\alpha}} \tag{18}$$

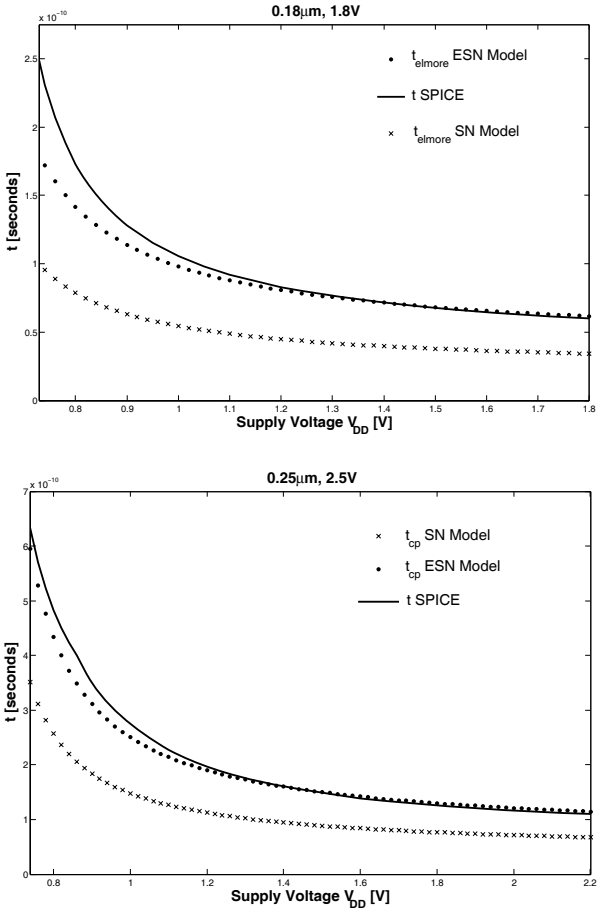


Fig. 4. T-SPICE delay and the values predicted by the modified centroid of current metric for CMOS inverter in 0.18μm and 0.25μm technologies. The supply voltage (V_{DD}) was varied in the range $2V_{TH}$ to $6V_{TH}$ with load capacitance C_L=$60fF$ (0.18μm) and C_L=$120fF$ (0.25μm), V_{TH}=$0.365V$ (0.18μm) and V_{TH}=$0.381V$ (0.25μm)

where, C_T is the total output capacitance discharged (C_L + parasitic capacitance), V_{DD} is the supply voltage, V_{TH} is the threshold voltage and α is the velocity saturation index. 'B' is the transistor width dependent transconductance parameter used in the ESN model equation (6). The above equation has been obtained by substituting the expression for drain current expression proposed by the ESN Model.

As shown in Fig 4, the modified centroid of current based metric matches closely with the SPICE delay values for fairly large variation of supply voltage from $3V_{TH}$ to $6V_{TH}$ with a high correlation coefficient of 0.98. However, it is unable to

accurately track the SPICE delay values for supply voltages below $3V_{TH}$.

Since at low supply voltages delay $\propto \frac{1}{V_{DD}^2}$, a delay metric based on the centroid of power dissipation was proposed as power $\propto$ (current)^2 [11]. In order to track the delay for supply voltage below $3V_{TH}$, we modify the centroid of power metric presented in [1] as per the ESN Model by replacing the parameter 'k' by the width dependent transconductance parameter 'B' of the ESN model equation (6). The modified expression is,

$$t_{cp_{ESN}} = \frac{C_T(3V_{DD}^3 + 3V_{DD}^2V_{TH} - 3V_{DD}V_{TH}^2 + V_{TH}^3)}{6BV_{DD}^2(V_{DD} - V_{TH})^\alpha} \tag{19}$$

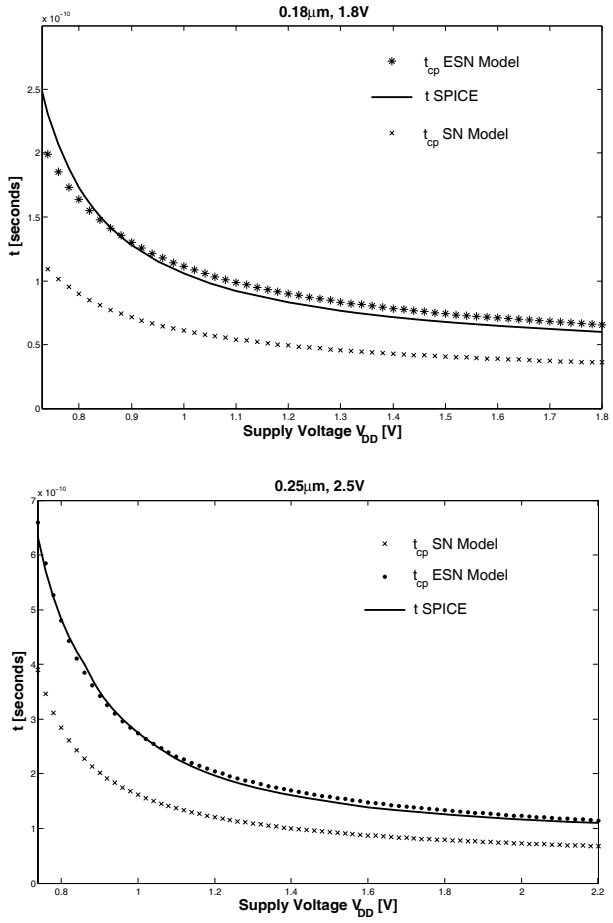


Fig. 5. T-SPICE delay and the values predicted by the modified centroid of power metric for CMOS inverter in 0.18μm and 0.25μm technologies. The supply voltage (V_{DD}) was varied in the range $2V_{TH}$ to $6V_{TH}$ with load capacitance C_L=60fF (0.18μm) and C_L=120fF (0.25μm), V_{TH}=0.365V (0.18μm) and V_{TH}=0.381V (0.25μm)

Fig. 5 shows that the modified centroid of power metric is able to accurately track the SPICE delay values for the entire range of supply voltage from $2V_{TH}$ and $6V_{TH}$ with a high correlation coefficient of 0.99. Hence, we have shown that by modifying the centroid of current based delay metric according to the ESN model, we can accurately predict the delay values for a fairly large range of supply voltage ($3V_{TH}$ to $6V_{TH}$). For supply voltage variation below $3V_{TH}$, the modified centroid of power based delay metric can be used.

V. Conclusion

In this paper we propose an extension to the Sakurai Newton model [3] for accurate estimation of MOSFET I-V characteristics in the deep submicron region for varying transistor widths. The new model originates from our observation that the constant transconductance parameter 'B' in the Sakurai Newton model can be modified into a width dependent parameter to incorporate the narrow width effects that are prevalent in DSM CMOS process technologies.

Our proposed model can accurately predict the propagation delay for CMOS inverters for a wide range of output capacitances, transistor widths and input ramp durations for 0.18μm and 0.25μm process technologies. Further, we have proposed modifications to the existing centroid-of-current and power based delay metrics presented in [1]. The modified centroid-of-current based metric tracks the SPICE delay with a correlation coefficient of 0.98 for supply voltages from $3V_{TH}$ to $6V_{TH}$ and the modified centroid-of-power metric tracks the SPICE delay with a correlation coefficient of 0.99 for supply voltages from $2V_{TH}$ to $6V_{TH}$. We anticipate the use of this model in applications where simple and quick analytical treatment of circuit behavior is required.

References

[1] Anand Ramalingam, Sreekumar V. Kodakara, Anirudh Devgan, and David Z. Pan, "Robust Analytical Gate Delay Modeling for Low Voltage Circuits", *Proceedings of the 2006 conference on Asia South Pacific design automation*, pp. 61-66, Yokohama, Japan, 2006.

[2] W. Shockley, *Proc. IRE*, vol. 40, pp. 1365-1376, Nov. 1952.

[3] T. Sakurai and A. R. Newton, "A simple MOSFET model for circuit analysis," *IEEE Transactions on Electron Devices*, vol. 38, no. 4, pp. 887-894, Apr. 1991.

[4] T. Sakurai and A. R. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE Journal of Solid State Circuits*, vol. 25, no. 2, pp. 584-594, April 1990.

[5] Makram M. Mansour, Mohammad M. Mansour, Amit Mehrotra, "Modified Sakurai-Newton Current Model and its applications to CMOS Digital Circuit Design", *Proceedings of the IEEE Computer Society Annual Symposium on VLSI* (ISVLSI'03).

[6] Y. Taur and T. H. Ning, *Fundamentals of Modern VLSI Devices*. Cambridge University Press, 1998.

[7] Hyunsik Im, M Song, T.Hiramoto and T.Sakurai, "Physical Insight into Fractional Power Dependence of Saturation Current on Gate Voltage in Advanced Short Channel MOSFETS (Alpha-Power Law Model)", *Proceedings of the 2002 international symposium on Low power electronics and design*, pp. 13-18, Monterey, California, USA, 2002.

[8] Keith A. Bowmann, Blanca L. Austin, John C. Eble, Xinghai Tang and James D. Meindl, "A Physical Alpha-Power law MOSFET model," *IEEE Journal of Solid State Circuits*, vol. 34, no. 10, pp. 1410-1414, October 1999.

[9] Kai Chen, Cheming Hu, Peng Fang, Min Ren Lin and Donald L. Wollesen, "Predicting CMOS Speed with Gate Oxide and Voltage Scaling and Interconnect Loading Effects, *IEEE Transactions on Electron Devices*, vol. 44, no.11, November 1997.

[10] Sung-Mo Kang and Yusuf Leblebici, *CMOS Digital Integrated Circuits Analysis and Design*, 3rd ed. New Delhi: Tata McGraw-Hill, 2003.

[11] Anand Ramalingam, Sreekumar V. Kodakara, Anirudh Devgan, and David Z. Pan, "Robust Analytical Gate Delay Modeling for Low Voltage Circuits", *ASP-DAC 2006 Archives*, 1C-1, 2006.

Measurement and Analysis of Parasitic Capacitance in FinFETs with high-k dielectrics and metal- gate stack

Abhisek Dixit [(1)], Anirban Bandhyopadhyay [(1)], Nadine Collaert [(2)], Kristin De Meyer [(2)], Malgorzata Jurczak [(2)]

(1) SRDC-Compact Modeling Group, IBM India Pvt. Ltd., D3-First Floor, Manyata Embassy Business Park, Nagwara Circle, Outer Ring Road, Bangalore-560045, INDIA.

(2) IMEC, Kapeldreef 75, 3001 Leuven, Belgium.

abhdixit@in.ibm.com

Abstract

FinFET is one of the promising device architectures for sub-32nm CMOS technology nodes. These non-planar devices benefit from near bulk-Si processing and improved control of short channels due to quasi gate-all-around operation. Their device operation is well studied and optimized in last half decade by various research groups. In this paper, we help evaluate the circuit potential of FinFETs by experimentally comparing their parasitic capacitance to that of the planar FDSOI MOSFETs. It is shown that n- and p-channel FinFETs achieve as high as 50% and 28% parasitic capacitance reduction compared to the planar FDSOI MOSFETs respectively.

1. Introduction

CMOS inverter is considered the basic circuit building block in various CMOS applications [1],[2]. Although multiple contributions, such as the branching effort and electrical effort determine the overall logical effort of an inverter in a given circuit configuration, the concept of electrical effort is pertinent to device parasitics [3]. The electrical effort of an inverter (H) is defined as the ratio of its output to the input-capacitance. Critical logic paths in a digital design require minimum delay. Consequently, logic circuits often employ buffer chains as is schematically shown in Fig. 1.

In this context, a buffer is an inverter, sized to a designed width (W_i) to drive a certain capacitive load. Thus, in a chain of inverters, the subsequent inverters (fan-out) determine the output capacitance while the preceding inverters (fan-in) determine the input capacitance of the inverter. However, as seen from Fig. 1, the gate-to-source (C_{gs}) and the gate-to-drain (C_{gd}) components of MOSFET's parasitic capacitance act as an output to input capacitive coupling in an inverter. For digital applications, these capacitances, represented by C_p in Fig. 1(A), can be critical to the propagation delay and power dissipation. For analog applications, a negative feedback through them will affect the gain-bandwidth product. Thus, an accurate estimation of these parasitic capacitances is required to evaluate a MOSFET's circuit application potential.

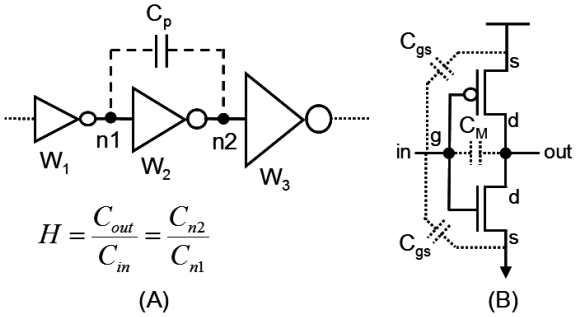


Figure 1: Relevance of MOSFET parasitic capacitance to the digital circuit design: (A) Concept of buffer chains and electrical effort `H' (B) CMOS inverter parasitic capacitances. C_{gs} is the gate-to-source capacitance and C_M is the gate-to-drain capacitance in presence of the Miller effect. Together these components form the capacitance C_p shown in (A).

FinFETs with their excellent short-channel control in un-doped fin configuration may provide an optimal solution for scaled-LSTP technologies [4]-[7]. Process induced FinFET parasitics have been addressed and it was shown that with required control on Fin-LER, FinFET SRAM cells could achieve superior stability than their planar bulk counterpart [8]. Amongst FinFET device parasitics, the S/D resistance was found to be dominant and raised S/D architecture has been reported as the viable solution [9]-[11].

The objective of this paper is to aid to the evaluation of circuit prospects of FinFETs by analyzing the missing link, namely the parasitic capacitance. The analysis is performed by comparing FinFET parasitic capacitance with that of the planar SOI MOSFET. Experimental data is combined with popular planar MOSFET models

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.80

and extraction methodologies to address FinFET parasitic capacitance.

2. Test Structures and Device Fabrication

Various capacitances encountered in a MOSFET are shown in Fig. 2, where C_{ox} is the intrinsic gate oxide capacitance, C_{do} is the direct capacitance in the gate-to-source overlapped region, C_{if} is the inner fringing capacitance [12]. C_{of} is the sidewall fringing capacitance associated with electric field lines emerging from the sides of poly-Si gate, going through the sidewall spacer and ending at S/D regions. Further, C_{pp} is the parallel plate capacitance associated with electric field lines emerging from the sides of poly-gate, going through the side-wall spacer and ending at S/D electrodes (contact plugs) [13]. C_{top} is the top fringing capacitance associated with field lines emerging from top surface of the poly-gate and going through the first layer of Passivation/planarization dielectrics and ending at S/D electrodes (contact plugs) [14].

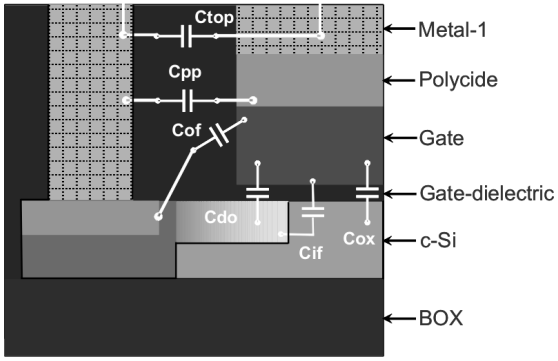


Figure 2: Capacitive components in a FDSOI MOSFET. Utilizing the symmetry, only left-half of the device is shown. BOX is Buried Oxide.

Estimation of parasitic capacitances in a MOSFET serves many purposes, e.g. it provides a way to compare efficiency of different layouts, it is essential for accurate estimation of gate capacitance and to some extent it reflects effective channel length of a MOSFET etc. Various models and experimental approaches to estimating parasitic capacitance of planar bulk MOSFETs exist [13], [15]-[17]. However, FinFET being an alternative and SOI architecture is different from the planar bulk MOSFET both in terms of the layout as well as layer stack. Therefore, it is likely that many of these capacitances have different values in FinFETs, knowing which the FinFET layouts can be optimized and future FinFET parasitic capacitance models can be appropriately developed.

Unlike the well contact in planar bulk MOSFETs, FDSOI architecture of FinFETs does not facilitate a contact to fins. As a result, only an inversion C-V analysis can be performed on FinFETs, unless other dedicated test structures are utilized. Also, many of the methodologies that are commonly used in planar bulk MOSFETs for accurately measuring the gate C-V are based on the C-V analysis in the accumulation region, and hence are inapplicable to FinFETs [18],[19]. Therefore, to accurately measure the C-V characteristics of FinFETs, special test structures are designed, as is shown in Fig. 3. As the purpose of this analysis is to compare the FEOL capacitances between FinFET and planar SOI transistors, the reference structure shown in Fig. 3(C) is utilized for removing C_{top}, C_{pp} and other buried oxide related capacitive components from the C-V curves measured on structures shown in Fig. 3(A) and (B).

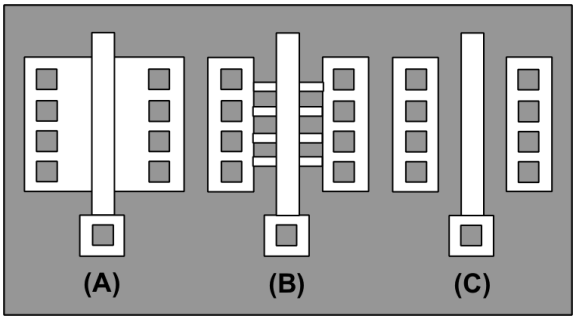


Figure 3: Device structures utilized for capacitance analysis: (A) Planar SOI transistor (B) SOI-FinFET (C) Reference Structure. Mask patterns for SOI, poly-Si gate and CA layers are shown.

Depending on the gate-to-S/D overlap area, the parasitic capacitance is usually a small number in comparison to the intrinsic gate capacitance. As most LCR meters utilized for the C-V measurements have a limited resolution for small capacitances (a few pF), it is important to keep the gate to S/D overlap area sufficiently large. In the planar SOI structure shown in Fig. 3(A), an equivalent gate width of approximately 14,000µm is targeted. The FinFET structure shown in Fig. 3(B) is designed for fin pitch (S_{fin}) of 0.35 and 0.2µm with 40,000 fins (fin height H_{fin}=62nm, fin width W_{fin}≈80nm) and 69,200 fins (H_{fin}=62nm, Wfin≈53nm) respectively. These devices have been fabricated utilizing a process-flow, as is briefly outlined in Table 1. Designed process splits are also shown in Table 1.

Process step or measurement	'D03'	'D04'	'D08'
2 nm Sacrificial Oxidation at 700^0C	X	X	X
H_2-anneal 900^0C	X	X	X
Gate dielectric	HfSiON	SiON	HfSiON
MOCVD TiN metal Thickness (nm)	5	10	5
Extension implantations	X	X	X
W_{spacer}	45 nm	45 nm	45 nm
HDD implantations	X	X	X
Activation anneal 1050^0C, Spike	X	X	X
Thickness of Ni (nm) used for S/D silicidation	6	6	6
Contact-hole Etch Stop Layer (CESL)	X	X	

Table 1: Process splits and flow for parasitic capacitance experiments. Contact Etch Stop Layer (CESL) is utilized to impart a uniaxial strain in the channel.

3. Measurement and Extraction of FEOL Parasitic Capacitances in FinFETs

To measure the inversion C-V characteristics of the fabricated structures, devices have been biased as shown in Fig. 4.

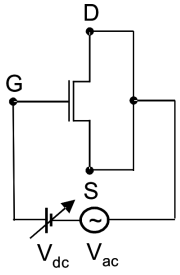


Figure 4: Transistor biasing for the inversion C-V measurements. A 30 mV, 1 MHz ac signal is utilized.

In these measurements, the S/D terminals are grounded with respect to the gate. A small ac signal (30 mV, 1 MHz) is superimposed on a slowly varying ramp, which is applied to the gate. In C-V measurements, actual bias is applied to the largest electrode in the device, e.g. the bulk contact in a planar bulk MOSFET. In this case of SOI FinFETs, the actual bias is applied on the gate terminal, as otherwise significant attenuation of the ac signal is seen to result in low measured capacitances.

Introduction of new gate-stacks, e.g. high-k dielectrics and metal gate, to the CMOS technology requires a careful investigation of existing capacitance extraction methodologies. In addition, for novel architectures, e.g. FinFETs, the analysis becomes even more vital. Gate dielectrics in modern CMOS technology are targeted to small EOT values, hence these are thin. With an appreciably large gate bias applied across them, these thin dielectrics often cause leakage and distort the measured C-V curves. An approach to understanding the usability of measured C-V curves is through the phase angle analysis [20]. Phase angles for the measured C-V curves are extracted. One such extracted phase angle vs. gate bias curves is shown in Fig 5.

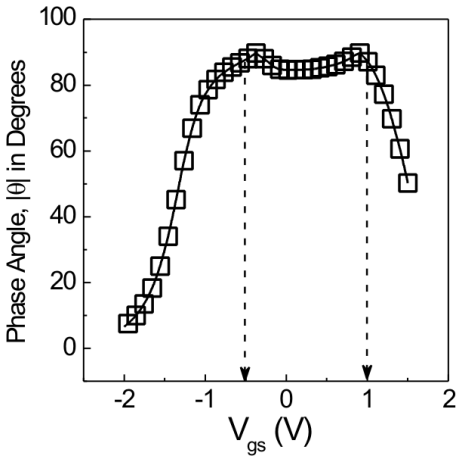


Figure 5: Validation of the C-V measurement using the phase angle analysis for the FinFET structure shown in Fig. 3(B). Measured n-channel FinFET has L_{gate}=0.46µm, W_{fin}=0.08µm, S_{fin}=0.35µm and number of fins (N_{fin})=40,000.

Since the capacitance is defined as the imaginary component of admittance (Y), a value of phase angle (Θ) close to 90^0 is desired. In the case of leaky dielectrics, a high frequency ac signal keeps the dissipation small and consequently achieves high values of Θ. C-V curves, measured on the reference structure, as shown in Fig. 3(C), are first subtracted from those measured on planar SOI and FinFET structures shown in Fig. 3(A) and (B). Consequently, parasitic capacitances associated with the S/D/G pads, i.e. C_{top} and C_{pp} in Fig. 2 and buried oxide are removed from the measured C-V curves. Measured C-V curves after this correction are shown in Fig. 6. In the phase angle analysis, acceptable portion of the curve shown in Fig. 5 is marked by the dotted arrows. Beyond this region, high gate bias can be understood to cause high leakage and perhaps unusable capacitance values. Thus, a minimum to maximum workable gate bias range is obtained for the curves shown in Fig. 6.

Further, these curves are normalized to effective gate width, as is shown in Fig. 7.

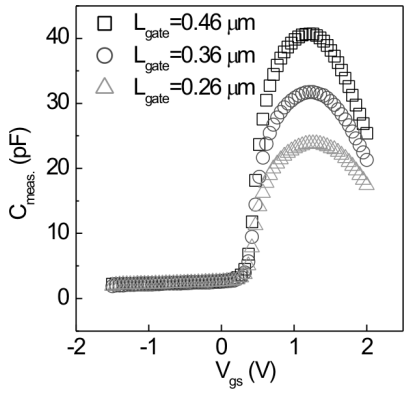


Figure 6: Measured FinFET C-V curves. Measured n-channel FinFETs have W_{fin}=0.08µm, S_{fin}=0.35µm and N_{fin}=40,000.

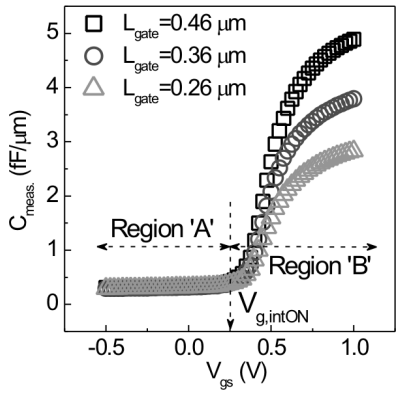


Figure 7: FinFET C-V curves shown in Fig. 6, normalized to the effective gate width.

A key guideline in separating parasitic capacitance from intrinsic gate capacitance is that the gate capacitance (normalized to the effective gate width) is a function of gate length, whereas the parasitic capacitance is not. Accordingly, the C-V curves shown in Fig. 7 are independent of L_{gate} in the Region `A', and vice versa in Region `B'. The bias point $V_{g,intON}$ that marks the boundary between the two regions in Fig. 7 provides a reference for extracting various components of the parasitic capacitance [15],[16]. To obtain the total parasitic capacitance C_{para} and effective channel length (L_{eff}), an iterative analysis is performed. In this analysis, L_{eff} is defined as

$$L_{eff}=L_{gate}-\Delta L_{eff} \quad \text{.....Equation 1.}$$

Here ΔL_{eff} is the overall reduction in L_{gate} due to metallurgical overlap between the gate and S/D extensions and other carrier injection considerations [16]. Measured capacitance values shown in Fig. 7 are plotted as a function of L_{eff} for all the bias points in Fig. 8.

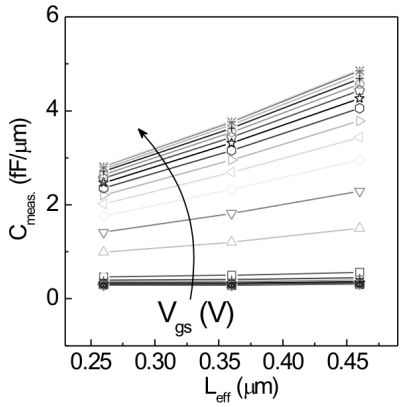


Figure 8: Measured capacitance, as shown in Fig. 7, plotted as a function of the effective channel length, L_{eff}.

First a dummy value of ΔL_{eff} is inserted in Eq. 1 and curves as shown in Fig. 8 are plotted. The Y-axis intercepts of the linear fits at ΔL_{eff} =0 provide an estimate of C_{para} at various bias points. These intercepts, i.e. C_{intc}, are plotted as a function of the gate bias in Fig. 9.

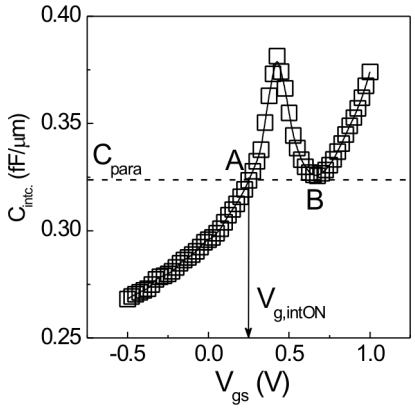


Figure 9: Y-intercepts, extracted from the curves shown in Fig. 8, are plotted as a function of the gate bias. Analysis on an n-channel FinFET with L_{gate}=0.46µm is shown.

Value of ΔL_{eff} in Eq. 1 is adjusted by iterating this procedure unless C_{intc} at $V_{gs}=V_{g,intON}$ (i.e. at point `A') equals the minimum $C_{intc}$ at $V_{gs}>V_{g,intON}$ (i.e. point `B') in Fig. 9. Among the parasitic capacitances shown in Fig. 2, C_{top} and C_{pp} are already removed from C_{intc} by

subtracting the pad capacitances. Of the remaining C_{of} is bias independent. Hence, the minimum value of C_{intc} in Fig. 9 represents C_{of}. In Fig. 9, as the gate bias increases from -0.5 V, the gate-to-S/D overlapped regions (n^+) go from depletion towards accumulation mode and C_{intc} keeps increasing. At this stage, both C_{do} and C_{if} contribute to C_{intc}. However, beyond point `A' a weak inversion in the channel is reached. Once the inversion layer is formed, it effectively screens electrical field lines originating from bottom of the gate and terminating at the inner periphery of the S/D extensions, thus suppressing C_{if} [12]. Due to this, C_{intc} comes back to the point `B' in Fig. 9. Beyond point `B', impact of gate leakage is present and C_{intc} is seen to increase with gate bias again. After removing C_{para}, which comprises of C_{of} and C_{do}, from the curves shown in Fig. 6, corrected C-V curves are normalized to the effective channel area using the knowledge of ΔL_{eff}. These gate C-V curves are shown in Fig. 10.

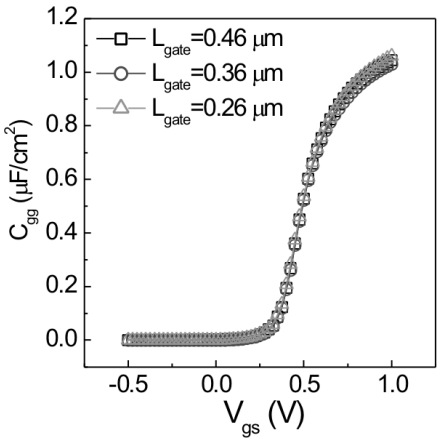


Figure 10: C-V curves, as shown in Fig. 6, after correction for C_{para} and normalization to the effective gate area, taking ΔL_{eff} into consideration.

Since C_{if} is gate bias dependent and exists only up to the weak inversion regime, $C_{para}=C_{of}+C_{do}$ can be taken as the effective FEOL parasitic capacitance of FinFETs. It should be noticed that the post-processed C-V curves shown in Fig. 10, were originally measured on devices with variable gate lengths. Assuming that all the parasitics are removed and capacitances are normalized to the effective channel area, these curves should be identical, as is seen in Fig. 10, which validates this analysis. The same analysis is performed on various device structures, as listed in Table 2.

Name	Device	Architecture
NPL	NFET	Planar, equivalent gate width=13995μm
PPL	PFET	Planar, equivalent gate width=13995μm
NFF200	NFET	MuGFET, N_{fin}=69200, S_{fin}=0.2μm, W_{fin}=0.053μm
PFF200	PFET	MuGFET, N_{fin}=69200, S_{fin}=0.2μm, W_{fin}=0.053μm
NFF350	NFET	MuGFET, N_{fin}=40000, S_{fin}=0.35μm, W_{fin}=0.08μm

Table 2: Nomenclature of device structures utilized for parasitic capacitance extractions. S_{fin} is the fin pitch and H_{fin}=0.062µm for all the structures. MuGFET is a synonym of FinFET.

Results of parasitic capacitance extractions on these structures are summarized in Fig. 11, where extracted C_{para} is shown. It can be seen from Fig. 11 that n- and p-channel FinFETs (NFF350 and PFF200) achieve as high as 50% and 28% parasitic capacitance reduction compared to the planar FDSOI MOSFETs (NPL and PPL) respectively.

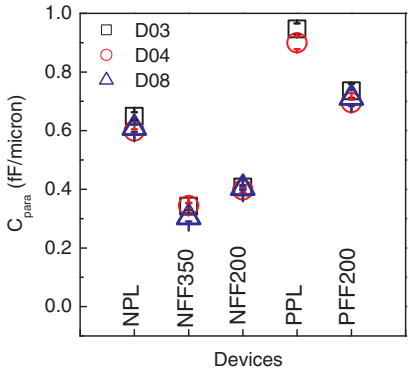


Figure 11: Parasitic capacitances as extracted for various device structures mentioned in Table 2.

4. Conclusion

FEOL components of FinFET parasitic capacitance have been analyzed on fabricated device structures. A parasitic capacitance extraction methodology, commonly used for planar devices, has been utilized. In presence of optimal de-embedding of BEOL capacitances and appropriate planar reference structures, it is shown that FinFETs with varying fin pitch have lower parasitic capacitance than their planar SOI counterparts. Further, the parasitic capacitance is shown to be higher for the pFETs with respect to nFETs, as usually is the case with planar MOSFTEs too. Future experiments shall be carried out to analyze the impact of raised S/D architecture, utilized to reduce FinFET S/D resistance, on FinFET parasitic capacitance. However, our present analysis concludes

that FinFETs have better potential for digital circuit applications over the planar FDSOI MOSFETs.

5. References

[1] N. H. E. Weste et al, Principles of CMOS VLSI Design: A Systems Perspective, by Addison Wesley Publication Company, 1998.

[2] J. M. Rabaey, Digital Integrated Circuits: A Design Perspective, by Prentice Hall of India Pvt. Ltd., 2000.

[3] I. Sutherland et al, Logical Effort: Designing Fast CMOS Circuits, by Morgan Kaufmann Publishers Inc., 1999.

[4] B. Yu et al, IEDM Tech. Dig., pp. 251, 2002.

[5] B. S. Doyle et al, IEEE Electron Device Lett., 24(4), pp. 263, 2003.

[6] F. -L. Yang et al, 25 nm CMOS Omega FETs, IEDM Tech. Dig., pp. 255,2002.

[7] A. Asenov, IEEE Trans. Electron. Devices, 45(12), pp. 2505, 1998.

[8] A. Dixit et al, IEDM Tech. Dig., pp. 709, 2006.

[9] N. Collaert et al, IEEE Electron Device Lett., 25(8), pp. 568, 2004.

[10] J. Kedzierski et al, IEEE Trans. Electron. Devices, 50(4), pp. 952, 2003.

[11] A. Dixit et al, IEEE Trans. Electron. Devices, 52(6), pp. 1132, 2005.

[12] Y. Taur et al, Fundamentals of Modern VLSI Devices, by Cambridge University Press, 1998.

[13] N. R. Mohapatra et al, IEEE Trans. Electron. Devices, 50(4), pp. 959, 2003.

[14] C. H. Wang et al, IEEE Trans. Electron. Devices, 43(6), pp. 965, 1996.

[15] J.-C. Guo et al, IEEE Trans. Electron. Devices, 41(10), pp. 1811, 1994.

[16] S. Severi et al, IEEE Electron Device Lett., 27(7), pp. 615, 2006.

[17] K. Romanjek et al, IEEE Electron Device Lett., 25(8), pp. 583, 2004.

[18] K. J. Yang et al, IEEE Trans. Electron. Devices, 46(7), pp. 1500, 1999.

[19] Z. Luo et al, IEEE Electron Device Lett., 25(9), pp. 655, 2004.

[20] J. Lin et al, IEEE Conference on Microelectronic Test Structures, pp. 289, 2004.

Session 4B

Application-Specific Architectures and Reconfigurable Computing

Design, Implementation and Validation of an Open Source IP-PBX/VoIP Gateway SoC

S. Apostolakos, G. Lykakis, A. Meliones, V. Vlagoulis, E. Touloupis, G. Konstantoulakis
inAccess Networks, 12 Sorou Str., Maroussi 15125, Athens, Greece
{spapost, glyk, meliones, vlv, etoul, gkonst}@inaccessnetworks.com

Abstract

The telephony world is consistently moving to the transmission of voice through packet networks, so as to unify data and voice and to enable the provisioning of new services in a less costly manner. Service providers are offloading the task of converting analog voice to VoIP to the end-points. This allows the ISPs and ITSPs to reduce their costs and increase the uniformity of their interfaces with their clients. In this paper we present an IP-PBX/VoIP Gateway system based on a single SoC that performs all the required processing. This SoC includes a CPU for hosting a full-fledged operating system and user applications, as well as a DSP subsystem for voice processing. The system targets the low density market of home gateways and SME IP-PBXs, where cost is the main factor. We prove it is feasible to implement a 2-4 channel IP-PBX/VoIP gateway on a SoC based purely on both software and hardware provided by the open-source community, reducing both upfront and final product costs thus allowing new players into the market.

1. Introduction

The telephony world is consistently moving away from the traditional, circuit-switched architecture to the transmission of voice through packet networks, so as to unify data and voice and to enable the provisioning of new services in a less costly manner.

Moreover, the service providers are offloading the task of converting analog voice to VoIP to the end-points, owned by their customers, such as the home gateway, at present offering the capacity of connecting up to 2 telephones as well as the SME and corporate PBXs, which are becoming IP-enabled. This allows the ISPs and ITSPs to reduce their costs and increase the uniformity of their interfaces with their clients. Nowadays, a broadband connection (either ADSL or wireless) suffices to convey voice, data and video to/from the home and business users. This de-centralization approach has entered the standards with the introduction of the IP Multimedia Subsystem (IMS) architecture [1].

In this paper we present an IP-PBX/VoIP Gateway system, based on a single SoC (called ERMES) that performs all the required processing. This SoC includes a CPU, for hosting a full-fledged operating system and user applications as well as a DSP subsystem for voice processing complemented with hardware accelerator logic for special functions. This system targets the low density market (2-4 concurrent voice channels) of home gateways and SME IP-PBXs, where cost is the main factor. To this extent, we have decided to rely not only on an open-source software platform, but also to use an open source CPU core, because of the prohibitive costs involved in licensing a commercial CPU core (e.g. ARM-based). LEON3, the latest version of the LEON core (see [2]), a SPARC V8 compliant, 32-bit RISC processor, having been used successfully in a number of SoC designs, was selected as the most suitable candidate.

Even though the LEON3 core is totally suitable for the execution of the operating system and user space applications, the envisioned SoC is also supposed to execute signal processing tasks, which a RISC processor is not expected to handle too well. This is why such targeted SoCs rely on the integration of specialized DSP cores, which are equally, if not more costly, compared to CPUs. The issue then is whether and to what extent the selected processor can also be used for signal processing.

2. Necessary algorithmic support for the ERMES SoC

Based on an extensive set of customer requirements collected over the years, the minimum DSP algorithmic support necessary to implement an IP-PBX/VoIP gateway SoC includes 4 voice processing and 2 telephony signaling algorithms: the G.711 encoder/decoder, the G.729A encoder/decoder and the telephony tone signaling tone generator/detector.

The G.711 encoder/decoder functions as a compander (compressor/expander). Audio data is encoded after

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.47

logarithmic scaling so that better precision is achieved in the representation of low amplitudes as opposed to higher ones, due to the increased sensitivity of the human ear in these low amplitudes. G.711 operates on a sample-per-sample basis and its data rate is 64 Kbits/sec.

G.729A is a parametric codec. It operates on 10 ms voice frames and parameterizes voice in a way that achieves high compression rates without compromising voice quality. Its data rate is 8 Kbits/sec.

The signaling tone generator must have the capability of producing any sum of 3 sinusoids with programmable frequencies and amplitudes provided that their frequencies are within the telephony bandwidth (300-3400 KHz). It must also support up to 3 programmable on-off intervals (cadences).

The signaling tone detector must detect the DTMF tones used during call setup, following specific requirements defined in the relevant standards.

3. Benchmarking of the DSP algorithms on LEON3

In order to identify whether the LEON3 core can serve as the processing element implementing such algorithmic blocks, an HDL simulator was used to measure the number of clock cycles required for DSP software execution.

The voice processing algorithms were first implemented in ANSI C and executed on a PC so as to validate their functionality according to relevant standards using specific test vectors. Then, they were integrated with appropriate LEON3 initialization functions and executed on the simulator. The results are summarized in Table 1. The G.729A codec was executed on a 10 ms voice frame (100 frames per second), while all other algorithms were executed on a 5 ms voice frame (200 frames per second).

Table 1. LEON3 cycles and instructions when executing the ERMES DSP algorithms

Algorithm	Cycles
Linear to G.711 A-law encoding (5 ms frame)	6508
Linear to G.711 u-law encoding (5 ms frame)	6158
G.711 A-law to linear decoding (5 ms frame)	1788
G.711 u-law to linear decoding (5 ms frame)	1499
Linear to G.729A encoding (10 ms frame)	1781032
G.729A to linear decoding (10 ms frame)	588112
Signaling tone (DTMF) detector (5 ms frame)	151247
Signaling tone generator (5 ms frame)	14455

Commenting on the table presented above, the implementation of a full-duplex voice channel requires some 236,914,400 cycles every second, since concurrent execution of a voice encoder (G.729A is the worst case) and a voice decoder (G.729A still the worst case) is required. The tone generator and DTMF detector operate only during call setup and the required processing power is significantly lower. This cycle figure in turn means that even if an operating frequency of 400 MHz is achieved for the SoC in an ASIC, not even a second concurrent voice channel could be supported.

4. Related work

What our analysis indicates up to this point is that it is desirable to complement the LEON3 core with DSP hardware support so as to improve its DSP performance and consequently the achievable voice channel density.

The idea is not new, since it is getting quite common to equip general purpose embedded CPUs with DSP capabilities. Most of the modern ARM cores feature certain DSP commands, like single-cycle multiply/ accumulate etc. An implementation of the G.729 codec family on ARM architecture with DSP extensions is reported in [3].

As far as the LEON core is concerned, [4] presents the idea of attaching scalar and vector coprocessors to the LEON core via a custom pipeline coprocessor port. Other approaches we have evaluated include (i) the implementation of the DSP logic internally to LEON and the definition of new instructions for its use and (ii) the implementation the DSP logic within an external peripheral unit.

Eventually, the latter option was selected because of a number of advantages it presents: (a) its operation is not dependent on the host processor and can be used as an autonomous peripheral unit in other systems; (b) it can operate in parallel with the CPU, however it needs proper software design for increased performance; (c) there is no need to go into time-consuming non-trivial modifications of LEON3 that can yield design faults that are not easily traceable and affect the system operation. The option of the mathematical co-processor was judged insufficient because the relevant LEON3 interface cannot support high data rates. On the other hand, DMA can be employed to implement transfers to/from the peripheral DSP unit.

5. The ERMES SoC architecture

The ERMES SoC architecture is depicted in Figure 1. All main interconnections are shown, while trivial details have been left out for the sake of readability (e.g. IRQ signals from peripheral units to the interrupt controller).

In order to detach the execution of DSP tasks from other software, since the execution needs (real-time/non real-time) differ considerably, a second LEON3 core has been employed as a DSP processor, together with the abovementioned peripheral unit (denoted as the Vector Arithmetic Unit – VAU in the ensuing). This LEON3 configuration used for that purpose is slightly different than the one used for the CPU, lacking the Memory Management Unit and using smaller instruction and data caches.

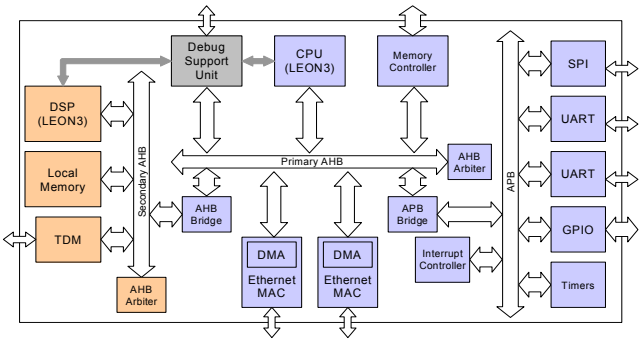


Figure 1. ERMES SoC architecture

ERMES features a dual AMBA AHB architecture (as per [5]), where a primary system bus serves the CPU and 2 Ethernet units, and a secondary system bus serves the DSP subsystem, which is further analyzed in section 6. Each bus employs its own arbitration unit and the two buses are interconnected via a bridge. This topology allows the units to communicate as if they were located on the same bus, while at the same time it allows the parallel functioning of the CPU and DSP subsystems, thus increasing the system throughput. Furthermore, an APB bus is used for the transport of control information as well as data transfer to/from the low data rate units.

The primary AHB bus interconnects the CPU (master), the external memory controller (slave), the AHB (master/slave) and the APB (slave) bridges, the Debug Support Unit (master/slave and direct interfacing with both processors) and the 2 Ethernet units (masters for data and slaves for control). The bus arbiter uses a round-robin algorithm.

The secondary AHB bus interconnects the DSP (master), the local memory (slave), the VAU (master/slave), the TDM unit (master) and the AHB bridge (master/slave). The bus arbiter uses a fixed priority algorithm, with higher priority assigned to the TDM unit. The APB bus interconnects the SPI unit, the UART transceivers, the GPIO unit, the timers, and the interrupt controller. The APB bus is accessed via the APB bridge, which also decodes the selection signals.

6. DSP subsystem architecture

The DSP subsystem is one of the key elements of the ERMES SoC. As depicted in Figure 2, it consists of the DSP (which is actually a LEON3), a local memory, the VAU autonomous peripheral unit and the TDM unit, which provides communication with external devices.

The VAU can greatly enhance the DSP capability of the ERMES SoC, since it is capable of performing vector arithmetic operations adapted to the particular requirements of voice processing. The unit exploits the LEON 32-bit system architecture so that it can transfer and process two 16-bit operands in parallel.

The VAU has the following modes of operation:

- **Vector Multiply and Accumulate (MAC) with 16-bit operands**. The VAU unit is programmable so as to: (i) Initialize the accumulator or not; (ii) Store the accumulator value in memory for every iteration or not; (iii) Hold one of the operands to a constant value for all operations or not; (iv) Add or subtract the result of each iteration from the accumulator, and (v) Use an accumulator 32 or 64 bits wide.
- **16×32 bits vector multiply**. The result is 32 bits wide. The unit is programmable, so that the 16 bit operand is held to a constant value or not.
- **32×32 bits vector multiply**. The result is 32 bits wide. The unit is programmable, so that one operand is held to a constant value or not.
- **Bit manipulation**. The VAU can perform vector bit insertion and extraction, left or right shifting with or without rounding.
- The VAU uses saturation arithmetic to carry out all abovementioned operations and can inform the DSP on probable overflows.

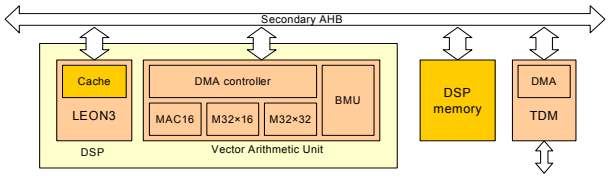


Figure 2. The DSP subsystem architecture

The TDM unit supports up to 128 TDM channels, has DMA access to the DSP memory and supports Least/Most Significant Bit first transfers with a programmable sample bit width.

7. SoC prototyping

The ERMES SoC has been prototyped using a Xilinx Spartan3 4000 FPGA, comprising of 4M gates, 27.648 slices and 1.728 Kbits of block RAM. Each slice contains 2 programmable Look Up Tables (LUTs), 2 flip-flops, multiplexers and arithmetic logic. The Xilinx ISE tool has been used for synthesis. The maximum operating frequency is 42 MHz with approx. 60% total utilization of the FPGA. The distribution of the FPGA resources among the various system units is depicted in Table 2.

8. System architecture

In order to validate the performance of the ERMES SoC presented in the previous sections, a system has been developed to host it. It consists of specialized hardware as well as software for the CPU and firmware for the DSP.

Table 2. FPGA resources allocated to SoC units

System Unit	LUTs	Flip-flops	Block RAMs
CPU	6590	2510	17
DSP	5400	2100	10
VAU	3300	860	-
DSP memory	20	12	32
Ethernet	2650	1450	1
TDM	1730	890	-
Memory controller	650	430	-
Debug Support Unit	1040	490	2
Interrupt controller	470	100	-
Timers	330	160	-
AHB/APB bus logic	980	380	-
UART	260	140	-
SPI	240	130	-

8.1. System hardware

The system hardware is built around the Xilinx FPGA and supports SDRAM and Flash memories on a 32-bit system bus, 2 general purpose RS232 interfaces, 1 dedicated RS232 interface for debugging, 2 Ethernet PHYs connected to the SoC MII ports and 4 FXS codecs, which interface the system with telephony devices, dealing with analog signal processing and conversion between analog and digital signals and offering basic analog telephony functions, such as ringing, off/on-hook detection and echo cancellation. Voice data are transferred to/from the SoC via a TDM bus, while an SPI bus is used for controlling the FXS codecs. The PCB also includes the power supply, clocking and reset circuits for the SoC and its peripherals.

8.2. DSP firmware

The architecture of the DSP firmware is depicted in Figure 3. It comprises of voice processing algorithms (signaling tone generator/detector, voice encoders/decoders), data transfer interfaces (to/from the TDM unit and the CPU), a signal router and mixer module and a scheduler. A double buffering mechanism is employed to simplify data transfers, providing stability and synchronization thus prohibiting data losses. The buffer lengths are configurable at boot time at an even integer number of samples. Preferable values are 40 16-bit words (5 ms interval @ 8kHz sampling rate) for communication with the TDM unit and 64 16-bit words for communication with the CPU. The latter are larger to accommodate VoIP requirements for voice-concurrent tone signaling and redundant voice encoding. All metrics presented in the paper have been extracted for 40/64 16-bit word buffering

The major DSP firmware control entities are: (i) the synchronization module, which synchronizes the various tasks with the periodic TDM interrupt requests, (ii) the channel status table, which is used by the CPU to configure and read the status of each of the supported voice channels, (iii) the scheduler, which invokes the appropriate signal processing functions, and (iv) the signal router and mixer, which controls the successive processing steps per signal. The architecture can be seamlessly expanded to any number of channels that the DSP processing power and memory can support.

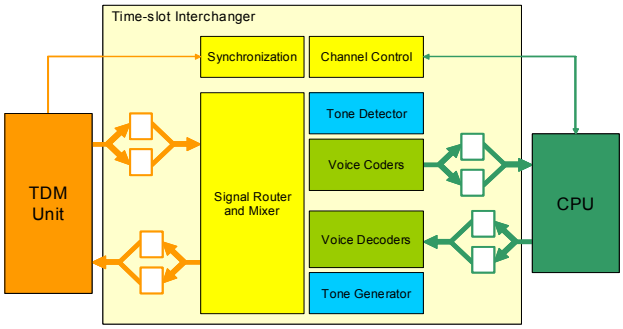


Figure 3. DSP firmware architecture

The communication between the CPU and the DSP is realized over the DSP local memory. The communication protocol is based on the exchange of packets carrying control and data information. Synchronization is achieved with the periodical interrupts produced by the TDM unit. The control of the communication is performed through the DSP firmware scheduler and the CPU driver of the telephony subsystem.

8.3. CPU software

The CPU software is based on the Linux 2.6 operating system. The general software architecture is presented in Figure 4. ERMES runs the Linux kernel 2.6.11 with several required services and applications. Most of them are included in the BusyBox toolset, which combines tiny versions of many common GNU utilities into a single small executable. The options that are included provide the expected equivalent functionality.

At the lower level, the device drivers implement the communication between the OS and the system specific hardware via standardized Linux interfaces. The ERMES software employs device drivers for the Ethernet, UART and telephony devices. The telephony drivers include device drivers for controlling the FXS codecs over the SPI, as well as a driver for the communication with the DSP.

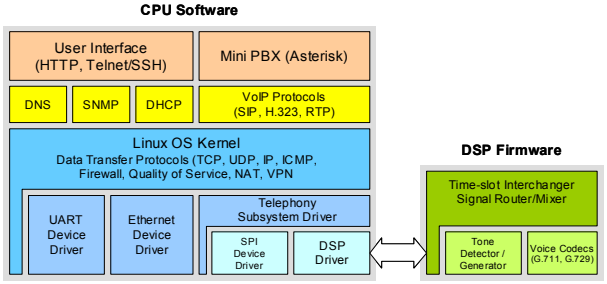


Figure 4. CPU software architecture

The telephony software is built around Asterisk, an open-source software which implements a mini PBX in a Linux environment [6]. It provides all basic PBX services and a lot more typically found in high spec systems. It allows calls to/from the attached telephony devices and their interconnection with telephony services, including VoIP. To this extent, it supports SIP and H.323 signaling and the RTP protocol. It features a flexible architecture based on a central call switch and a list of software functions which can be combined dynamically, depending on application needs. It can collaborate with the majority of standardized telephony equipment, if combined with the appropriate hardware and device drivers.

9. System performance evaluation

In order to evaluate the performance of the system employing the ERMES SoC, 4 sets of tests have been carried out. First, the algorithmic performance has been evaluated against relevant standards, to verify that they operate as expected. Then, the VAU performance was measured in an HDL simulator in order to quantify the increase in performance the DSP extensions offer to the LEON3 core. Afterwards, the performance of the ERMES SoC regarding the implementation of DSP algorithms has been measured on the FPGA prototype, without and with use of the VAU. Finally, delay and packet loss measurements (FXS to Ethernet port) have been carried out to ensure the proper operation of the ERMES SoC as a whole (DSP functionality, drivers and the Asterisk PBX application).

9.1. Algorithmic performance

G.711: The performance was evaluated against the ITU-T G.711 recommendation. All valid input words (16 bits yield 65536 different input words) were encoded and the respective encoder outputs were validated. The same check was carried out for the decoder. Both the encoder and decoder functions operate as recommended.
G.729A: The performance was evaluated against the ITU-T G.729A recommendation using the ITU-T test vectors for the encoder and the decoder. In all test cases, as defined by ITU-T, both the encoder and decoder functions operate as recommended.
Signaling Tone Detector: The performance was evaluated against the ETSI ES 201 235-3 v.1.2.1, ITU-T Q-24, EIA/TIA-464A and Bellcore GR-181-CORE standards, meeting all applicable requirements.
Signaling Tone Generator: The performance of the DTMF digit detection algorithm was evaluated against the ITU-T Q.23, ITU-T Q.35/E.180, ITU-T E.182, Bellcore GR-181-CORE and Bellcore TR-NWT-000506, meeting all applicable requirements.

9.2. VAU performance on the simulator

The implemented algorithms were executed within the HDL simulator and the number of CPU cycles and instructions required for their execution was measured. The VAU was employed to offload the LEON3 core from DSP tasks. The reported cycles for G.729A correspond to the processing of a 80-sample voice frame (the algorithm is invoked every 10 ms), while all other algorithms process 40-sample voice frames (invoked every 5 ms). The results are presented in Table 3 and are directly comparable to those in Table 1.

All G.711 to linear format conversions (lines 1-4 in the table below) do not employ the VAU, since they involve only table lookups. This is why the numbers in Tables 1 and 3 are exactly the same. Comparing Table 3 with Table 1, it is evident that the VAU greatly improves the performance of the LEON3 core when it implements DSP algorithms, which shall also become apparent in the following paragraph.

Table 3. LEON3 DSP cycles when executing the ERMES DSP algorithms with the VAU

Algorithm	DSP cycles
Linear to G.711 A-law encoding (5 ms frame)	6508
Linear to G.711 u-law encoding (5 ms frame)	6158
G.711 A-law to linear decoding (5 ms frame)	1788
G.711 u-law to linear decoding (5 ms frame)	1499
Linear to G.729A encoding (10 ms frame)	640439
G.729A to linear decoding (10 ms frame)	211560
Signaling tone (DTMF) detector (5 ms frame)	38335
Signaling tone generator (5 ms frame)	4659

9.3. DSP performance on the FPGA prototype

The measurements presented in sections 3 and 9.2 were also carried out on the FPGA prototype, the operating frequency of which was equal to 40 MHz (corresponding to a cycle period of 25 ns). The comparative measurements (with and without the VAU) are listed in Table 4. They demonstrate a reduction of 60% to 75% in the execution time through the use of the VAU peripheral unit. No measurements are provided for G.711 to linear conversions for the reasons already presented in section 9.2.

Table 4. Comparison of DSP algorithm execution on the FPGA prototype with and without the VAU

Algorithm	Execution time DSP only	Execution time DSP + VAU
Linear to G.729A encoding (10 ms frame)	44.5ms	16.01ms
G.729A to linear decoding (10 ms frame)	14.7ms	5.29ms
Signaling tone (DTMF) detector (5 ms frame)	3.78ms	0.94ms
Signaling tone generator (5 ms frame)	0.36ms	0.11ms

9.4. Delay and packet loss performance

The main factors determining the end-to-end Quality of Service in VoIP telephony are the delay and packet loss, to which, the operation of the endpoint contributes to a large extent. In order to measure the overall delay (Tx and Rx) and packet loss that the ERMES system introduces to a voice stream, we set up a trial environment. A PC initiates a VoIP call to the ERMES SoC and transmits a sequence of RTP packets, which contain a G.711-encoded reference signal with white noise characteristics. The ERMES system receives and processes this sequence and then sends the digital signal to the FXS codec. The FXS codec has been configured in digital loopback mode so that the digital signal is fed back to the SoC, which processes and transmits the signal back to the network. A logging PC, residing on the same LAN, uses the Ethereal tool to capture the RTP streams from both directions. The contents of the RTP packets are extracted and the two signal sequences are recomposed. Then, they are aligned in time and their mathematical cross-correlation is calculated. Since white noise is aperiodic, it has very good auto-correlation characteristics and consequently good cross-correlation characteristics with a lagged version of itself, so the delay can be accurately measured within the duration of a sample (125 μs).

The averaging of several such measurements revealed a combined delay (Tx plus Rx) of 40 to 45 ms, which is in accordance to [7] for a Class A endpoint. The packet loss was less than 0.05%.

10. Technology evaluation

The results presented in the previous section indicate that the ERMES SoC operates as required, though about 80 MHz are necessary for the implementation of a voice channel. This in turn means that an ASIC implementation, achieving a 200 MHz operating frequency shall be able to fulfill the requirement for concurrent operation of 2 voice channels with considerable headroom. [4] presents cycle measurements on the G.729A voice encoder which are remarkably close to the ones presented in this paper, since the deviation is less than 2%. The necessary processing power is, of course, about 4-8 times higher than that achieved by commercial implementations of the G.729A codec on custom DSP chips (see [8] and [9]), but if one calculates the non-recurring costs involved in the acquisition of a DSP processor core, this advantage is quickly negated for small channel densities.

Preliminary investigations also indicate that the LEON3 core used as the DSP is still the processing bottleneck. So, more LEON3 cores can be used to increase the channel density and this is where the main advantage of the VAU peripheral unit comes into the picture, since it can be accessed by multiple processors over the AMBA bus. Future work includes the development, verification and implementation of the ERMES SoC with multiple DSP LEON3 cores, so as to explore the limits of the VAU in terms of performance. A 2 DSP core - 1 VAU combination implementation on a 200 MHz ASIC seems capable of achieving the 4 channel target.

In conclusion, it is feasible to implement a 2-4 channel IP-PBX/VoIP gateway on a SoC based purely on both software and hardware provided by the open-source community, reducing both upfront and final product costs thus allowing new players into the market.

Acknowledgment

The described work has been partially supported by the Greek Secretariat of Research and Development of the Ministry of Development under the industrial research contract No. 04BEN34.

References

[1] ETSI standard ES 282 007, V2.0.0, 2008-03

[2] GRLIB IP Library User's Manual, Version 1.0.15 www.gaisler.com

[3] "G.729E Algorithm Optimization for ARM926EJ-S Processor", Technical Report CECS-03-09, Anshuman Tripathi, Shireesh Verma, Daniel D. Gajski, Center for Embedded Computer Systems, University of California, Irvine, 21/03/2003

[4] "Configurable scalar and vector coprocessors for accelerating the G.723.1 and G.729A speech coders", S. R. Parr, K. Koutsomyti, V. A. Chouliaras, J.L. Nunez, D. J. Mulvaney, Proceedings of the IASTED International Conference on Signal and Image Processing (ACIT-SIP), Novosibirsk, Russia, June 20-24, 2005

[5] AMBA Specification (Rev 2.0), www.arm.com

[6] The official Asterisk site, www.asterisk.org

[7] ITU-T P.1010 Recommendation, "Fundamental voice transmission objectives for VoIP terminals and gateways", 07/2004

[8] Application note AN2151, "ITU-T G.729A Implementation on the StarCore™ SC140/SC1400 Cores", Freescale Semiconductor 1/2005

[9] "ITU G.729A/G.729A+B Speech Coder" datasheet, The SpiritDSP company, 2008

2009 22nd International Conference on VLSI Design

Efficient Implementation of Floating-Point Reciprocator on FPGA

Manish Kumar Jaiswal
M.S.(by Research)
Department of Electrical Engineering,
IIT-Madras,Chennai-36, India.
e-mail: ee06s024@smail.iitm.ac.in

Nitin Chandrachoodan
Assistant Professor,
Department of Electrical Engineering,
IIT-Madras,Chennai-36, India.
e-mail: nitin@ee.iitm.ac.in

***Abstract*—In this paper we have presented an efficient FPGA implementation of a reciprocator for both IEEE single-precision and double-precision floating point numbers. The method is based on the use of look-up tables and partial block multipliers. Compared with previously reported work, the modules occupy less area with a higher performance and less latency. The designs trade off either 1 unit in last-place (ulp) or 2 ulp of accuracy (for double or single precision respectively), without rounding, to obtain a better implementation. Rounding can also be added to the design to restore some accuracy at a slight cost in area.**

***Index Terms*—Floating-point arithmetic, reciprocator, FPGA, double-precision, partial block-multipliers, binomial expansion**

I. Introduction

Floating point arithmetic is widely used in many scientific and signal processing applications. The greater dynamic range and lack of need to scale the numbers makes development of algorithms much easier. However, implementing arithmetic operations for floating point numbers in hardware is very challenging. Among the operations (add, subtract, multiply, divide), division is generally the most difficult to implement in hardware. Division is a fairly common operation in many scientific and signal processing applications, so there is a need for efficient hardware implementations for division.

The IEEE standard for floating point (IEEE-754) defines the format of the numbers, and also specifies various rounding modes that determine the accuracy of the result. For many signal processing, and graphics applications, it is acceptable to trade off some accuracy [1] (in the least significant bit positions) for faster and better implementations.

A lot of work has been done on obtaining efficient implementations for this operation. Generally, this operation can be done into two parts, first take the inverse of divisor and then multiply with dividend. Because of this, many hardware dividers focus on efficiently obtaining the reciprocal of floating-point number. Different proposed architectures in the literature are based on Newton-Raphson method [3], [5], [7], [10], [11], digit-recurrence method [3], [8], [11], [15], seed-architecture [12], etc. Previous works had used huge look-up tables, along-with wider multipliers, which affects the area and performance.

Our approach also focuses on finding the reciprocal. It is based on the well known binomial-expansion, contains small look-up table, and uses partial block-multipliers, resulting in less area, less delay, and correct up to required level (accuracy trade off). We have restricted ourselves only to normalized numbers. All the exceptional cases are detected, and indicated as invalid input/output. Comparisons of our implementation with previous works mentioned in the literature show that we are able to obtain small look-up tables and overall very efficient hardware.

We have used Xilinx ISE-8.2 synthesis tool, ModelSim SE 6.1b simulation tool, and X2VP30-7ff896 as our FPGA platform.

II. Approach

The format of a floating-point number is as follows:
For Single Precision

$$\overbrace{Sign-bit}^{1-bit}\overbrace{exponent}^{8-bits}\overbrace{mantissa}^{23-bits}$$

For Double Precision

$$\overbrace{Sign-bit}^{1-bit}\overbrace{exponent}^{11-bits}\overbrace{mantissa}^{52-bits}$$

In this paper, we do not discuss the exponent manipulation as it is a standard process. The benefits of our implementation are in the computation of the inverse of the mantissa.

Let y be the inverse of the mantissa a. Then,

$$y = \frac{1}{1.a}, where\ in\ 1.a,\ 1\ is\ hidden\ bit\ of\ mantissa.$$

We have divided the mantissa in two parts, a_1 and a_2. a_1 is used to fetch some pre-calculated data from a look-up table.

Now,since

$$\begin{aligned} y &= \frac{1}{a_1 + a_2} \\ &= (a_1 + a_2)^{-1} \\ &= a_1^{-1} - a_1^{-2}.a_2 + a_1^{-3}.a_2^2 - a_1^{-4}.a_2^3 + \cdots \end{aligned} \quad (1)$$

The content of each term of equation(1) will be as follows:

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.12

$$
\begin{aligned}
a_1^{-1} &= 0.\ \overbrace{xxxxxxxx}^{full\ significant\ bits} \\
a_1^{-2}.a_2 &= 0.\ \overbrace{00\cdots00}^{m-zero\ bits}\ \overbrace{xx\cdots xx}^{significant\ bits} \\
a_1^{-3}.a_2^2 &= 0.\ \overbrace{00\cdots00}^{2m-zero\ bits}\ \overbrace{xx\cdots xx}^{significant\ bits} \\
a_1^{-4}.a_2^3 &= 0.\ \overbrace{00\cdots00}^{3m-zero\ bits}\ \overbrace{xx\cdots xx}^{significant\ bits} \\
&\cdots and\ so\ on \\
. &\quad where\ m\ is\ the\ number\ of\ bits\ of\ a_1.
\end{aligned}
$$

We can see that as we move towards higher terms their contribution to main result are decreasing. Thus, depending upon our precision choice we can take suitable number of terms from equation(1) for calculating inverse, based on value of m.

For our implementation, based on experiments over a large number of random test cases, we have chosen the number of terms as described below. In case of single-precision we have taken the first three terms, while for the case of double-precision 7 terms have been taken. The value of m we have chosen is 8 for both cases. These values were selected based on available FPGA resources, as will be shown soon. We have simplified the desired terms in such a way so that we can use less hardware with low latency and good accuracy.

For single-precision we have taken all the three terms as available, like

$$y = a_1^{-1} - a_1^{-2}.a_2 + a_1^{-3}.a_2^2 \quad (2)$$

For double precision, simplified form will be as,

$$
\begin{aligned}
y = a_1^{-1} - a_1^{-1}[&(a_1^{-1}.a_2 - a_1^{-2}.a_2^2) \\
&(1 + a_1^{-2}.a_2^2 + a_1^{-4}.a_2^4)]
\end{aligned} \quad (3)
$$

Though we can simplify above equations a little more, it will affect the area, latency and accuracy. The accuracy is affected due to the fact that floating-point operations are not completely associative, i.e. u(v + w) may not be exactly equal to (uv + uw). This is due to the finite number of bits used to represent the numbers.

III. Implementation

We have shown the implementations for single precision and double precision separately as different issues arise in each case.

Some of the design decisions are based on the fact that multipliers of size 18×18 are readily available as hard IP cores on many common FPGA families. We have based our computations on the Xilinx Virtex II platform. However, the basic ideas of saving some of the block multiplications hold even if a different sized multiplier core is used, although the exact numbers would change.

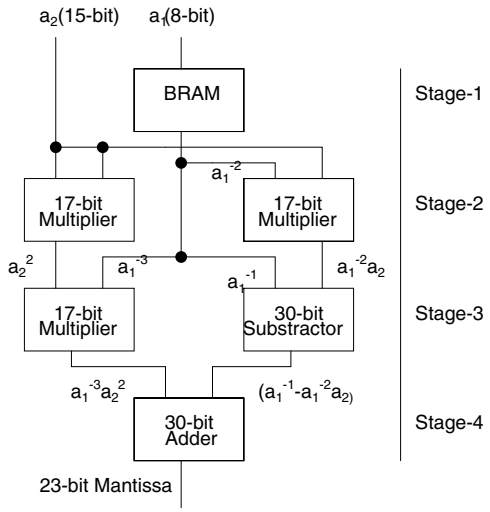


Fig. 1. Architecture for single-precision floating-point reciprocator

A. Single-precision Floating-point

The architecture of single-precision floating-point reciprocator is shown in Fig. 1. It includes a Block-Memory (BRAM) which contains pre-calculated values of $a_1^{-1}(24 - bits),\ a_1^{-2}(17 - bits),\ and\ a_1^{-3}(17 - bits)$ in a single data-word(58-bits), with 8-bit (content of a_1) as address bits. The contents of the BRAM have been calculated using a separate program written in C, with float data type for the numbers. The content of a_1^{-1} has been extended to 30-bits (by appending 6-bits "111111" at least significant bit (LSB's)) for addition/subtraction purpose. Here we can also do above operation with only value of a_1^{-1}, but it will increase the total operation latency and size of multipliers. In both cases we will use only a single BRAM on FPGA, so we prefer the first approach.

The architecture has latency of four, though we can include the BRAM access in the first stage with a slight loss in maximum operating frequency. By using pipelined multiplier we can approximately double the overall frequency. We have shown the result with the latency four. Our aim here is to only show the use of less necessary hardware. We can do pipelining in the given architecture very easily.

B. Double-precision Floating-point

The architecture of double-precision floating-point reciprocator is shown in Fig. 2. It also includes a single BRAM which contains pre-calculated values of only $a_1^{-1}(54 - bits)$ with 8-bit (content of a_1) as address bits. The content of BRAM has been calculated using a C-program, with double as data-type of floating-point numbers. The content of a_1^{-1} has been extended to 60-bits (by appending 6-bits "111111" at LSB's) for addition/subtraction purpose. Here we have a huge saving on block-memory compared to other methods discussed later.

There are three type of multiplier (based on Xilinx MULT18x18 block) that have been used. Second, third and

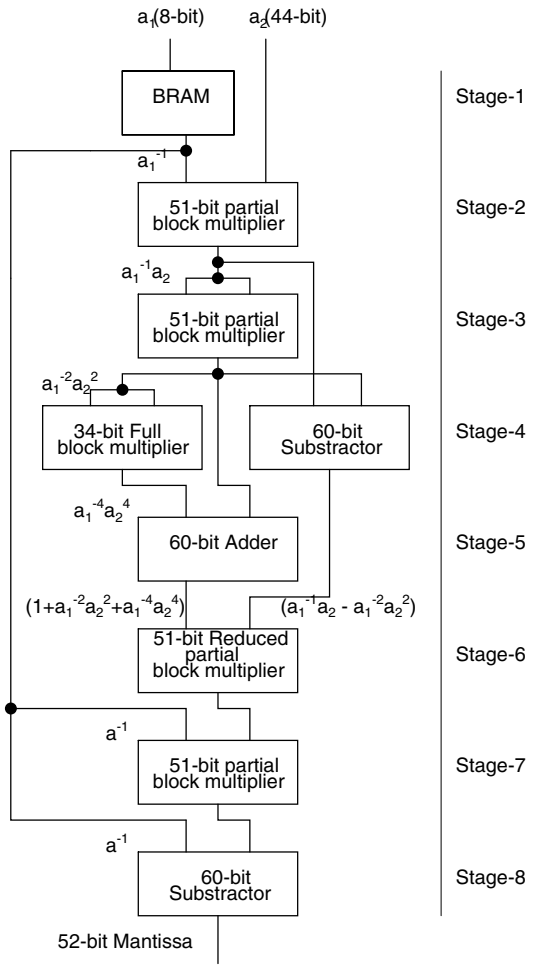


Fig. 2. Architecture for double-precision floating-point reciprocator

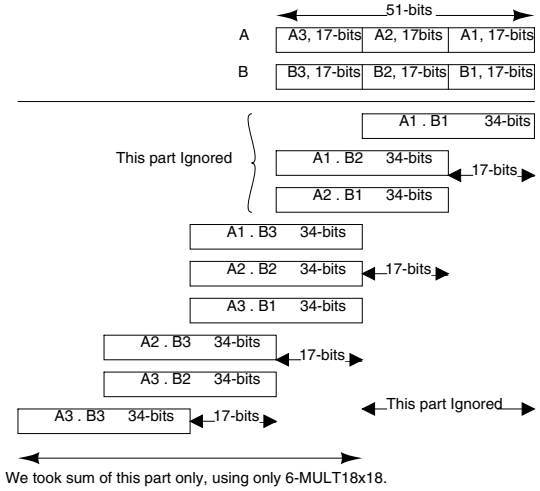


Fig. 3. Partial 51-bit multiplier for stages 2,3 and 7

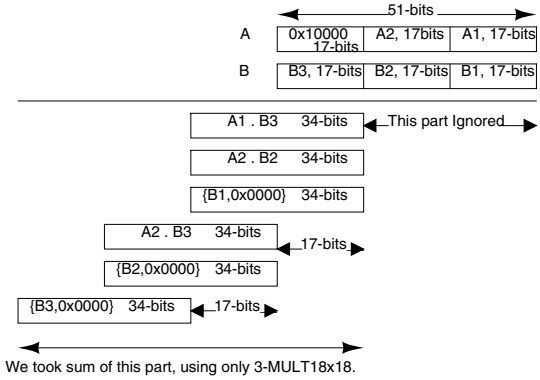


Fig. 4. Reduced Partial 51-bit multiplier for stage 6

seventh stage has 51-bit partial multiplier, which is shown in Fig. 3. It uses only six-MULT18x18 block instead of nine, to produce more than 52-bit (MSB) of correct result, which is all that we need. Stage six is also a 51-bit partial multiplier, but due to it's specific input nature (17-bits of first input is 0x10000 in hex), it contains only three-MULT18x18 block Fig. 4. The fourth stage multiplier is a 34-bit full multiplier, but instead of using IP-core for it we have designed it using four MULT18x18 block (shown in Fig. 5) which is taking less (about 2/3) glue logic and is faster than the IP-core available from Xilinx. Overall latency of module is eight, which we can increase further using pipelining as discussed in the case of single precision, for better performance.

IV. Results

Hardware utilization and performance of both the single-precision and double-precision is shown in Table-I. Since our implementation neglects some of the lower order bits in the computation, it is important to estimate the impact of this on the overall accuracy of results. For the error performance 5-millions randomly generated test cases were used to check the errors. The error performance is shown in Table-II for both versions of floating-point numbers. The error was obtained by comparing results from the proposed module with the results produced by a C compiler on a workstation. In all cases, it was found that the maximum error in the case of single precision was 2 ulp (unit last place), while in the case of double precision numbers, it was 1 ulp. The error we got is without rounding.

TABLE I
Hardware Utilization and Performance Table

Parameters	Single-precision	Double-precision
MULT18x18	3	25
BRAM	1	1
Slices	108	672
Freq(MHz)	192.365	88.594
Latency	4	8

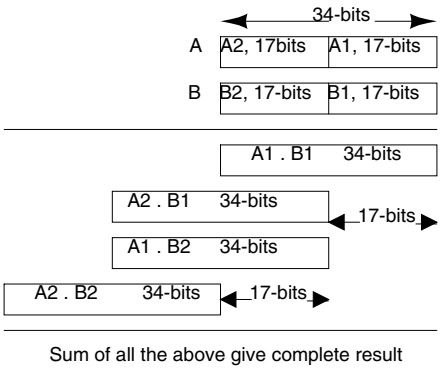


Fig. 5. 34-bit block multiplier for stage 4

TABLE II
ERROR PERFORMANCE

Error	Single-precision	Double-precision
Max. ULP	2	1
Mean	4.7867e-08	7.7636e-17
Mean(absolute)	5.0060e-08 ($2^{-24.2518}$)	7.8125e-17 ($2^{-53.5070}$)
Variance	2.5959e-15	7.5177e-33
Variance(absolute)	2.3812e-15	7.4415e-33

V. COMPARISON

The basis of our implementation is a well known technique of using look-up tables and multipliers. The main benefits are in the optimizations of the resource usage. In this section, we compare our implementation against many previous approaches mentioned in the literature.

Our comparisons are based around the Xilinx hardware resources. Even on this platform, many different multiplier implementations are available with differing speed-area-latency tradeoffs. By using different instances, we can obtain suitable tradeoffs. Similarly, on a different platform with different basic resources, the main ideas developed in this paper will still hold. Only the details of the hardware usage will differ.

For many of the comparisons, direct FPGA implementations of the methods are not available. In such cases, we have estimated the resource usage based on the components in the design. The number of block RAM cores required to implement a given look-up table and number of MULT18x18 block required to implement a given multiplication has been estimated from the Xilinx core generator software.

One of the most popular methods used for computing reciprocals is the Newton Raphson iterative procedure [3], [5], [7], [10], [11]. The Newton-Raphson iteration for reciprocal of A is given by, $x_{i+1} = x_i(2 - x_i.A)$. For each iteration it requires two multiplication and one subtraction. The value of x_0 is usually taken from a look-up table. Thus for two iterations (results are based on [7]), in the case of single-precision it requires one look-up table in 8-bit address space, two 8×16 multiplication and two 16×32 multiplication (equivalently 1 BRAM and 6 MULT18x18). For double-precision it requires one look-up table in 15-bit address space, two 15×30 multiplication and two 30×60 multiplication (equivalently 28 BRAM and 20 MULT18x18). The error performance of 2-NR method is discussed in [3], which presents the division of double-precision floating-point number by combining Newton-Raphson method and digit-by-digit recurrence method with a link module. Thus as a whole it will take more area than 2-NR method. They have shown the error produced by 2-NR iteration for computing reciprocal is minimum of 1.999999993e-55 and maximum of 1.284729483e-49, which is more than our method.

Ito *et al.* [5] implement the reciprocal computation using multiply-accumulate unit (similar to the NR method). They mention a linear initial approximation and proposed an accelerated convergence method. It results in a speedup with respect to conventional NR, but requires an additional look-up table.

Hung *et al.*[6] proposed the computation of division based on the expression $\frac{X}{Y} = \frac{X(Y_h - Y_l)}{Y_h^2}$ where X and Y are 2m-bits mantissa. Y_h is (m+1)-bits MSB of Y, used as address for look-up table for Y_h^2 of (2m+2)-bits. Thus computation of division is done by first 2m bit $Z = X \times (Y_h - Y_l)$ multiplication and then (2m+2)-bits $Z \times Y_h^2$ multiplication. For our comparison we are not including the final multiplication. Even then, for Single precision it needs $2^{13} \times 26 - bits$ (12 BRAM) look-up table and 4 MULT18x18. For double-precision $2^{27}\times56-bits$ look-up table (impractical on available FPGA platforms) and 16-MULT18x18.

Ercegovac *et al.*[7] propose a method in which reciprocal of Y (m-bit) has been computed in three steps, namely reduction, evaluation,and post-processing. They are taking 7-MULT18x18 and 1-BRAM for single precision, and 10-MULT18x18 and 30-BRAM for double-precision. In [12], the authors propose a method for computing the initial seed approximation. However, they require look-up tables addressed by the complete word, making it difficult to use for the 23-bit and 52-bit mantissas in floating point.

The methods described in [4] and [2] both require relatively large look-up tables and overall more resources than our implementation. In [9], Jeong *et al.* propose an idea that is based on [6]. Though the area is less than in [6], it is still larger than our proposed method. In [10] division operation is based on method in [5]. It first take a initial approximation and then using NR-iteration compute the reciprocal and then division. Approximately it will also take same hardware resources as in [5].

In [8], the authors have reported the floating-point division and square-root using SRT [1] division method on FPGA. In terms of performance, the pipelined approach is closest to our proposed implementation, but requires significantly more area (3245 slices and 14 BRAM for clock period of 6ns and latency of 47 cycles). [15] presents another SRT based implementation that has similar area to ours but considerably less throughput and speed.

Wang *et al.* [13] have presented a library for floating-point operations. For division this library has used the method of [6].

[1]Sweeney, Robertson and Tocher - inventors of the algorithm

Thus for Single precision it needs $2^{13} \times 26 - bits$ (12 BRAM) look-up table and 4 MULT18x18. For double-precision $2^{27} \times 56 - bits$ (impractical in available FPGA platforms) look-up table and 16-MULT18x18. Also for single-precision in spite of a relatively large latency (14 cycles, while our method only has 4) the maximum frequency is 129 MHz (we have 192 MHz).

In [16] division of double precision floating point number has been performed using Goldschmidt's algorithm, implemented on a ALTERA STRATIX-II FPGA platform. The area reported is large relative to our design (about 3500 ALMs, equivalent to about 4600 slices on a Virtex II [17]), and has less performance and throughput.

In terms of performance, the floating point library from Sandia Labs [14] and the cores from Xilinx [18] are among the best. The Sandia implementations reported here obtain high frequency of operation at the cost of increased latency (33-cycle for single-precision and 62-cycle for double-precision), while the reported areas (BRAM sizes are not mentioned in the paper) are similar to the area for our implementation. These designs are hand-optimized and are specific to the Xilinx platform, whereas we have used an HDL implementation that is easy to re-target.

Table III presents a direct resource comparison across some of the reported implementations. Since many of the implementations do not give accurate numbers for RAM usage, it is difficult to form a proper comparison. However, from this table and the above explanation it is clear that our implementation is very efficient in terms of resources. As can be noted from the operating frequencies mentioned earlier, it is clear that the proposed implementation also maintains high performance with less latency.

TABLE III
RESOURCE COMPARISON

Method	Single-precision		Double-precision	
	MULT18x18	BRAM	MULT18x18	BRAM
2-NR	6	1	20	28
[2]	14	2	36	2
[5][10]	12	1	48	29
[6][13]	4	12	16	impractical
[7]	7	1	10	30
[9]	8	1	32	50
[12]	-	impractical	-	impractical
Proposed Method	3	1	25	1

VI. CONCLUSION

We have implemented an efficient reciprocal unit on FPGA for both single and double precision floating-point numbers. The method uses the idea of neglecting higher order terms in the partial block multiplication to reduce the number of multipliers. At the same time, the look-up table requirements are kept to a minimum, and are the least reported in the literature for double precision implementation. Initial latency for our module is also less (4 for single and 8 for double-precision), that too with promising frequency, which we can improve by pipelining them very easily. The error performance is also within acceptable range (1-ulp for double-precision).

The implementation can thus form a useful core for use in hardware dividers, especially for applications like signal processing that could be more tolerant of inaccuracies in the least significant bits.

REFERENCES

[1] J. Hopf, "A parameterizable HandelC divider generator for FPGAs with embedded hardware multipliers", *IEEE International Conference on Field-Programmable Technology*, Pages 355-358, Dec-2004.

[2] W. F. Wong, Member, IEEE, and E. Goto, "Fast Hardware-Based Algorithms for Elementary Function Computations Using Rectangular Multipliers", *IEEE Transactions on Computers*, Issue 3, VOL. 43, March-1994.

[3] P. Montuschi, L. Ciminiera, A. Giustina, "Division unit with Newton-Raphson approximation and digit-by-digit refinement of the quotient", *IEE Proceedings - Computers and Digital Techniques*, Issue 6, Vol. 141, Pages 317 - 324, Nov-1994.

[4] W. F. Wong, Member, IEEE, and E. Goto, "Fast Evaluation of the Elementary Functions in Single Precision", *IEEE Transactions on Computers*, Issue 3, Vol. 44, Pages 453-457, March-1995.

[5] M. Ito, N. Takagi, and S. Yajima, "Efficient Initial Approximation and Fast Converging Methods for Division and Square Root", *Proceedings of the 12th Symposium on Computer Arithmetic*, Pages 2-9, July-1995.

[6] P. Hung, H. Fahmy, O. Mencer, M. J. Flynn, "Fast division algorithm with a small look-up table", *33th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA., Vol-2, Pages 1465-1468, Oct-1999.

[7] Milos D. Ercegovac, Tomas Lang, Jean-Michel Muller, Arnaud Tisserand, "Reciprocation, Square Root, Inverse Square Root, and Some Elementary Functions Using Small Multipliers", *IEEE Transactions on Computers*, Issue 7, VOL. 49, July-2000.

[8] Xiaojun Wang, B. E. Nelson, "Tradeoffs of designing floating-point division and square root on Virtex FPGAs", *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2003)*, Pages 195- 203, Apr-2003.

[9] Jong-Chul Jeong, Woo-Chan Park, Woong Jeong, Tack-Don Han, Moon-Key Lee, "A cost-effective pipelined divider with a small look-up table", *IEEE Transactions on Computers*, Issue-4, Vol-53, Pages 489- 495, April-2004.

[10] U. Kucukkabak, A. Akkas, "A Combined Interval and Floating-Point Reciprocal Unit", *Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*, pages 1366- 1371, Nov-2005.

[11] E. Antelo, T. Lang, P. Montuschi, A. Nannarelli, "Low latency digit-recurrence reciprocal and square-root reciprocal algorithm and architecture", *17th IEEE Symposium on Computer Arithmetic*, Pages 147- 154, June-2005.

[12] M. Ercegovac, J. M. Muller and A. Tisserand. "Simple seed architectures for reciprocal and square root reciprocal", *39th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California, USA., Pages 1167- 1171, Oct-2005.

[13] Xiaojun Wang, Sherman Braganza, Miriam Leeser, "Advanced Components in the Variable Precision Floating-Point Library", *14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines(FCCM-06)*, Pages 249-258, April-2006.

[14] K. Scott Hemmert, Keith D. Underwood, "Open Source High Performance Floating-Point Modules", *14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '06)*, pages 349-350, April-2006.

[15] H. Bessalah, M. Anane, M. Issad, N. Anane, K. Messaoudi, "Digit recurrence divider: Optimization and verification", *International Conference on Design & Technology of Integrated Systems in Nanoscale Era*, Pages 70-75, Sept-2007.

[16] R. Goldberg, G. Even, P. M. Seidel, "An FPGA implementation of pipelined multiplicative division with IEEE Rounding", *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007)*, Pages 185-196, Apr-2007.

[17] Stratix II vs. Virtex-4 Density Comparison. [Online]. Available: http://www.altera.com/literature/wp/wpstxiixlnx.pdf

[18] Xilinx Floating-point unit v2.0. [Online]. Available: www.xilinx.com

ReConfigurable Technologies

It has been envisioned that in the future it would be possible for the designer to have the complete flexibility that software offers at the hardware speeds which will ensure reduction in cost and the product turn-around time substantially. Optimal performance needs of applications can be met if fine-grained field reconfigurations can be made possible in hardware. There are several problems and challenges which need to be addressed – these include specification of reconfigurable architectures and processors, software environments that support reconfiguration, increasing heterogeneity and complexity of the systems and SoCs and power management. It is one of the goals of this talk to stimulate a discussion on reconfigurable design by introducing some key Issues.

Dr. Mona Mathur has sixteen years of experience with embedded electronics and its various applications. Currently she is employed with ST Microelectronics and heads an ST lab set up in the premises of IIT Delhi with an objective of pursuing state of the art technologies and applications. She has a PhD and MTech in Electrical Engineering from IIT Delhi and a BTech from NSIT. She has several publications in notable international journals and has taught in several institutes.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.123

Session 4C

Embedded Systems I

2009 22nd International Conference on VLSI Design

High-speed on-chip event counters for embedded systems

Nilanjan Mukherjee, Artur Pogiel*, Janusz Rajski, and Jerzy Tyszer*

Mentor Graphics Corporation
8005 S.W. Boeckman Road
Wilsonville, OR 97070, USA

*Poznań University of Technology
ul. Polanka 3
60-965 Poznań, Poland

Abstract

The paper presents new discrete event counters that are based on ring generators – high performance linear feedback shift registers. These devices outperform earlier solutions by providing an unprecedented speed of operations. A complete data required to implement the new event counters is also provided.

1. Introduction

State-of-the-art designs routinely require some form of discrete event counting performed at speed. Observability of their internals is crucial for embedded systems debugging, testing, and monitoring. Some useful events to be observed may include bus idle and data cycles, the number of grants, the number of retries, device acquisition time, device ownership time, error detections, interruptions, anomalies in gigabit Ethernet interfaces, and many others. Typically, counters are part of an on-chip facility that is to handle occurrences and durations often comparable to switching times of single logic components. The content of a conventional binary counter is easy to interpret and use – its successive stages simply form a binary number corresponding to the number of clock cycles applied once the counter is reset and activated. Unfortunately, the speed of counting is here severely affected by a critical path that spans all stages of a sequential counter (we do not consider asynchronous devices as allowing external events to affect a counter whenever they occur often causes erroneous operations). Although [2] and [7] present binary ripple-carry counters which scale to virtually unlimited size yet increment in a constant time, this design style is quite complex, requires significant hardware real estate, and introduces unacceptable delays. Other solutions [6], including prescaled counters or counters with a next state generator, feature less regular structures and are, therefore, relatively cumbersome to set up and use in automated synthesis.

An alternative approach to fast counting assumes the use of linear feedback shift registers (LFSRs) that feature much shorter critical paths, and, preferably, implement primitive characteristic polynomials. It is a non-trivial task, however, to recover the actual number of events recorded by a given n-bit LFSR. Assuming the Galois form of an LFSR (with XOR gates interspersed between flip-flops), Clark and Weng proposed a discrete logarithm-based method [1] to compute the number of shifts (events) concealed in the LFSR state. The same authors used characteristic trinomials in order to reduce the number of XOR gates to one. Since there are no primitive trinomials of degree $8k$, $k = 1, 2, \ldots$, they resort to non-primitive trinomials having the longest periods, if needed. Although the use of characteristic trinomials is not harmful to the counting itself, it is desirable to employ LFSRs with primitive polynomials having a larger number of terms. Such devices are then amenable to resource sharing, and can serve in a variety of applications as pattern generators with significantly reduced dependencies in output sequences, response compactors with shorter transition periods, on-chip data decompressors with high encoding capabilities, and many more.

In this paper we introduce a new class of discrete event counters that rest on *ring generators* (RG). These circuits enable unprecedented speed of counting and are free of many drawbacks of earlier solutions. The proposed technique uses, in a synergistic manner, the dis-

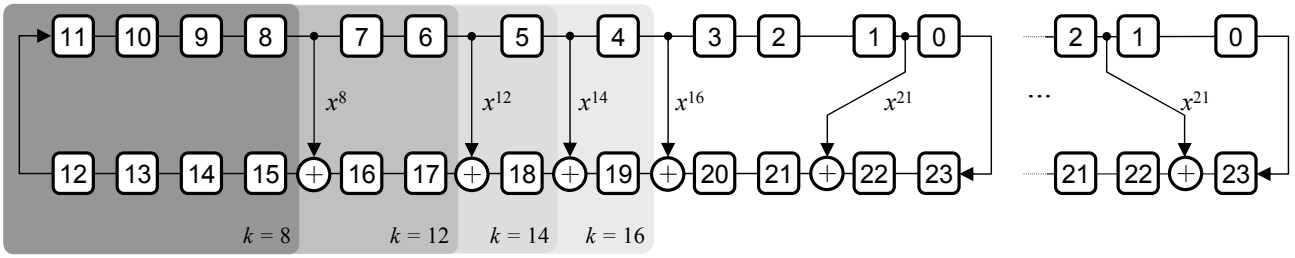


Fig. 1. Ring generator implementing polynomial $x^{24} + x^{21} + x^{16} + x^{14} + x^{12} + x^8 + 1$

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.15

crete logarithms and several other schemes, including our O(n^2) LFSR simulation algorithm. We also provide a complete data required to implement ring generator-based event counters of sizes up to 70.

2. Ring generator as event counter

A ring generator [4], [3] is a linear finite state machine obtained by applying transition function preserving transformations to canonical forms of LFSRs in such a way that the resultant circuit features:

- significantly reduced levels of XOR logic,
- minimized internal fan-outs,
- simplified circuit layout and routing.

Typically (see Fig. 1), a ring generator uses one 2-input XOR gate per a polynomial term, and each of its internal fan-outs is reduced to at most two branches. Such design style maximizes the operating speeds of the circuit and makes the whole structure highly modular.

There are several techniques of synthesizing ring generators [4]. A rule of thumb is to form a circle that consists of all memory elements, and then to add gradually feedback connections (with XOR gates) which correspond to successive terms of a characteristic polynomial. Given tap x^k, the latter step creates a feedback loop by encompassing k adjacent flip-flops, always beginning with the leftmost ones, as demonstrated in Fig. 1 for the polynomial $x^{24} + x^{21} + x^{16} + x^{14} + x^{12} + x^8 + 1$. Note that in this approach, two feedback lines cannot cross each other. Instead, they can form a very regular ladder-like structure provided a suitable characteristic polynomial is deployed. Extensive collections of primitive polynomials serving this purpose are available in [5] and [3]. It is also worth noting that both implementations of a feedback tap x^{21} (shown in the figure) are equally acceptable. Clearly, the resultant ring generators will feature different state trajectories, still producing the same m-sequence though, just differently phase-shifted in both cases.

In order to count events, a ring generator is initialized with the value of 10…0, and is subsequently shifted anytime an event occurs. Successive states visited by the ring generator will then correspond to a cumulated number of monitored events. Unfortunately, likewise other forms of LFSRs, information provided by such a counter cannot be used directly, and it needs some extra post processing. In principle, we can use the discrete logarithm-based approach of [1]. However, that method is only applicable to the Galois LFSRs. Consequently, we will present how to convert states of a given ring generator into the corresponding Galois LFSR states before a result of counting can be eventually determined. Furthermore, we will demonstrate that certain steps of the discrete logarithm-based scheme can be improved by using fast simulation of linear finite-state machines. Interestingly, applicability of this technique goes much beyond the event counting.

3. State mapping

Despite different state trajectories featured by a Galois LFSR and its derived ring generator, both circuits produce identical (though shifted) m-sequences on their outputs [4]. In particular, there are outputs where m-sequences are completely aligned. Consider a 10-bit Galois LFSR and the corresponding RG, both implementing polynomial $p(x) = x^{10} + x^8 + x^4 + x^3 + 1$, as shown in Fig. 2. Table I lists successive LFSR and RG states assuming the initial state 10…0 in both cases. Clearly, their most significant bits g_9 and r_9 yield the same sequences of bits. In fact, it can be easily verified that setting the leftmost stage of any n-bit Galois LFSR to 1 while resetting the remaining bits results, during subsequent clock cycles, in the following n-bit sequence observed on the leftmost bit: 10 … 0. The same can be obtained by setting to 1 the lower rightmost stage of the corresponding n-bit RG (with all other bits de-asserted). Hence, these stages produce the identical aligned m-sequences. Note that only the Galois LFSR performs a true polynomial division – compare the second column (a remainder) with the binary content of the Galois LFSR presented in the subsequent column.

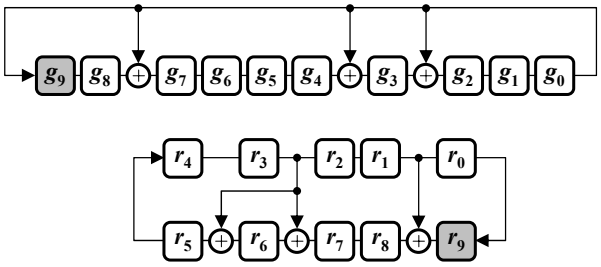


Fig. 2. Galois LFSR & the corresponding RG

In terms of functional correspondence between Galois LFSRs and ring generators, the question of interest is what, if any, mapping allows mutual conversions between states of these two devices. Such a mapping would allow one to determine a Galois LFSR state corresponding to a given RG state. As a result, once a content of the RG-based counter is known, the actual number of applied shifts (or clock cycles) could be retrieved by first converting this state into the corresponding LFSR state, and then by applying the discrete logarithm technique to finally decipher the number of recorded events.

Our solution employs some form of symbolic simulation to create linear expressions in initial variables for each flip-flop, and for n successive clock cycles. These expressions are then used to form a set of n equations associated with the two flip-flops that produce aligned m-sequences. In each equation, the expression representing a Galois LFSR is equated with the corresponding expression obtained for a ring generator. The actual mapping is then derived by using Gaussian elimination to reduce the set of equations so that variables associated

with the Galois LFSR assume the role of leading variables. As a result, given the content of a ring generator, one can easily compute the corresponding state of a Galois LFSR, as shown in the following example.

Table I. States for circuits of Fig. 2

n	$x^n \bmod p(x)$	$g_9 \ldots g_0$	$r_9 \ldots r_0$
0	1	**1**000000000	**1**000000000
1	x	**0**100000000	**0**100000000
2	x^2	**0**010000000	**0**010000000
3	x^3	**0**001000000	**0**001000000
4	x^4	**0**000100000	**0**000100000
5	x^5	**0**000010000	**0**000010000
6	x^6	**0**000001000	**0**000001000
7	x^7	**0**000000100	**0**001100100
8	x^8	**0**000000010	**0**000110010
9	x^9	**0**000000001	**0**100011001
10	$1+x^2+x^6+x^7$	**1**010001100	**1**011101100
11	$x+x^3+x^7+x^8$	**0**101000110	**0**100010110
12	$x^2+x^4+x^8+x^9$	**0**010100011	**0**110001011
13	$1+x^2+x^3+x^5+x^6+x^7+x^9$	**1**011011101	**1**110100101
…	…	…	…
1023	1	**1**000000000	**1**000000000

Example. Consider again the two registers shown in Fig. 2. Let $x_9 \ldots x_0$ and $z_9 \ldots z_0$ be initial states of the Galois LFSR and the ring generator, respectively. Linear equations over GF(2) in these 20 variables representing values of bits g_9 (left hand side) and r_9 (right hand side) in 10 successive clock cycles are as follows:

$$\begin{aligned}
x_0 &= z_0\\
x_1 &= z_1\\
x_2 &= z_2\\
x_0 + x_3 &= z_3\\
x_0 + x_1 + x_4 &= z_4\\
x_1 + x_2 + x_5 &= z_5\\
x_0 + x_2 + x_3 + x_6 &= z_3 + z_6\\
x_1 + x_3 + x_4 + x_7 &= z_3 + z_4 + z_7\\
x_2 + x_4 + x_5 + x_8 &= z_4 + z_5 + z_8\\
x_9 &= z_9
\end{aligned}$$

The above set of equations reduces to:

$$x_0 = z_0,\quad x_1 = z_1,\quad x_2 = z_2,\quad x_3 = z_3 + z_0$$
$$x_4 = z_4 + z_1 + z_0,\quad x_5 = z_5 + z_2 + z_1,$$
$$x_6 = z_6 + z_2,\quad x_7 = z_7,\quad x_8 = z_8 + z_0,\quad x_9 = z_9$$

Suppose the ring generator has reached state $r_9 \ldots r_0 =$ 0110001011 (row 12 in Table I). Using the mapping shown above, we can now compute values of successive bits of the corresponding Galois LFSR state. For instance, $g_4 = r_4 + r_1 + r_0 = 0 + 1 + 1 = 0$. Thus, $g_9 \ldots g_0 =$ 0010100011. This can be easily verified by checking row 12 of Table I again.

4. Recovering the number of events

After having determined a particular state of the Galois LFSR that corresponds to a recorded state of the RG-based counter, we can now recover the number of shifts concealed in the LFSR content. One of the simplest solutions is to use a lookup table for all possible values. Clearly, such an approach is only feasible for small LFSRs, and thus we resort to a more flexible technique based on discrete logarithms [1]. In the following, we recall the major steps of this technique and demonstrate how its efficiency can be improved.

Let $m = 2^n - 1$ be a period of an n-bit LFSR (we assume the use of primitive characteristic polynomials here). The following three pre-processing steps are required in order to arrive with some useful tables:

1. Find a prime factorization $m_1 \cdot m_2 \cdot \ldots \cdot m_k$ of m.
2. For each m_i, create a table of size m_i, initialize the LFSR with the state 100…0, and simulate it to obtain every m/m_i state which is then stored in the table.
3. For each m_i, find an integer v_i (by using, for example, the extended Euclidean algorithm) such that $v_i \cdot m/m_i \equiv 1 \bmod m_i$. Numbers v_i are required to complete the discrete logarithm-based procedure [1].

Clearly, the best candidates for counters are LFSRs with decent storage requirements, i.e., those associated with relatively small prime factors $m_1, m_2, \ldots, m_k$.

Example. Consider a Galois LFSR implementing the primitive polynomial $x^4 + x + 1$. A prime factorization of its period $m = 2^4 - 1 = 15$ is comprised of $m_1 = 3$ and $m_2 = 5$. A state trajectory of this LFSR is shown in Fig. 3 along with the corresponding tables of sizes 3 and 5 that store every $15/3 = 5^{\text{th}}$ and every $15/5 = 3^{\text{rd}}$ state, respectively. Furthermore, for the same circuit we get $v_1 = 2$ and $v_2 = 2$.

Given the state w of a Galois LFSR with characteristic polynomial $p(x)$, the number of recorded events can be determined as follows. For each factor m_i, determine $S_i = w^{k(i)} \bmod p(x)$, where $k(i) = m/m_i$, and find, in the respective table, location r_i that corresponds to S_i. The number of applied clock cycles is then equal to

$$\sum_{i=1}^{k} r_i \cdot \frac{m}{m_i} v_i \bmod m$$

Example. Suppose the LFSR of Fig. 3 has reached state 1010, i.e., $w = 1 \cdot x^0 + 0 \cdot x + 1 \cdot x^2 + 0 \cdot x^3 = x^2 + 1$. Since $m = 15$ and $m_1 = 3$, we have:

$$w^{k(1)} \bmod p(x) = (x^2 + 1)^{15/3} \bmod (x^4 + x + 1) = x^0 = 1.$$

As polynomial x^0 represents state 1000 (decimally 8), we look up this state in the respective table to learn that $r_1 = 0$. Similarly, we carry on for $m_2 = 5$:

$$w^{k(2)} \bmod p(x) = (x^2 + 1)^{15/5} \bmod (x^4 + x + 1) = x + 1.$$

Polynomial $x + 1$ corresponds to state 1100 (decimally 12), and thus $r_2 = 4$ (see Fig. 3). Finally, since $v_1 = 2$ and $v_2 = 2$, the number of clock cycles that have been applied is determined as follows:

$$(r_1 \cdot v_1 \cdot m/m_1 + r_2 \cdot v_2 \cdot m/m_2) \bmod m =$$
$$(0 \cdot 2 \cdot 15/3 + 4 \cdot 2 \cdot 15/5) \bmod 15 = 24 \bmod 15 = 9.$$

This result can be easily verified by checking the graph of Fig. 3, where state 1010 is reached in the 9th step.

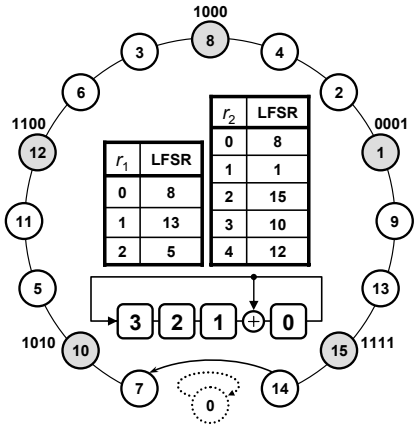


Fig. 3. Tables for polynomial $x^4 + x + 1$

Computing remainders S_i is carried out by iteratively squaring polynomial w, whereas the actual result is obtained by finding a product of selected powers of w, and then by dividing the result by $p(x)$. Both operations can be implemented using n-bit sequences representing polynomials. In particular, to arrive with certain polynomials modulo $p(x)$, we mimic a Galois LFSR which accepts successive bits of a dividend polynomial. The entire process can be illustrated as follows.

Suppose one wants to find $R = w^{58} \bmod p(x)$. In principle, polynomial w^{58} can be obtained as $w^2 \cdot w^8 \cdot w^{16} \cdot w^{32}$. The only difficulty with such an approach is that some products obtained here may be too large to work with conveniently. Fortunately, we can compute all intermittent polynomials modulo $p(x)$ too, as shown below:

$$
\begin{aligned}
&y \leftarrow w^2 \bmod p(x), && R \leftarrow y,\\
&y \leftarrow y^2 \bmod p(x), &&\\
&y \leftarrow y^2 \bmod p(x), && R \leftarrow R \times y \bmod p(x),\\
&y \leftarrow y^2 \bmod p(x), && R \leftarrow R \times y \bmod p(x),\\
&y \leftarrow y^2 \bmod p(x), && R \leftarrow R \times y \bmod p(x).
\end{aligned}
$$

Another time-critical aspect is searching tables. The number of comparisons needed to find locations of the remainders S_k in a table can take a long time, especially for counters with large factors m_k. This is ameliorated by using, for instance, hashing, at the price of a certain increase in storage requirements.

Constructing tables is a time consuming process itself as it may require almost 2^n simulation steps, where n is the size of the LFSR. A method to quickly determine k^{th} state of a given n-bit LFSR is therefore instrumental in running efficient post-processing on data obtained from LFSR-based counters. The next section presents such a method. It has ability to handle large finite state machines and arrives with desired states in at most $O(n^2)$ time.

5. Fast simulation of LFSRs

As detailed in the previous section, the discrete logarithm-based algorithm works with certain tables. They store selected states of a given n-bit LFSR altogether with the corresponding indices. These states could be collected by simply running the LFSR for up to 2^n clock cycles. This naive approach would clearly fail for large values of n. Therefore, we resort to a more subtle technique which we sketched in [5]. It allows one to determine, in a very time-efficient manner, a state that a linear finite state machine reaches after applying a given number of clock cycles.

For the sake of illustration, consider a 4-bit ring generator implementing the primitive polynomial $x^4 + x + 1$ (the procedure described here is applicable to any linear finite state machine, including canonical forms of LFSRs and cellular automata). In order to determine a state that this circuit reaches after a given number of clock cycles, we will use a lookup table that consists of four rows and four columns, as shown in Fig. 4. The entry located in ith row and jth column represents a state which the ring generator enters after 2^i steps assuming that the initial state features a single one on the jth position in its binary representation.

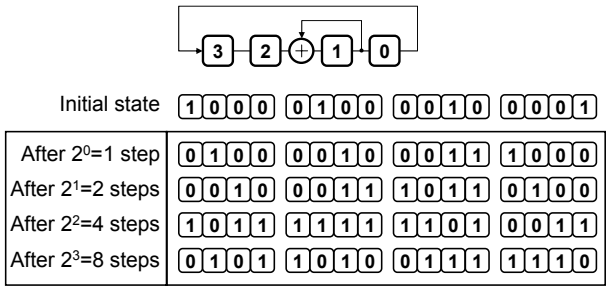


Fig. 4. Lookup table for fast simulation of RG

It is a matter of single simulation steps to determine the content of the first row of the table. Recall that it contains states reachable after applying a single clock cycle. As a result, the first row entry in column k stores the state that immediately follows a state comprised of all zeros but a single one on position k. For instance, the ring generator of Fig. 4 initialized with state 0010 moves to state 0011 in one step, and this is represented by the entry in the first row and the third column of the table. Entries in the remaining rows of the table can be determined by using the principle of superposition as follows.

Suppose one wants to determine a state that the ring generator of Fig. 4 reaches after two clock cycles assuming that its initial state was 0010. This problem reduces to finding a state that the same circuit reaches after one clock cycle provided its current state is 0011 (see the first row of the lookup table). The principle of superposition allows further decomposition of the problem into two simpler tasks: finding immediate successors of states

0010 and 0001, as these are the states whose superposition yields state 0011. From the first row of the lookup table we get that the ring generator moves in one step from states 0010 and 0001 to states 0011 and 1000, respectively. Their bit-wise sum results in state 1011. This is exactly the state that should be placed in the second row and the third column of the lookup table. Entries in the remaining rows are obtained in a similar fashion by using iteratively data computed in the previous steps. Thus, any state that a linear circuit reaches after 2^k cycles is determined by adding respective states reachable after 2^{k-1} clock cycles.

Using tables as the one shown in Fig. 4, we can find a state reachable after an arbitrary number C of cycles in at most n basic steps, each comprising up to n lookups. The computational complexity of this process is thus $O(n^2)$. It starts by expressing C as a sum of powers of 2, and then follows the rules presented above. The following example illustrates this technique.

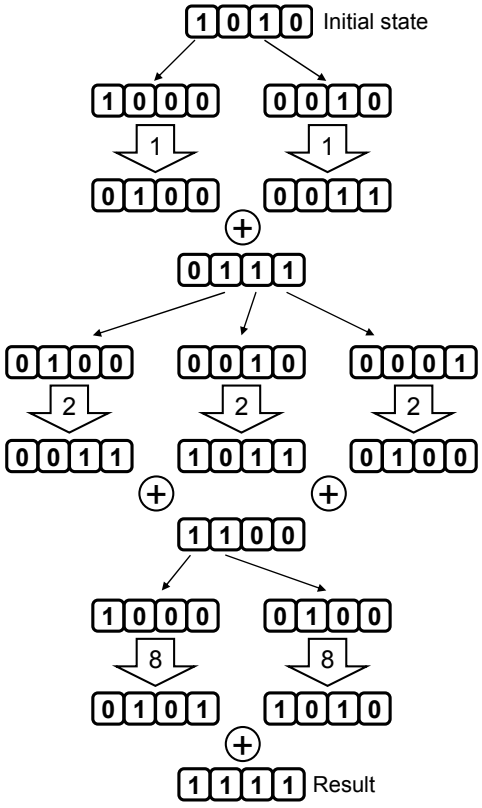


Fig. 5. RG fast simulation

Example. Let the ring generator of Fig. 4 be initially in state 1010. Suppose we seek a state that this circuit reaches after next 11 clock cycles. Since $11 = 2^0 + 2^1 + 2^3$, we proceed in three steps (see Fig. 5). An immediate successor of state 1010 can be found by determining immediate successors of states 1000 and 0010 which are, according to the lookup table of Fig. 4, states 0100 and 0011, respectively. Their sum yields state 0111. We decompose this state into three components, lookup states that can be reached from them in 2 steps, and find their sum, i.e., 1100. Eventually, the same approach is applied to state 1100, this time by retrieving states reachable in 8 clock cycles, and by computing the final state which is 1111.

6. Recommended RG-based counters

This section provides a list of primitive polynomials of degree up to 70 over GF(2) with 5, 7 and 9 terms, which are recommended for ring generator-based event counters. The polynomials are presented in Table II altogether with additional data needed to run the algorithms described in the previous sections. For example, a sequence 16 10 7 4 0 stands for the polynomial $x^{16} + x^{10} + x^7 + x^4 + 1$. Since the prime factors of the period $2^{16} - 1$ are $m_1 = 3$, $m_2 = 5$, $m_3 = 17$, and $m_4 = 257$, the corresponding coefficients v_1, v_2, v_3, and v_4 assume the values of 2, 3, 4, and 128, respectively. The last column provides the total storage (in bytes) required to maintain the respective lookup tables.

It is worth noting that some degrees n are omitted in the table due to the fact that the corresponding periods $2^n - 1$ feature prime factors of unacceptably large values, including Mersenne primes. More specifically, Table II reports only polynomials whose periods feature prime factors smaller than 1,000,000.

Table II. RG-based event counters

n	Polynomials [5]	m_i	v_i	LUT
16	16 10 7 4 0 16 13 12 9 6 3 0 16 15 13 10 8 6 4 2 0	3, 5, 17, 257	2, 3, 4, 128	564
18	18 15 9 4 0 18 14 12 9 6 3 0 18 16 13 11 9 6 4 2 0	3^3, 7, 19, 73	22, 6, 13, 47	378
20	20 13 9 5 0 20 17 13 10 6 3 0 20 17 15 12 9 7 4 2 0	3, 5^2, 11, 31, 41	1, 7, 10, 8, 9	333
21	21 17 11 5 0 21 17 13 10 6 3 0 21 18 15 12 10 7 5 2 0	7^2, 127, 337	29, 85, 249	1539
22	22 16 12 5 0 22 17 12 10 6 3 0 22 19 17 14 11 8 5 2 0	3, 23, 89, 683	2, 19, 60, 569	2394
23	23 17 11 5 0 23 19 15 11 7 3 0 23 19 15 12 10 6 4 2 0	47, 178481	15, 121519	536K
24	24 20 11 5 0 24 19 16 13 8 5 0 24 20 17 15 12 9 6 3 0	3^2, 5, 7, 13, 17, 241	1, 2, 1, 11, 14, 163	876
25	25 18 12 6 0 25 22 17 13 8 4 0 25 22 18 15 12 9 6 3 0	31, 601, 1801	25, 126, 1772	9732
26	26 17 13 6 0 26 21 17 13 9 4 0 26 23 20 16 13 9 6 3 0	3, 2731, 8191	1, 455, 4096	44K

27	27 20 13 7 0 27 24 19 14 10 5 0 27 23 20 16 12 9 6 3 0	7, 73, 262657	4, 7, 87381	1.05M
28	28 21 15 7 0 28 25 20 15 10 5 0 28 26 22 18 14 10 6 3 0	3, 5, 29, 43, 113, 127	2, 1, 1, 25, 30, 32	1280
29	29 20 14 8 0 29 24 19 14 10 5 0 29 25 21 17 14 10 6 3 0	233, 1103, 2089	101, 137, 924	13.7K
30	30 20 13 8 0 30 24 19 14 9 5 0 30 27 22 18 13 9 6 3 0	3^2, 7, 11, 31, 151, 331	8, 5, 3, 26, 143, 112	2160
32	32 25 15 7 0 32 27 21 16 10 5 0 32 28 23 20 17 12 8 4 0	3, 5, 17, 257, 65537	1, 4, 2, 64, 32768	263K
33	33 25 16 8 0 33 29 23 17 11 5 0 33 29 24 20 16 12 8 4 0	7, 23, 89, 599479	2, 5, 40, 28449	3.00M
34	34 24 15 7 0 34 27 20 16 10 5 0 34 30 26 21 16 12 8 4 0	3, 43691, 131071	2, 36409, 65536	874K
35	35 27 17 8 0 35 28 23 17 10 5 0 35 31 26 22 17 12 8 4 0	31, 71, 127, 122921	9, 1, 51, 36141	616K
36	36 25 17 8 0 36 29 24 18 12 6 0 36 31 27 22 17 13 8 4 0	3^3, 5, 7, 13, 19, 37, 73, 109	11, 3, 3, 3, 16, 1, 60, 70	1360
38	38 28 20 9 0 38 33 26 20 12 6 0 38 33 28 23 18 14 9 4 0	3, 174763, 524287	1, 29127, 262144	3.50M
39	39 28 18 9 0 39 32 26 19 13 7 0 39 34 29 24 19 14 9 5 0	7, 79, 8191, 121369	6, 50, 5461, 102343	648K
40	40 29 21 10 0 40 34 27 19 12 6 0 40 36 30 26 20 15 10 5 0	3, 5^2, 11, 17, 31, 41, 61681	2, 16, 5, 5, 4, 25, 12699	309K
42	42 31 19 10 0 42 34 28 20 14 7 0 42 36 31 26 21 16 11 5 0	3^2, 7^2, 43, 127, 337, 5419	7, 39, 31, 106, 293, 7	35.9K
44	44 31 22 11 0 44 37 29 21 14 7 0 44 38 32 27 23 17 11 5 0	3, 5, 23, 89, 397, 683, 2113	1, 2, 21, 30, 339, 626, 520	19.9K
45	45 32 22 10 0 45 37 31 23 16 8 0 45 39 33 27 22 16 11 5 0	7, 31, 73, 151, 631, 23311	1, 7, 48, 45, 12, 15310	145K
48	48 38 26 13 0 48 41 34 25 16 7 0 48 43 36 30 25 19 12 6 0	3^2, 5, 7, 13, 17, 97, 241, 257, 673	5, 1, 4, 12, 7, 63, 202, 214, 12	7.91K
50	50 39 24 12 0 50 43 34 24 16 8 0 50 44 37 30 25 18 12 6 0	3, 11, 31, 251, 601, 1801, 4051	1, 4, 28, 187, 63, 886, 235	47.2K
51	51 39 25 12 0 51 42 33 25 16 8 0 51 44 37 31 25 19 12 6 0	7, 103, 2143, 11119, 131071	5, 14, 641, 2046, 87381	1.01M
52	52 37 25 12 0 52 43 34 25 17 9 0 52 45 38 31 24 18 12 6 0	3, 5, 53, 157, 1613, 2731, 8191	2, 4, 28, 113, 729, 1593, 2048	89.3K
54	54 40 27 13 0 54 46 36 27 18 9 0 54 47 40 33 26 19 13 6 0	3^4, 7, 19, 73, 87211, 262657	76, 2, 17, 40, 58169, 175019	2.45M
55	55 41 26 13 0 55 47 38 28 18 9 0 55 47 39 31 24 18 12 6 0	23, 31, 89, 881, 3191, 201961	3, 17, 24, 857, 2991, 28564	1.44M
60	60 44 31 16 0 60 51 39 30 19 9 0 60 52 45 37 29 22 15 8 0	3^2, 5^2, 7, 11, 13, 31, 41, 61, 151, 331, 1321	4, 19, 6, 7, 7, 13, 3, 39, 147, 56, 646	16K
63	63 46 29 14 0 63 52 42 31 21 10 0 0 63 54 46 38 30 21 14 7 0	7^2, 73, 127, 337, 92737, 649657	26, 3, 113, 83, 32774, 609917	5.94M
66	66 48 33 16 0 66 55 45 33 23 11 0 66 59 52 43 34 25 16 8 0	3^2, 7, 23, 67, 89, 683, 20857, 599479	2, 1, 14, 47, 20, 645, 13180, 313964	5.59M
68	68 49 33 17 0 68 56 44 34 22 11 0 68 59 51 42 34 25 17 8 0	3, 5, 137, 953, 26317, 43691, 131071	1, 1, 75, 670, 1303, 40050, 32768	1.82M
70	70 52 35 17 0 70 58 46 35 22 11 0 70 61 52 43 35 26 17 9 0	3, 11, 31, 43, 71, 127, 281, 86171, 122921	2, 6, 20, 10, 36, 89, 76, 67632, 79531	1.89M

7. Conclusion

In this paper, we present very fast synchronous event counters. As these devices build on ring generators, they feature significantly improved structural properties and remarkably enhanced overall performance, as compared to previous schemes based on LFSRs. Advantages of the proposed technique include a single level of XOR logic, reduced internal fan-outs, and simplified circuit layout and routing. Consequently, one can synthesize highly modular counters that can operate at much higher speeds than earlier solutions.

8. References

[1] D. W. Clark and L.-J. Weng, "Maximal and near-maximal shift register sequences: efficient event counters and easy discrete logarithms," *IEEE Trans. Comput.*, vol. 43, No. 5, 1994, pp. 560-568.

[2] M. D. Ercegovac and T. Lang, "Binary counter with counting period of one half adder independent of counter size," *IEEE Trans. Circuits and Systems,* vol. 36, No 6, 1989, pp. 924-926.

[3] G. Mrugalski, N. Mukherjee, J. Rajski, and J. Tyszer, "High performance dense ring generators," *IEEE Trans. Comput.*, vol. 55, No. 1, 2006, pp. 83-87.

[4] G. Mrugalski, J. Rajski, and J. Tyszer, "Ring generators – new devices for embedded deterministic test," *IEEE Trans. CAD*, vol. 23, No. 9, 2004, pp. 1306-1320.

[5] J. Rajski and J. Tyszer, "Primitive polynomials over GF(2) of degree up to 660 with uniformly distributed coefficients," *Journal of Electronic Testing: Theory and Application,* vol. 19, 2003, pp. 645-657.

[6] M.R. Stan, A.F. Tenca, and M.D. Ercegovac, "Long and fast up/down counters," *IEEE Trans. Comput.*, vol. 47, No. 7, 1998, pp. 722-735.

[7] J.E. Vuillemin, "Constant time arbitrary length synchronous binary counters," *Proc. IEEE Symp. Computer Arithmetic*, 1991, pp. 180-183.

A Workbench for Analytical and Simulation based Design Space Exploration of Software Defined Radios

T. Kempf, S. Wallentowitz, G. Ascheid, R. Leupers, and H. Meyr
Institute for Integrated Signal Processing Systems, RWTH Aachen University, Germany
kempf@iss.rwth-aachen.de

Abstract— This paper presents a workbench addressing the issue of early design space exploration for Software Defined Radios (SDRs). Key contribution is a pre-simulation mathematical analysis based on Synchronous Data Flow (SDF) graphs, which supports system architects in their soft- and hardware design decisions at early design stages. The analysis is integrated into an Electronic System Level (ESL) based simulation framework allowing a seamless design flow from purely mathematical analysis down to the final implementation of the SDR. In a case study of an exemplary selected physical layer processing the usefullness of the workbench is highlighted.

1 Introduction

The application of multiple wireless communication standards in today's communication networks opens completely new opportunities in future, like cognitive radios and networks. Current implementations of such different communication standards on one wireless device demand designers to apply multiple dedicated hardware parts. To cope with this issue, industry and research opts for an SDR, where different radios are implemented as software allowing reuse of hardware components. However, development of such SDRs is a complex task as both soft- and hardware have to be developed under stringent real-time constraints along with energy efficiency. Those real-time constraints occur particularly within so called *feedback-loops* and *critical paths* [1]. Prominent examples of such are the automatic gain control or the channel prediction.

The SW development of today's applied wireless standards can consume a significant amount of effort. This effort is surely in the range of several man years and is expected to increase for future wireless communication standards. Therefore, this high investment in effort and cost forces system architects to validate before starting the real implementation if the addressed implementation of both soft- and hardware fulfills the given requirements.

Early performance exploration is a key technology to ensure the adherence of such requirements right from the start of the design process. To enable a performance exploration of the complete SDR system, designers have to investigate the timing characteristics of the SDR including both soft- and hardware. Unfortunately, timing characteristics of even a single task are mostly unknown or just estimated at the start of the design. Additionally, the task's implementation is yet not known or not completed. Therefore, new performance evaluation methods have to be incorporated at early design stage to allow validation of the SDR performance characteristic and to avoid taking cost intensive false decisions.

In this paper an SDR[1] workbench for early design space exploration is presented. It is based on a mathematical analysis to allow system architects to evaluate their design decisions at early design stage and supports them in their further design process. The analytical workbench is combined with an existing Electronic System Level (ESL) based workbench allowing a seamless refinement process of the implementation model down to the final SDR implementation as illustrated in Figure 1.

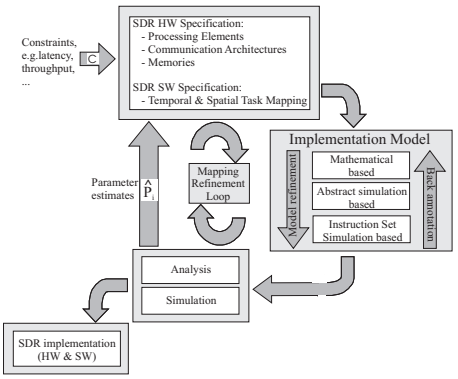


Figure 1: Iterative design process with analysis/simulation based evaluation

The structure of the workbench allows for an iterative design loop, which is as follows: The performance parameters serve as an optimization for the later design process. In the domain of wireless communication those parameters mostly relate to latency and throughput requirements. Starting with an initial guess of a suitable hardware platform and a temporal & spatial task mapping the design loop is entered. In a first design iteration an implementation model based on estimated parameters is composed. Based on this model system architects can iterate over different design decisions and evaluate their characteristics by the proposed mathematical analysis discussed in Section 3 later. With one or multiple identified implementation candidates the implementation model can be refined to the next abstraction level. Moving from the mathematic based analysis to an abstract simulation, system architects can evaluate their implementation candidates in a more fine grained manner. The abstract simulation model [2] can seamlessly re-use the mathematical models in early design steps. By a later model refinement each part of the soft- and hardware is refined till a final *Instruction Set Simulation* based implementation model exists. This can later serve as basis for lower level implementations like Register Transfer Level and below. Unfortunately, the design process is mostly not that straightforward, thus the proposed workbench offers at each point in time a back annotation of the necessary information.

This paper focuses on the mathematical implementation model and the respective implementation analysis. The link to the abstract simulation based level and finally to ESL designs and Virtual Platforms (VPs) exists by a refinement of the implementation model. The proposed refinement flow is sketched in Section 3.3.

[1]This paper focuses on SDRs, however the proposed methodology can be applied to any kind of Multi-Processor System-on-Chips (MPSoCs).

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.24

After the subsequent discussion of the related work, the mathematical analysis is introduced. Here first the overall technology is highlighted followed by a detailed discussion of the underlying technique and the link to simulation based implementation models. Finally, in Section 4 the case study illustrates the design process and the capabilities of the proposed approach on an exemplary physical layer processing.

2 Related Work

With the increasing demand for early design space exploration a wide range of performance evaluation approaches have been proposed. The urge of HW and SW design decisions even in early steps of the design process leads to the trade-off between estimation accuracy, evaluation time and existence of an executable model. The approaches to the evaluation of a single design point can be classified into two kinds: simulation based and analytical frameworks [3]. System-level simulation frameworks are available with support for mixed levels of abstraction utilizing different types of architecture and application modeling. Today most of the available Electronic System Level (ESL) simulation frameworks are based on SystemC [4], which has become in 2005 the IEEE Standard 1666^{TM}-2005. Such frameworks typically operate on cycle and instruction accurate level by the technique of Instruction Set Simulation (ISS). Recently Virtual Platforms (VPs) have originated from such frameworks. Prominent Virtual Platforms, which focus on fast simulation to allow SW development before the HW is available, are provided by Synopsys [5], CoWare [6] among many others. For design space exploration on a particular *early* design stage those frameworks suffer by nature from the following key issues:

- *HW and Compiler:* ISS based simulation requires a fixed and defined processor core as well as a compiler to generate object code from the SW, mostly developed in C programming language. At *early* design stage typically neither the instruction set e.g. of an Application Specific Instruction Set Processor (ASIP) is fixed, nor is a compiler available. Unfortunately, those are required to develop optimized SW and to evaluate the performance. Therefore, the designers have to wait till both are available, which is far too late in the design process.
- *SW:* Due to the tight constraints of baseband processing, SW has to be typically optimized for the specific architecture. Especially for performance critical parts like SW executed on DSPs and ASIPs the impact of such HW specific optimizations can be significant, e.g. in [7] a speed-up factor of up to 15 for DSPs has been measured. At *early* design stage the final SW implementation is mostly not finalized or even not started. In best cases only the actual performance of the SW can be measured, which might be far off the final one. In worst cases the simulation is just infeasible due to the lack of the SW implementation.

To cope with these issues, particular approaches increase the abstraction level to an abstract processor modeling, which allows for design space exploration without or with approximate SW available. Those abstract models such as the ones described in [8],[9], [10] and [2] work on the principle of timing annotation. Recent case studies from industry have proven the capabilities of such abstract simulation [11].

Those abstract simulation models are well suited for early design space exploration and can be used to evaluate different points of the large design space in a simple and fast manner. However, system architects have to define a first suitable HW platform and temporal & spatial task mapping by an initial guess. Additionally, simulation based approaches are in comparison to mathematical analysis of e.g. worst case execution time (WCET) comparably slower. Also those approaches depend on the simulation stimulus which makes it challenging to cover all different cases. Here analytical models can support design decisions and provide an initial guess before entering the simulation based evaluation of the system.

Contrary to simulation based approaches analytical ones can only be applied in case of deterministic or at evaluation of WCET behavior. Recent research has focused on event stream based models, like the ones proposed by Richter et. al [12] and Wandeler et. al [13]. Here the basic principle is to compute the system-level behavior on basis of event streams represented as so called arrival curves. Such analytical models can speed-up the exploration of different design points and help to find corner cases. However, those cannot replace simulation models completely as they are inaccurate in their results compared to cycle accurate simulation and cannot detect dynamic effects. Additionally, application of such analytical models suffer from the following:

- Development of analytical models with a certain degree of precision can be extremely difficult and can take significant development effort [14].
- Such analytical models lack the real implementation, therefore additionally to the time consuming process of analytical model development, the final SW has to be developed completely separated within a different environment.

Combining both analytical and simulation based approaches has mostly been performed to inspect memory and cache behavior in the past. Here trace-based performance analysis as discussed in Ref. [15] is key e.g., the frameworks introduced in [16] and [17]. Lahiri et. al [18] have extended the scope to design space exploration of the communication architectures. Other approaches utilize an analytical approach with initial calibration based on simulation results in the domain of network processing [19].

In this paper an analytical model based on Synchronous Data Flow (SDF) [20] graphs is proposed which extends a simulation based framework for SDR design space exploration. It is directly embedded into the design process to allow a seamless refinement from high analytical model down to an Electronic System Level (ESL) based implementation.

3 Analytical Workbench

In Figure 1 the overall design process including the proposed analysis is illustrated. Within the following section this analysis shall be discussed in more detail, which has been implemented using Matlab [21]. First an overview will be given, followed by a detailed discussion of the underlying technique of the analysis. Finally, the link to simulation based analysis will be sketched.

3.1 Analysis Overview

In SDRs tight constraints for *critical paths (CPs)* such as *feedback loops* exist. The proposed analysis aims to determine at an earliest possible design stage the two main issues: (i.) a suitable HW architecture including Processing Elements, Communication Architecture and Memories and (ii.) a suitable temporal & spatial task mapping.

The proposed mathematical analysis is based on the concept of a workbench, thus *no* automatic mapping and HW architecture exploration is performed. However, system architects can evaluate a certain specified design point. This allows analysis if the given constraints are likely to be accomplished or not. In such case it gives hints where modifications should be applied either to the HW platform or to the mapping. To ensure addressing critical parts first the workbench gives hints for the implementation order of tasks.

In the design space of wireless communication the assumption for applications, such as a mode of a wireless standard, is that those can

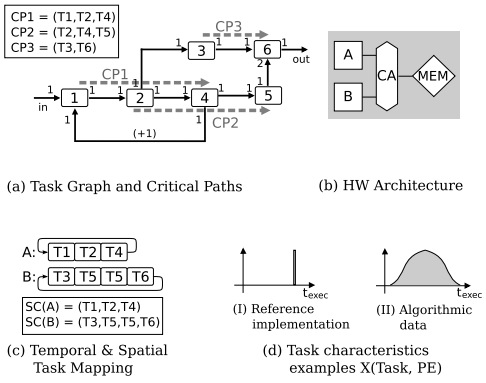


Figure 2: Exemplary analysis components

be considered as a Synchronous Data Flow (SDF) task graph. Additionally, static schedulers e.g. round-robin or other time slicing schedulers are applied. Different modes or standards can be described by multiple applications each composed out of an SDF graph.

On a simplified example in Figure 2 the analysis is highlighted here and in the subsequent section. The general application decomposed into multiple tasks (Fig. 2a) is mapped temporally & spatially (Fig. 2c) on the depicted HW architecture (Fig. 2b). For this particular case the workbench allows the evaluation of the latency and throughput of the CPs (Fig. 2a). This evaluation needs sufficiently precise characterization of the processing and communication behavior of each task on its mapped processing element (Fig. 2d). Unfortunately, those are not known or bonded with a particular uncertainty at such early design stage. Thus an iterative design loop is mandatory to validate the system at each design step with the available knowledge.

The task's execution and communication characteristic is conceived in a *random variable* $X_i = X(Task, PE)$. The uncertainty of those characteristics is tightly coupled to the designer's knowledge of the task ranging from complete knowledge of the execution, e.g. an FIR filter implementation, to merely algorithmic level where only rough estimates of the required operations exist. Respectively these characteristics can be described by a *probability density function (pdf)* given e.g. in the first case by a dirac delta function (Fig. 2d.I) whereas it might be given in second case as a gaussian-like distribution with a high variance (Fig. 2d.II). Please note that the proposed analysis operates on those estimates, such that better ones obtain results which better match the real final implementation. This implies at the start of the design phase a particular degree of uncertainty. During the design process taken measurements, for instance acquired by simulation, can provide more specific implementation knowledge which can help to remove the uncertainty.

Since the analysis is based on estimates the result can not just be given by a boolean decision, rather it is based on a likelihood whether the analyzed SDR is feasible to work or not. Therefore the result for each CP is a random variable determined by its pdf. Figure 3 depicts possible results:

- *Likely feasible (Fig. 3.a):* The highlighted random variable keeps most likely the threshold. Depending on the margin developers might consider modifications of the HW and/or task mapping since such systems tend to be overdesigned.
- *Uncertainty dominated (Fig. 3.b):* The high margin between threshold and expected value makes it likely feasible that in spite of the failure probability which might be caused by imprecise estimates the system should work. In such cases developers should focus on the implementation of tasks with a high uncertainty, respectively tasks with imprecise knowledge, first. After implementation more precise values can be determined and analysis should be re-run to verify that the addressed implementation still holds true.
- *Expected value dominated (Fig. 3.c):* Here the pdf shows only a minor variance, but only a small margin between the expected value and the threshold exists. Such systems tend to fail due to unexpected behavior. Therefore system architects should inspect deeply if the envisioned implementation has to be modified or if the system really has to work on its limits to achieve this requirement.
- *Unlikely feasible (Fig. 3.d):* Since a high probability exists that the given constraint can not be satisfied, system architects have to re-consider either the addressed HW or task mapping.

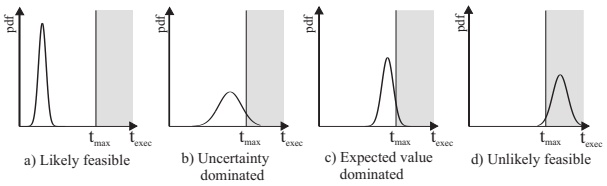


Figure 3: Exemplary analysis results for latency constraints

Before applying the given analysis workbench to a more realistic application from the wireless communication domain the underlying methodology of the analysis is discussed in the subsequent section.

3.2 Analysis Methodology

The analysis algorithm is a graph based calculation of the random variables describing the performance of the evaluated critical paths. Inputs to the algorithm are the task graph, the architecture description, and the temporal & spatial task mapping for the processing elements as depicted in Figure 2. The task graph $TG = (T, E, r, d)$ is a Synchronous Data Flow graph [20]. The TG is composed of tasks T forming the nodes and directed edges E representing the data flow. The edges have a static annotation $r : E \rightarrow \mathbb{N} \times \mathbb{N}$ of the number of tokens, that are produced and consumed by the corresponding tasks. In addition, inter-iteration data flow $d : E \rightarrow \mathbb{N}$ is visualized in the form `(+i)`. This inter-iteration annotation is of particular interest for feedback loops, e.g. adaptive filters, where a particular input influences the processing of later samples. Furthermore the critical paths $CP = \{(t_i, .., t_j) : t_i, .., t_j \in T\}$ are the evaluated constraints.

The analysis workflow partly depicted in Algorithm 1, is structured into the following three phases:

1. *Initialization phase:* Prior to the analysis phase the TG is modified into a *directed acyclic graph (DAG)*. The applied operations are mainly duplication of tasks and unrolling of the instances as necessary for the inter-iteration dependencies. For each of the PEs a *control flow graph (CFG)* is generated based on the underlying temporal mapping (Figure 2c). In a joint representation of those the overall CFG is a clustered DAG representing the temporal & spatial task mapping. To allow an analysis of the critical paths an *Analysis Graph (AG)* is formed by combining the TG and the CFG (lines 1-11). The construction of the AG is as follows: In the first step for each node a random variable X_i, describing the tasks execution characteristic, is stored as a probability density function. In the succeeding step, nodes are inserted for the data transfers, that induce a delay[2] based on the underlying communication architecture ca_c. Finally, the control flow dependency edges are inserted. Since the critical paths are related to the AG, critical paths over several instances, e.g. containing feedback loops, are enabled.

[2]For simplification in later examples in this section the communication nodes are neglected. However, the analysis takes those into account.

Algorithm 1: The analysis algorithm

Input: Modified Task Graph TG, Control Flow Graph CFG, Communication Architecture Mapping $ca(t)$, Processing Element Mapping $pe(t)$, List of evaluated Critical Paths CP
Output: Performance characteristic distributions X_{cp} for CPs

1 $AG = (V(TG), \emptyset)$;
2 **foreach** $v_i \in V(AG)$ **do**
  $X(v_i) =$ mapped_performance_characteristic$(v_i, pe(v_i))$;
3 **foreach** $(v_t, v_h) \in E(TG)$ **do**
4 **if** *transfer* $c = (v_t, v_h)$ *has communication delay* $d > 0$ **then**
5 Add new vertex v_c to $V(AG)$;
6 $X(v_{c(t,h)}) =$ mapped_performance_characteristic$(c, ca(c))$;
7 $E(AG) = E(AG) \cup \{(v_t, v_c), (v_c, v_h)\}$;
8 **else**
9 $E(AG) = E(AG) \cup (v_h, v_t)$;
10 **end**
11 $E(AG) = E(AG) \cup E(CFG)$;
12 $(AG, X) =$ complexity_reduction(AG, X);
13 $V_S = \{v \in V(AG) : |\{(v, v_i) \in E(AG), v_i \in V(AG)\}| > 1\}$;
14 $V_J = \{v \in V(AG) : |\{(v_i, v) \in E(AG), v_i \in V(AG)\}| > 1\}$;
15 $\tilde{V}_J =$ sort(V_J); /* Ordered set subject to reachability */
16 **foreach** $v_j \in \tilde{V}_J$ **do**
17 $V_{\text{predec}} = \{v \in V : \exists (v, v_j) \in E\}$;
18 $V_D = \emptyset$;
19 **foreach** $v_p \in V_{\text{predec}}$ **do**
20 Add vertex v_d in AG;
21 $E(AG) = (E(AG) \setminus (v_p, v_j)) \cup \{(v_p, v_d), (v_d, v_j)\}$;
22 **end**
23 $v_s =$ first common split on reverse paths;
24 **foreach** $v_d \in V_D$ **do**
25 $X_{(v_s,..,v_d)} =$ calculate_performance_characteristic(v_s, v_d);
26 **end**
27 **foreach** $v_d \in V_D$ **do**
28 $X_m = max(\{X_{(v_s,..,v_i)} : \forall v_i \in V_d \setminus \{v_d\}\})$;
29 $X_d = max(0, X_m - X_{P(s,i)})$;
30 **end**
31 **end**
32 **foreach** $cp = (v_s, .., v_e) \in CP$ **do**
33 $X_{cp} =$ calculate_execution_characteristic(v_s, v_e);
34 **end**

2. *Pre-Analysis phase:* In this phase short-cuts are eliminated from the AG and vertices are merged where possible for complexity reduction (line 12). Those short-cuts are defined as edges (v_t, v_h) that could also be reached through a path $(v_t, .., v_h)$. Furthermore, *dependency delay* vertices are pre-calculated (lines 13-31), that are necessary for critical paths starting in parallel paths as described subsequently.

3. *Analysis phase:* Here the critical paths and feedback loops are analyzed according to the AG (lines 32-34). The applied analysis differentiates between two patterns in critical paths $(v_s, .., v_e)$ and their evaluation, which are:

a) *Straight Paths & Parallel Execution Paths:*
For *straight paths* the execution characteristic of a critical path is in general the sum of the random variables along its path. For the previous example CP1 (Fig. 4), while neglecting the communication delays for the sake of clarity, this computes to:

$$\mathrm{X_{CP1}} = \mathrm{X_1} + \mathrm{X_2} + \mathrm{X_4}$$

In case more than one path exist for a pair (v_i, v_j) in a critical path $(v_s, .., v_i, .., v_j, .., v_e)$ this path is referred as a *parallel execution path*. Hence, the maximum of the parallel paths $p_1, .., p_n$ from v_i to v_j is used in replacement for these paths:

$$X_{CPi} = X_s + .. + max(X_{p_1}, .., X_{p_n}) + .. + X_e.$$

For example, in the critical path $CP2$ (Fig. 4) T_5 can not start execution before both paths from T_2 to T_5 finished their execution such that $CP2$ computes to:

$$\mathrm{X_{CP2}} = \mathrm{X_2} + max(\mathrm{X_4}, \mathrm{X_3}) + \mathrm{X_5} + \mathrm{X_5}.$$

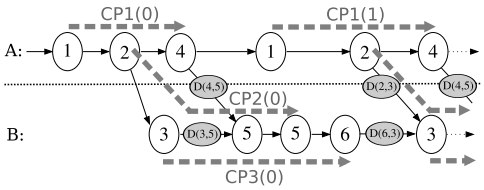


Figure 4: Exemplary AG with Critical Paths and added dependency vertices for computation of $CP3 = (T3, T6)$
(For simplification the communication nodes have not been illustrated, which are taken into account by the proposed analysis.)

b) *Critical Path starts in Parallel Paths:*
When calculating a critical path $CP_k = (v_s, .., v_e)$ starting in a subgraph of parallel paths, one has to consider that *dependency delays* $D(i, j)$ exist. Those may exist for so called *joins*: $V_J = \{v_j \in V : |\{(v_i, v_j) \in E\}| > 1\}$. If a join has incoming edges, that are not part of the subgraph, that is spanned by the paths from v_s to v_e, those dependency delays have to be considered, i.e. for joins v_j that fulfill the condition $\exists (v_i, v_j) \in E : \nexists (v_p, .., v_i), v_p \in CP_k$. Then the computation of the CP is in general again the sum of the random variables including the dependency delays along the CP:

$$X_{CPi} = X_s + .. + X_{D(i,j)} + .. + X_e.$$

The critical path $CP3$ in the previous example is such a path (Fig. 4). When assuming As-Soon-As-Possible (ASAP) scheduling a delay between the finish of T_3 and the start of T_5 induced by the execution characteristic of T_4 may occur. To consider this a *dependency delay vertex* $D(4, 5)$ is added in the incoming edge of T_5 as illustrated in Figure 4 and Algorithm 1 in lines 17-30. For the final calculation of the critical path the both preceding patterns are applied.

As discussed in Section 2 analytical models typically lack the link to simulation based approaches and the final implementation. Since the proposed design process combines analysis and simulation based design space exploration the subsequent section highlights the link between those approaches.

3.3 Simulation Link

Both analytical and simulation models have their own advantages and disadvantages. Typically those approaches are completely separated, which requires twice development effort in two different environments. To bridge this gap in SDR and MPSoC design flows the proposed analytical analysis is integrated into a simulation based framework. Figure 5 depicts the refinement flow from the mathematical down to the abstract simulation model.

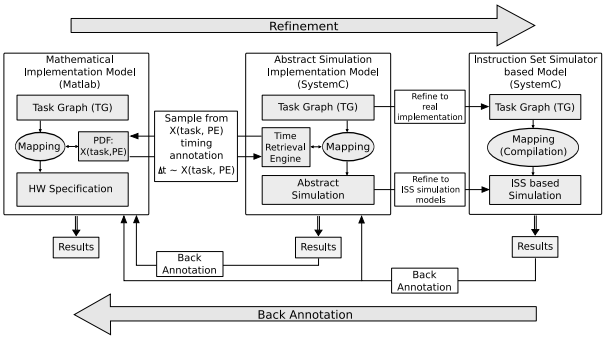
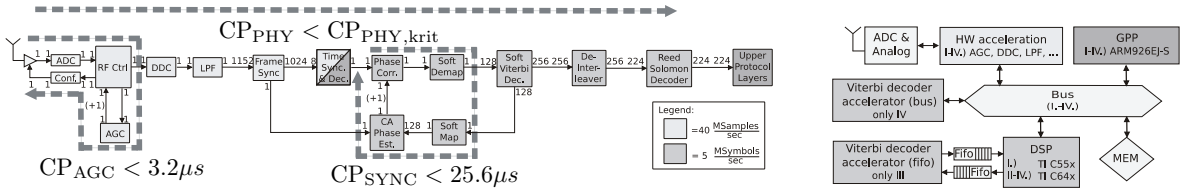


Figure 5: Refinement of the implementation model



(a) Application: Physical layer processing including iterative synchronization loop and critical paths

(b) HW architecture

Figure 6: Case study application and HW architecture with configurations: (I-II) SW centric, (III-IV) SW with HW accelerator

Accordingly at the start of the design process a mathematical implementation model is developed based on the previous discussed analytical workbench. This computes for each critical path the characteristic $\{X_{CPi}\}$ based on the individual task characteristics $X(task, PE)$. If a suitable candidate has been identified, developers can seamlessly refine the mathematical model to a SystemC simulation model based on the framework discussed in [2]. The central element in this framework is an *abstract processor simulator* operating on *timing annotation* to reflect the task's execution characteristic. The seamless refinement is supported by the *Time Retrieval Engine* which samples from the pdf $X(task, PE)$ a timing annotation Δt at run-time ($\Delta t \sim X(task, PE)$). This timing annotation value is then propagated to the abstract processor simulator which annotates this. Thus, *without any* modification the mathematical implementation model can be evaluated in a simulation based framework. Encountered effects within the simulation can be iteratively back annotated to the mathematical implementation model.

Since the final implementation is naturally the goal of such a design process, developers can refine the abstract simulation model further to an Instruction Set Simulator (ISS) based implementation model. This step comprises on the one hand implementation of the SW mostly in C programming language and on the other hand replacement of the abstract processor simulator by an ISS. Such simulation models allow measurement of the actual performance in terms of instruction or cycle accurate behavior. Again the acquired results can be back annotated to the abstract simulation and mathematical implementation model. Here, as previously discussed, uncertainty due to unknown implementation knowledge can be reduced with the actual measured performance. This enhances the estimates and provides results which are closer to the real implementation in later design iterations.

In the following section the proposed analysis is applied and the refinement step to the abstract simulation model is briefly introduced.

4 Case Study

To demonstrate the feasibility of the proposed approach a simple physical layer processing (Fig. 6) including an iterative synchronization loop from the domain of wireless communication has been selected. Such physical layer processing is today dominated by Digital Signal Processors (DSPs) and HW accelerator based designs. Upper protocol layers are mostly processed on General Purpose Processors (GPPs). For simplification the design space has been restricted to the following standard elements:

- *GPP processor cores:* ARM926EJ-S
- *DSP processor cores:* TI C55x and C64x
- *HW accelerators:* arbitrary in type and number
- *Communication architecture:* point-to-point and bus based
- *Memories:* arbitrary

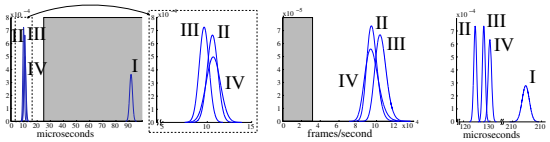
Based on those components a suitable HW architecture along with a temporal & spatial task mapping shall be determined for the application meeting the given constraints (Fig. 6a,b). Following the basic idea of SDRs would lead to a complete SW implementation of the application either on the DSP or GPP. Since the data rates close to the RF part are very high an initial calculation is performed:

- Symbols each having an I/Q value with two times 16 bit (=32bit) arrive at a data rate of $R = 40\frac{Msamples}{second}$ at the RF frontend. Respectively the sample time is $\Delta t_{sample} = 25\frac{\text{ns}}{\text{sample}}$.
- Unfortunately this time is mostly too short for a general SW implementation (5 cycles at 200 MHz clock frequency). Respectively those tasks are directly mapped onto HW accelerators. Since they have a fixed latency and throughput the critical path CP_{AGC} has been once verified and does not need further analysis in the following.

To determine suitable candidates for such an SDR application an analysis based on the proposed workbench is performed. The first investigated mapping and HW is DSP centric (Fig. 6b configuration *I*). Here the remaining physical layer processing is mapped to a TI C55x DSP (clock frequency 200MHz) and the upper protocol layers are mapped to an ARM926EJ-S GPP (clock frequency 250MHz). Based on the functionality of each task the processing and HW characteristics have been defined by a system architect. Those are based on HW and SW estimates as well as on TI's DSP Libraries [22]. The results depicted in Figure 7 highlights that configuration *I* fails to meet the required constraints. Inspecting the results more in depth reveals that the *Soft Viterbi* and *Reed-Solomon decoder* are the dominating tasks on the latency.

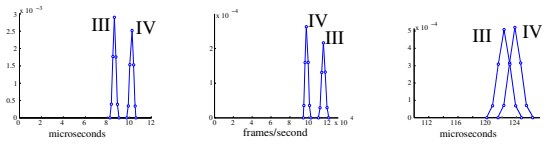
From an engineering perspective one simple solution to cope with this is to replace the unsuited processor core by a more powerful one. In configuration *II* (Fig. 6b) the C55x is replaced by a C64x DSP (clock frequency 500MHz). The results in Figure 7 highlight a significant speed-up compared to configuration *I*. The sketched constraints can be satisfied with this configuration. In typical TI SoC platforms addressing much more complex physical layer processing than the discussed one, a Viterbi Co-Processors (VCP)[23] running at the third of the processor's clock frequency is added to relieve the DSP in computing the algorithm. Evaluation of platforms including such a VCP can be simply applied by adding a VCP processing element and modifying the mapping in the underlying description stored in XML format. Please note that this rather simple physical layer processing does not need the accelerator to match the constraints. However, to demonstrate the exploration and the link to simulation of realistic platforms the impact of such a HW accelerator is evaluated.

The VCP is investigated in two configurations. In configuration *III* the VCP is directly connected to the DSP by FIFOs, whereas in *IV* it is connected to the bus like in TI's platforms. The retrieved analysis results are illustrated in Figure 7. Moving the Viterbi decoding to the VCP is a classical trade-off decision. On the one hand parallelism is added to the system since the Deinterleaver and Reed-Solomon decoding of the previous frame can be executed in parallel on the DSP. On the other hand this parallelism introduces communication overhead to the system as the data has to be transfered to the VCP and back again. Due to this overhead the frame latency increases as visible in Figure 7c. While utilizing direct FIFO communication links between the DSP and the VCP (configuration *III*) this overhead is rather low as no



(a) Feedback loop $\mathrm{CP}_{\mathrm{SYNC}}$ (III-IV) (b) Throughput (II-IV) (c) $\mathrm{CP}_{\mathrm{PHY}}$ (I-IV)

Figure 7: Mathematical analysis results for the evaluation of the synchronization loop, the corresponding throughput and the latency of the physical layer processing.



(a) $\mathrm{CP}_{\mathrm{SYNC}}$ (III-IV) (b) Throughput (III-IV) (c) $\mathrm{CP}_{\mathrm{PHY}}$ (III-IV)

Figure 8: Abstract simulation based analysis results for the evaluation of the synchronization loop, the corresponding throughput and the latency of the physical layer processing.

contentions on the communication architecture occur. Therefore, both the synchronization loop latency (Fig. 7a) and the frame throughput (Fig. 7b) gain performance as the effect of parallelization outweights the communication overhead. Contrary, configuration *IV* trades approximately the achieved gain by parallelization against the communication overhead. Please note that the execution times on the one hand of the Viterbi decoder and the other hand of the De-Interleaver and Reed-Solomon decoder differ by approximately a factor of three. Thus the possible parallelism cannot be exploited optimally. Future HW modifications and mapping decisions can and should address this issue.

However for the given case study both VCP HW configurations are supposed to be selected candidates for an implementation. In the previous section the seamless design flow from abstract mathematical analysis down to the final implementation has been sketched. To demonstrate this both have been set-up according to the abstract simulation framework [2]. The simulation results are illustrated in Figure 8. Here only the marked design points have been measured in simulation. To form the random variables an interpolation has been performed. The measurements show that the mathematical analysis differs in the expected value of the throughput, the frame and synchronization loop latency *less than 10%* compared to the abstract simulation based results. Main reason for this difference is the conservative estimate of the communication behavior in the mathematical analysis. This information could now be back annotated to the mathematical analysis model to improve the analysis results.

The proposed candidates achieve the given requirements, thus developers might start with one of those with the refinement of the application and hardware till the final system implementation is available. Otherwise developers can quickly step back and evaluate other implementation options on the high abstraction layers.

5 Conclusion

In this paper a mathematical analysis for SDR design has been proposed. This analysis and the developed workbench are fully integrated into a seamless design flow from high statistical analysis down to the final implementation in Electronic System Level (ESL) design. As exemplary illustrated in a case study the high level analysis allows at early design stage the identification of different design solutions. Those identified ones have then been evaluated seamlessly by an abstract simulation based environment to invest dynamic behavior of such SDRs. Our future work will concentrate on larger case studies including refinement to instruction and cycle accurate Instruction Set Simulators.

References

[1] M. Speth, H. Dawid, and F. Gersemsky. Design & Verification Challenges for 3G/3.5G/4G Wireless Baseband MPSoCs. In *MPSoC'08*, June 2008.

[2] T. Kempf, M. Doerper, R. Leupers, G. Ascheid, H. Meyr, T. Kogel, and B. Vanthournout. A Modular Simulation Framework for Spatial and Temporal Task Mapping onto Multi-Processor SoC Platforms. In *Proc. Design, Automation and Test in Europe (DATE'05)*, 2005.

[3] M.Gries. Methods for evaluating and covering the design space during early design development. *Integration, the VLSI Journal*, 38(2), 2004.

[4] Open SystemC Initiative. http://www.systemc.org.

[5] Synopsys Inc. http://www.synopsys.com.

[6] CoWare Inc. http://www.coware.com.

[7] T. Kempf, E.M. Witte, V. Ramakrishnan, G. Ascheid, M. Adrat, and M. Antweiler. SDR Baseband Processing Portability: A Case Study. In *SDR'08*, Washington, D.C., USA, October 2008.

[8] G. Schirner, A. Gerstlauer, and R. Domer. Abstract, Multifaceted Modeling of Embedded Processors for System Level Design. In *Proc. Asia and South Pacific Design Automation Conference ASP-DAC '07*, pages 384–389, 2007.

[9] A. Bouchhima, I. Bacivarov, W. Youssef, M. Bonaciu, and A.A. Jerraya. Using abstract CPU subsystem simulation model for high level HW/SW architecture exploration. In *Proc. Asia and South Pacific Design Automation Conference the ASP-DAC 2005*, pages 969–972, 2005.

[10] A. Gerstlauer, Haobo Yu, and D.D. Gajski. RTOS modeling for system level design. In *Proc. Design, Automation and Test in Europe Conference and Exhibition*, pages 130–135, 2003.

[11] D. Piergentili and D. Coupe. ESL Methods for Optimizing a Multi-media Phone Chip. EDA DesignLine, May 2008.

[12] K. Richter, M. Jersak, and R. Ernst. A Formal Approach to MpSoC Performance Verification. *Computer*, 36(4):60–67, 2003.

[13] Ernesto Wandeler, Lothar Thiele, Marcel Verhoef, and Paul Lieverse. System architecture evaluation using modular performance analysis: a case study. *International Journal on Software Tools for Technology Transfer (STTT)*, 8(6):649–667, November 2006.

[14] P. Ienne. Analytical Models of Communication for MPSoCs. In *MPSoC'08*, June 2008.

[15] R. A. Uhlig and T. N. Mudge. Trace-driven memory simulation: a survey. *ACM Comput. Surv.*, 29(2):128–170, 1997.

[16] W. Fornaciari, D. Sciuto, C. Silvano, and V. Zaccaria. A design framework to efficiently explore energy-delay tradeoffs. In *Proc. Ninth International Symposium on Hardware/Software Codesign CODES'01*, 2001.

[17] T.D. Givargis, J. Henkel, and F. Vahid. Interface and cache power exploration for core-based embedded system design. In *IEEE/ACM International Conference on Proc. Digest of Technical Papers Computer-Aided Design 1999*, pages 270–273, 1999.

[18] K. Lahiri, K. Lahiri, A. Raghunathan, and S. Dey. Performance analysis of systems with multi-channel communication architectures. In *Proc. Thirteenth International Conference on VLSI Design*, 2000.

[19] T. Wolf and M.A. Franklin. Performance models for network processor design. 17(6):548–561, 2006.

[20] E.A. Lee and D.G. Messerschmitt. Synchronous data flow. *Proceedings of the IEEE*, 75(9):1235–1245, 1987.

[21] The MathWorks Inc. MATLAB. http://www.mathworks.com/.

[22] Texas Instrument. DSP Libraries for TMS320C64x and TMS320C55x. http://www.ti.com/.

[23] Texas Instruments. TMS320C645x DSP Viterbi-Decoder Coprocessor 2 Reference Guide. http://www.ti.com/litv/pdf/spru972, April 2006.

2009 22nd International Conference on VLSI Design

Improved-quality Real-time Stereo Vision Processor

Sang-Kyo Han[1], SeongHoon Woo[2], Mun-Ho Jeong[2], Bum-Jae You[2]
[1]*University of Maryland, College Park, MD, U.S.A*
sangkyo@umd.edu
[2]*Korea Institute of Science and Technology, Seoul, South Korea*
mhjeong@kist.re.kr

Abstract

This paper presents a stereo vision processor with the form of ASIC that achieves enhanced quality depth maps and real-time performance. Our vision processor can be used broadly in practical applications. To improve depth map quality, pre- and post-processing units are adopted, and SFRs (Special Function Registers) are assigned to vision parameters for controllable quality. To meet real-time requirements, the stereo vision system is implemented on hardware using sophisticated design. We integrate image rectification, bilateral filtering, depth estimator and left-right consistency check blocks on a single silicon chip. This processor is fabricated in a 0.18-um standard CMOS technology, and can operate at 120MHz clock frequency achieving over 140 frames/s depth maps with 320 by 240 image size and 64 disparity levels. The system exploits 8-bit sub-pixel disparities for depth accuracy, and shows the throughput over 707 million PDS, which is better than results of any published work. The unrectified and unfiltered images taken at real environment are used as test inputs for performance and quality evaluation. Comparisons with previous ASIC implementations are presented to verify the improvement of this task.

1. Introduction

Stereo vision can produce depth information from two or more images taken on slightly different positions. The advantage over prevalent distance measurement devices like ultrasonic or laser equipments is that a complete scene is captured at once, yielding a contact-free acquisition of the spatial impression [1]. Therefore, stereo vision has potential uses in autonomous navigation, object recognition and surveillance systems, providing much more detailed information than devices based on wave reflection. For these applications, stereo vision should provide real-time performance and high quality depth (or disparity) map.

However, since to establish the correspondence between a pair of images has the computational complexity, this requires high performance for stereo matching. Moreover, a matching method affects on the quality of disparity map. Thus, various matching algorithms have been devised, and they can be classified into the area-based, the feature-based and the phase-based types. At the area-based approach, the correspondence between image points is solved by matching image intensity patterns. The feature-based method is first to detect edges and then to seek matches between these edges' intersections. The resulting maps are not as detailed as the area-based type to calculate the depth for every pixel. It is also ineffective in image regions without edges. At the phase-based algorithm, the depth is proportional to the phase displacement due to transform of the images using FFT (Fast Fourier Transform).

Recently, to meet the real-time requirement, these algorithms have been implemented on hardware instead of software. [2], [4] and [6] employed the area-based approaches such as dynamic programming, SAD (Sum of Absolute Differences) algorithm and Census transform, respectively, for stereo matching on FPGAs. [3] and [5] configured an FPGA-based stereo system using the phased-based method. Besides, stereo vision processors with the form of ASICs have been reported [1][7][12][13]. However, further improvement and development are required for vision to be ubiquitously used in practical applications.

Our research aims to develop high quality and real-time stereo vision processor on an ASIC, which can be used broadly in consumer products. We integrate the depth estimator based on SAD algorithm, pre- (image rectification, bilateral filtering) and post-processing (consistency check and more) units on a single silicon chip using a 0.18-um CMOS standard cell library. This processor can run at 120MHz clock frequency,

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.89

achieving over 140 fps dense depth maps with image size of 320*240 and 8-bit sub-pixel disparities. Our processor has the following advances different with previous stereo vision hardware.

First of all, image rectification, one of the pre-processing steps, is integrated on an ASIC to solve matching problem more efficiently. Rectification is to transform the images so that the epipolar lines are aligned horizontally, and thus simplifies stereo matching from a 2-D area search to a 1-D search along the epipolar line. In case of arbitrary placement of camera, (which is common), the epipolar line is skewed and the 1-D search will be ineffective [9]. Therefore, rectification is indispensable for stereo vision to be applicable to real world. In general, image rectification requires first camera calibration that obtains the intrinsic and the extrinsic parameters of camera. We find the homography matrix (**H**) from these parameters and apply it to unrectified images.

Secondly, we adopt the bilateral filtering [10] as another pre-processing step to improve the accuracy. Image filtering is obviously necessary for preserving the signal details while removing the noise [8]. In addition, any stereo algorithm should compensate for photometric variations between the cameras of the stereo rig [11]. It has been verified that the bilateral filter is suitable for these goals. Considered the real-time requirement, conventional filters with iterative property in algorithm are not adequate to vision applications.

Finally, this processor provides the consumer with adjustability and flexibility. Vision parameters which have impacts on the depth map quality are assigned to SFRs (Special Function Registers). Thus, users can control the quality of depth images. By parameter setting, output from each block can be easily adjusted.

This paper is organized as follows. Section 2 deals with backgrounds of stereo matching based on SAD, pre- and post-processing. In section 3, the architecture of vision processor and its operations are described in detail. Section 4 evaluates the quality of disparity map in real environment, discusses on chip performance and presents comparison data with previous works. Concluding remarks can be found in Section 5.

2. Backgrounds

A. Stereo matching based on SAD

Figure 1 shows the basic geometry of the parallel camera at which rectified images come out. The

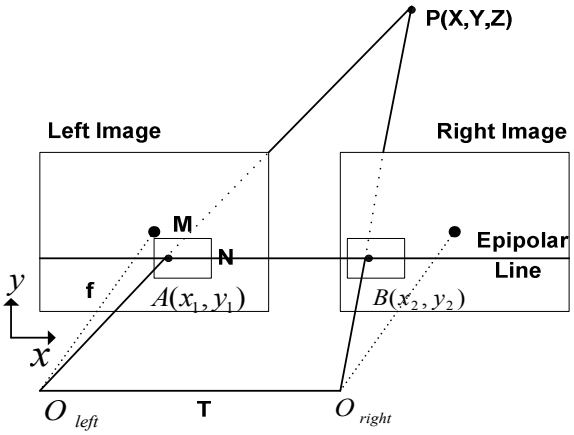


Figure 1. Basic geometry of stereo vision

baseline (T) denotes a line connecting the centers of two lenses, and the focal length (f) means a distance from a center of lens to image plane. When one point P on 3-D object is projected onto two image planes by perspective projection, two points indicate *A* and *B*, respectively. *The pixel B onto which P is projected in right image is called the corresponding pixel of A.* A plane through the baseline and point P is termed an epipolar plane, and the two straight-line intersections of an epipolar plane with the two image planes are called epipolar lines. Due to $y_1 = y_2$, each point in one image must be observed in the other image on a known epipolar line, which is the epipolar constraint. Suppose that the window size on each image is $M \times N$, *every number in the matrix indicates a pixel of the window. The disparity (d) denotes a distance between the centers of two windows. The SAD algorithm is one of the area-based approaches and defined as*

SAD(x, y, d) =

$$\sum_{u=-\frac{M-1}{2}}^{\frac{M-1}{2}} \sum_{v=-\frac{N-1}{2}}^{\frac{N-1}{2}} | I_L(x+u, y+v) - I_R(x+u+d, y+v) | \quad (1)$$

SAD becomes 0 in case of exact matching of two windows.

B. Image rectification

At the previous section it is assumed that two images were already rectified, but this is not realistic. As a pre-processing step of stereo matching, image rectification is an essential operation to simplify the correspondence problem. The task transforms the images so that the epipolar lines are aligned

horizontally. In this case the stereo matching can easily take advantage of the epipolar constraint, and the search area is reduced from two dimensional to one dimensional horizontal line. Given a pair of images with general epipolar geometry, rectification exploits the homography matrices with one per image. However, since an arbitrary arrangement of cameras makes the epipolar line be skewed, to obtain a proper homography generally needs camera parameters that can be known by calibration. Although stereo rigs are highly analogous each other, its cameras don't usually have same intrinsic and extrinsic parameters, which means different **H** matrix each rig. Therefore, in order to use effectively vision systems, the homography matrix should be able to be adjusted according to stereo system. In addition, the radial distortion of lens should be considered at rectification step. To meet the requirements, we assign the homography matrix and radial distortion parameters to SFRs on vision processor.

C. Bilateral filtering

Toward improved stereo vision, a filtering that does not blur across range discontinuities and compensates for photometric variations between a pair of cameras takes an important position. A conventional filtering with an iterative property conflicts with the real-time requirement. Recently, C. Tomasi and R. Manduchi [10] proposed an alternative non-iterative bilateral filter. This is a weighted average of the local neighborhood samples, where the weights are computed based on temporal (or spatial in case of images) and radiometric distances between the center sample and its neighbors. In [11], Ansar suggested a novel approach incorporating a bilateral filtering into a background subtraction step, and verified that it is superior to conventional methods such as Laplacian filtering, DoG (Difference of Gaussians) filtering, Rank and Census Filtering. We adopt the bilateral filtering as another pre-processing stage, and the following describes its algorithm. Typical background subtraction method is composed of a low-pass filtering and a subtraction of the low-filtered image from an original image as follows;

$$I' = I - I * G(\sigma_{l\arg e}) \tag{2}$$

where I is an original image, $G(\sigma_{l\arg e})$ is a Gaussian mask with a variance $\sigma_{l\arg e}$. In Ansar's approach, the low-pass filtering with a Gaussian mask is replaced with a bilateral filtering, which computes the weighted average of the pixels with the weights depending on both the spatial and intensity difference between the central pixel and its neighbors [11]. The filter returns $B(x)$ at x in I as follows;

$$B(x) = \frac{\sum_{\xi \in \Omega} I(\xi) c(\xi, x) s(I(\xi), I(x)) d\xi}{\sum_{\xi \in \Omega} c(\xi, x) s(I(\xi), I(x)) d\xi} \tag{3}$$

where Ω is the filter support, $I(x)$ is the intensity at x, and c and s are the weight functions defined as

$$c(\xi, x) = \exp(-\frac{1}{2}(\frac{|\xi - x|}{\sigma_d})^2) \tag{4}$$

$$s(I(\xi), I(x)) = \exp(-\frac{1}{2}(\frac{|I(\xi) - I(x)|}{\sigma_r})^2) \tag{5}$$

where σ_d and σ_r are the standard deviations of the spatial component and the intensity component, respectively. Finally, the filtered output is obtained as

$$I' = I - B \tag{6}$$

D. Left-right consistency check

Since stereo images are taken from a different position, there are areas that are visible to one camera only. This is because protruded objects hide objects in the background. This is called occlusion effect and results in unmatched points. This problem can be overcame by left-right consistency check, in which stereo matching is performed twice from left to right (LR) and from right to left (RL). Matched points are confirmed only when the minimum disparity of LR matching is very close to that of RL matching. In our system, when the difference of minimum disparities between two matches comes within a threshold value, the pixel is finally recognized as a matched point. High threshold increases the number of candidates of matching points while the accuracy of depth map decreases. On the contrary, low threshold decreases the number of candidates and even some of points can be left unmatched. Therefore, the adjustable threshold disparity value like our processor will be one effective way to generate reliable disparity maps.

3. ASIC Architecture

A. Top level architecture

Figure 2 shows the top level architecture of our stereo vision processor with the form of ASIC. CIS Interface, Image Rectification, Bilateral Filtering, Depth Estimator, Interface Block and SFRs are integrated on a single silicon chip. A pair of CMOS Image Sensor (CIS) cameras composes of a stereo camera. Through CIS Interface, input images of 320*240 sizes from cameras transfer to Image Rectification in which images are rectified using the homography matrix. In Depth Estimator, bilateral filtered images are processed to extract disparity maps with 3-D information. Interface Block delivers dense depth maps with 8-bit sub-pixel disparities to user's machine, and receives user's values to set SFRs. The followings are parameters assigned to SFRs: homography matrix, radial distortion of lens, threshold disparity value and so on.

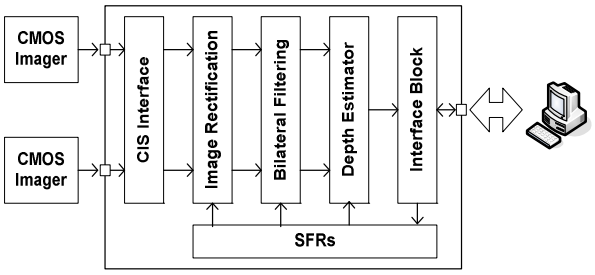


Figure 2. Top level architecture of stereo vision processor

B. Image rectification block

Figure 3 presents the diagram of image rectification block. The diagram incorporates image rectification and compensation of radial distortion. Frame Buffer stores images from CIS Interface, and provides pixel addresses and data to other blocks. Matrix Calculation block generates the address of transformed pixel using the homography matrix from SFR, and delivers it to Radial Distortion block that offsets radial distortion of lens. With results of two computations, New Pixel Generator produces the address of new pixel, and then fetches 4 pixels data including its neighbors from Frame Buffer. Finally, a rectified image comes out after interpolation based on 4 pixels. Notice that computations are mainly composed of findings of address because rectification is a mapping of pixel onto a new location.

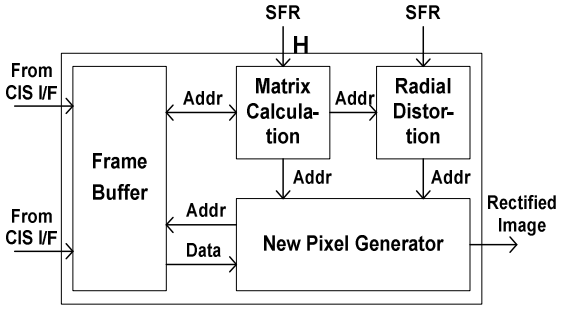


Figure 3. Block diagram of image rectification

C. Bilateral filtering block

Figure 4 describes the block diagram of bilateral filtering. We adopted 11*11 bilateral background subtraction with σ_d =3.7, σ_r =30 in (4) and (5), respectively. In order to filter one pixel, 11*11 pixels are required and they are stored at vertical buffers. Weight calculation block finds weights depending on both the spatial and intensity difference between the central pixel and its neighbors. The spatial and intensity difference values are stored at each table with 121 levels and 256 levels, respectively. ABS block calculates absolute difference between weighted average and original image pixel value, that is, $I' = I - B$ in (6). [8] deals with more details about the architecture below.

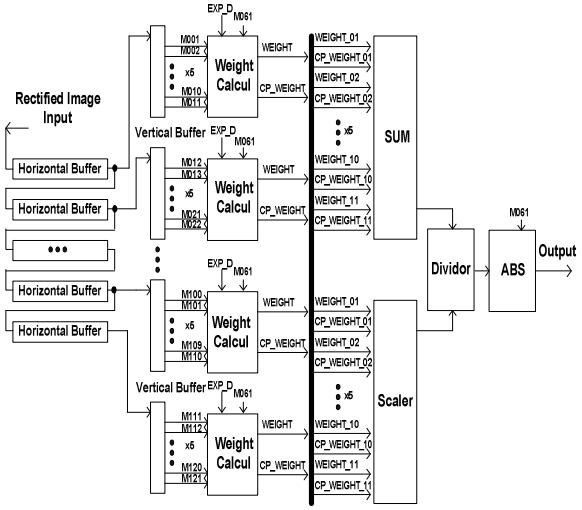


Figure 4. Block diagram of bilateral filtering

D. Depth estimator block

Figure 5 shows the block diagram of depth estimator with left-right consistency check. In LR SAD block, a window with the size of 11*11 around candidate pixel is taken at each image along epipolar

lines, and then SAD is computed at the condition of fixed left window and moving right window by 64 disparity levels. This computation is performed for every pixel at right image. RL SAD block also performs same tasks except fixed right window and moving left window. Local Minima block enhances the disparity accuracy. This unit finds the disparity to have minimum SAD satisfying local minimum among 3 neighboring SAD values, which are also used at Sub-pixel Processing block. In Consistency Check block, minimum disparity from Local Minima is compared with disparity from RL SAD. If the difference falls within a threshold value, matching is successful.

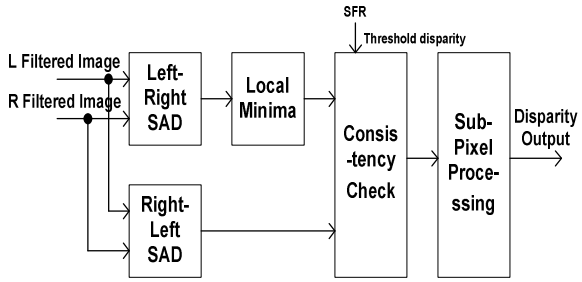


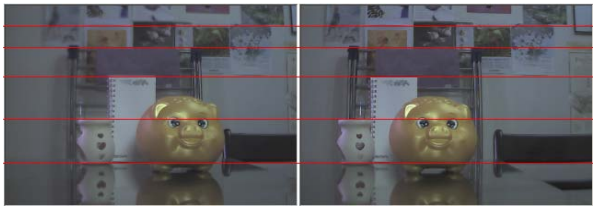
Figure 5. Block diagram of depth estimator

4. Results and Discussion

The stereo vision processor described in this paper generates 320 by 240 depth images with 64 disparity levels based on 8-bit sub-pixel processing. At the 120MHz operating clock frequency, a dense disparity map can be produced at the rate of 144 frames/s. The design integrates image rectification, bilateral filtering, depth estimator with LR check and more on a single silicon chip that is fabricated using 0.18um CMOS technology. In Table 1, the performance and specifications of our system are compared with previous works based on ASIC implementation. From listed processors, we can say that presented design not only achieves better performance, but is the improved system at overall aspects. Especially, a high disparity level is remarkable because it has an important role at system quality. Besides, to integrate rectification block and to adopt SFRs for quality control are unique tasks.

To measure the throughput of presented vision system, we apply one common comparison metric, which is the PDS (Points times Disparity per Second). From a simple calculation (320*240*144*64), the processor achieves about 707 million PDS that is better result than any published work [1][2][14][15]. We verify the design with input images taken at real world. As shown in Figure 6 (a) and (b), unrectified and unfiltered stereo images are used as left-right test inputs. Two images are skewed each other within 5 pixels. Note that right image is a little obscure due to photometric or geometric variations of right camera, which can be occurred in every stereo rig. Figure 6 (c) and (d) are rectified images, where red straight lines are used to mark the difference of *y*-coordinates after rectification. We could find that bilateral filtering operates as an edge-preserving smoother in Figure 6 (e) and (f). Figure 6 (g) is the depth map extracted from filtered images, and closer objects have more intensity in our system. At the object boundary, disparity data is blurred a little bit. However, this is mainly because the processor was designed to invalidate uncertain matches. In order to reduce errors, it is considered that to invalidate uncertain matches is better as long as correct matches are not invalidated radically [16].

(a) input left image (b) input right image



(c) rectified left image (d) rectified right image

(e) filtered left image (f) filtered right image

(g) depth map image

Figure 6. Output images from real inputs

5. Conclusions

We have presented a stereo vision processor extracting disparity images with improved quality in real-time. For stereo vision to be ubiquitously used in practical applications, the system was fabricated on

ASIC form. By integrating pre- and post-processing units on a single chip, improved depth maps with the size of 320*240 are generated with the speed of 144 fps. It makes for vision parameters to be assigned to SFRs the quality of depth map controllable. The processor achieves significantly high throughput over 707 million PDS and provides 8-bit sub-pixel disparities.

Acknowledgement

This work was supported by the IT R&D program of MKE/IITA, 2006-S-028-01, Development of Cooperative Network-based Humanoids Technology.

Table 1. Comparison of ASIC implementations

	[7] (2002)	[1] (2004)	[13] (2004)	[12] (2006)	Ours
Technology	0.25um	0.25um	0.18um	0.18um	0.18um
Clock Freq.	100MHz	75MHz	125MHz	100MHz	120MHz
Performance	- (80GOPS)	50fps	10fps	1000fps (estimated)	144fps (707M PDS)
Image Size	512*480	256*192	320*240	32*32	320*240
Disparity Levels	N/A	25	N/A	N/A	64
Rectification	Without	Without	Without	Without	Integrated
Image Filter	Filter	Median Filter	Without	Without	Bilateral Filter
Stereo Algorithm	SAD	SSD & Census	SAD	SAD	SAD
Post-processing	LR Check	LR Check	Without	Without	LR Check
Other Feature	Motion Estimator	-	-	-	SFRs

6. References

[1] M. Kuhn et al., "Efficient ASIC Implementation of a Real-time Depth Mapping Stereo Vision System", Circuits and Systems, Proceedings of the 46th IEEE International Midwest Symposium on Volume 3, 27-30 Dec. 2003 Page(s):1478 - 1481.

[2] S. Park and H. Jeong, "Real-time Stereo Vision FPGA Chip with Low Error Rate", Multimedia and Ubiquitous Engineering, International Conference on, 26-28 April 2007 Page(s):751 – 756.

[3] Divyang K. Masrani and W. James MacLean, "A Real-Time Large Disparity Range Stereo System using FPGAs", Computer Vision Systems, IEEE International Conference on**,** Publication Date: 04-07 Jan. 2006 page(s): 13- 13.

[4] M. Hariyama et al., "FPGA implementation of a stereo matching processor based on window-parallel-and-pixel-parallel architecture", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Volume E88-A, Issue 12, Dec 2005, Page(s):3516-3522.

[5] A. Darabiha, J. Rose and W.J. MacLean, "Video-Rate Stereo Depth Measurement on Programmable Hardware", Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, Volume 1, 18-20 June 2003 Page(s):I-203 - I-210.

[6] J. Woodfill and B.V. Herzen, "Real-Time Stereo Vision on the PARTS Reconfigurable Computer", Proceedings of the 5th IEEE Symposium on FPGA-Based Custom Computing Machines, Page(s):201, 1997.

[7] P.J.Burt, "A pyramid-based front-end processor for dynamic vision applications", Proceedings of the IEEE, July 2002, Volume: 90, Page(s): 1188-1200.

[8] Sang-Kyo Han et al., "Architecture and Implementation of Real-Time Stereo Vision with Bilateral Background Subtraction", Lecture Notes in Computer Science, 2007, Volume: 4681, Page(s): 906-912.

[9] H.-H.P. Wu et al., "Projective rectification based on relative modification and size extension for stereo image pairs", IEE Proc. Image Signal Processing, Oct 2005, Volume: 152, Page(s):623-633.

[10] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images", in Proc. IEEE Intl. Conf. Computer Vision, 1998, pp. 839-846.

[11] Adnan Ansar, Andres Castano and Larry Matthies, "Enhanced Real-time Stereo Using Bilateral Filtering", in Proc. Intl. Symp. on 3D Data Processing, Visualization, and Transmission, 2004, Page(s):455-462.

[12] Masanori Hariyama et al., "1000 frame-sec Stereo Matching VLSI Processor with Adaptive Window-Size Control", Solid-State Circuits Conference, 2006. ASSCC 2006. IEEE Asian, Page(s):123 – 126.

[13] Masanori Hariyama et al., "VLSI processor for reliable stereo matching based on window-parallel logic-in-memory architecture", VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on, Page(s):166 - 169.

[14] Divyang K. Masrani and W. Janes MacLean, "A Real-Time Large Disparity Range Stereo-System using FPGAs", Computer Vision Systems, 2006 ICVS '06. IEEE International Conference on, Page(s): 13- 13.

[15] Javier Diaz et al., "High Performance Stereo Computation Architecture", Field Programmable Logic and Applications, 2005. International Conference on, 2005, Page(s): 463-468.

[16] Heiko Hirschmuller, "Improvements in Real-time Correlation-Based Stereo Vision", Proceedings of the IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV'01). Page(s): 141.

Session 5A

SRAM and Random Number Generation

A 7T/14T Dependable SRAM and Its Array Structure to Avoid Half Selection

Hidehiro Fujiwara, Shunsuke Okumura, Yusuke Iguchi, Hiroki Noguchi, Hiroshi Kawaguchi, and Masahiko Yoshimoto
Graduate school of Engineering Kobe University, Kobe, Japan
fujiwara@cs28.cs.kobe-u.ac.jp

Abstract

We propose a novel dependable SRAM with 7T cells and their array structure that avoids a half-selection problem. In addition, we introduce a new concept, "quality of a bit (QoB)" for it. The dependable SRAM has three modes (normal mode, high-speed mode, and dependable mode), and dynamically scales its reliability and speed by combining two memory cells for one-bit information (i.e. 14T/bit). Monte Carlo simulation demonstrates that, in a 65-nm process technology, the minimum voltages in read and write operations are improved by 0.20V and 0.26V, respectively, with a bit error rate of 10^{-8} kept. The cell area overhead is 11%, compared to the conventional 6T cell in the normal mode.

1. Introduction

Recently, they have paid attention to dependable computing systems, as silicon LSIs support massive infrastructure in society. However, the advanced process technology tends to cause accidental errors like a soft error and negative bias temperature instability (NBTI), more frequently. In addition, there might be some errors left in a design, manufacturing, or test phase. It is supposed to be almost impossible to perfectly eliminate these human-induced errors in a future complicated LSI. That is, a product will be shipped with some errors, and accidentally malfunction. We no longer expect error-free LSIs with sufficient operating margins.

Since reliability is varied with operating conditions (speed, supply voltage, temperature, and even altitude corresponding to a soft error), it is desirable to dynamically improve the reliability on worse conditions. Furthermore, required reliability depends on an application software, which indicates that the reliability should be changed in accordance with the application.

Considering this background, we propose an SRAM that can dynamically control its reliability. An SRAM has recently dominated operating margins of a chip due to a large number of transistors [1-7]. Other than the reliability, the proposed SRAM also achieves fast operation and/or low-poer operation, with the same reliability kept as the conventional SRAM. Namely, the proposed SRAM can change quality of its information, in terms of reliability, speed, and/or power.

In the next section, we describe the overview of the proposed SRAM. In Section 3, we propose a novel 7T memory cells to dynamically improve the quality of information, and introduce a new concept called "quality of a bit" (QoB). In Section 4, we discuss the reliability of the proposed memory cell, from view points of a cell current and a bit error rate. In Section 5, we mention the new cell array structure for the proposed SRAM to avoid the half-selection problem. The final section summarizes this paper.

2. Dependable SRAM: Overview

Operating conditions affect reliability of an SRAM, while the reliability depends on an application software that uses the SRAM. An encryption program and a screen saver program demand different levels of reliability. This means that the reliability is changed by the operating conditions, and is dependent on the application.

In our proposed dependable SRAM, reliability and speed of an SRAM can be dynamically changed on a block-by-block basis, as illustrated in Figure 1. In the normal mode (Blocks 0-3), assignment is as usual as one memory cell has one bit. On the other hand, in the dependable or high-speed mode (Blocks 4 and 5), one-bit information is stored in two memory cells by combining a pair of memory cells. This mechanism selectively realizes the high reliability or high speed, while the memory capacity becomes a half in these modes.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.54

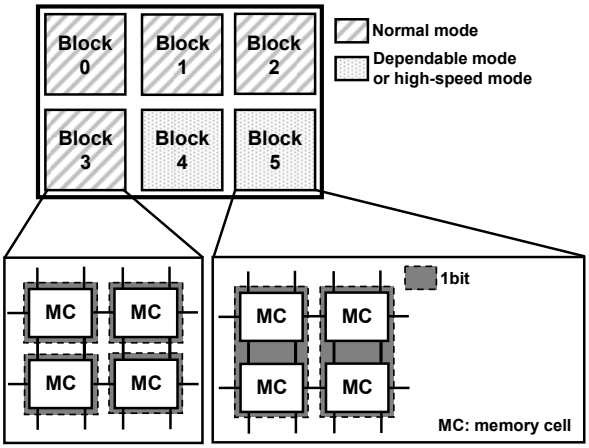


Figure 1. Dependable SRAM.

However, this dynamic switching between the typical dependability and high dependability opens up new resource allocation in an SRAM. For instance, an operating system (OS) can allocate an encryption program to the high-dependability block. An application software can also change the reliability of its data by system call. Encryption data or personal information should be in the high-dependability block. If memory utilization of programs and data is 50% or less, the high-dependability mode can be aggressively exploited by the OS, without the memory-capacity overhead. A small code with small data always runs in the high-dependability mode.

In the next section, we explain how to achieve the proposed dependable SRAM on the circuit level.

3. Dependable Memory Cell and the Concept of the Quality of a Bit (QoB)

Figure 2 depicts the proposed 7T cell (14T for two cells). Two pMOSes are added to internal nodes ("N00 and N10", "N01 and N11") in a pair of the conventional 6T memory cells shown in Figure 3.

3.1. Normal Mode

If the additional transistors are turned off (/CTRL = "H"), the 7T cell acts as the conventional 6T cell. This is called "a normal mode" in this paper.

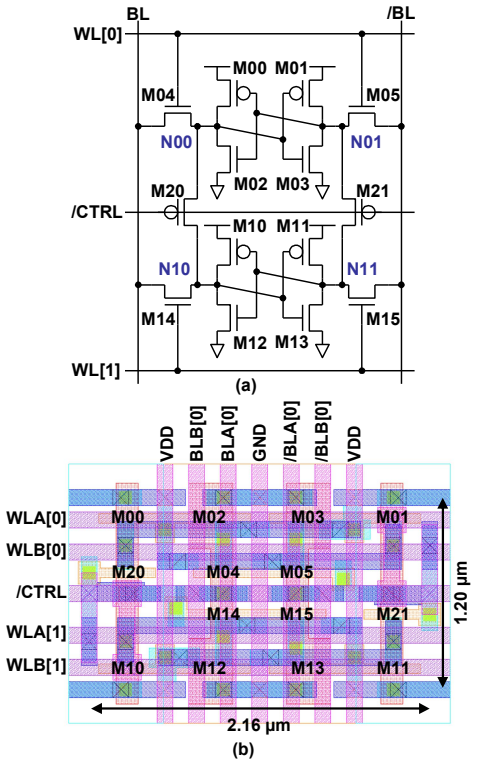


Figure 2. Proposed 7T memory cell: (a) schematic and (b) layout.

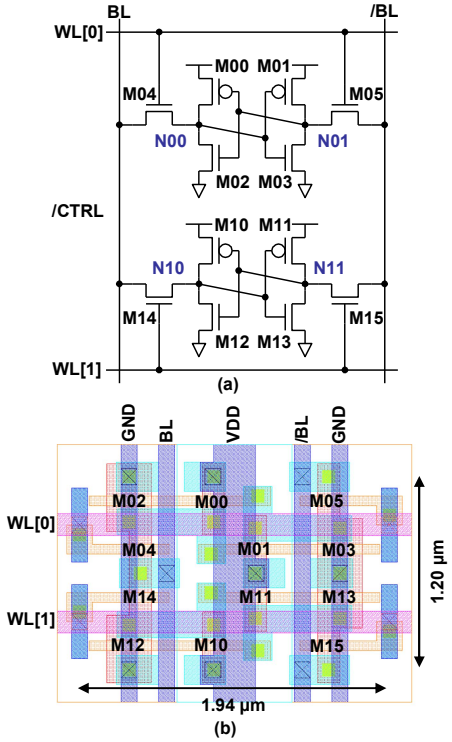


Figure 3. Conventional 6T memory cell: (a) schematic and (b) layout.

3.2. High-Speed Mode

Alternatively, if the additional transistors are turned on (/CTRL = "L"), and the internal nodes are shared by the pair of memory cell. The high speed is achieved when both WL[0] and WL[1] are driven, which enables a faster readout.

3.3. Dependable Mode

The most significant usage of the proposed 7T memory cell is the dependable mode. The additional transistors are activated, but either WL[0] or WL[1] is asserted. Thus, only one word line is asserted. By doing so, a larger β ratio and static noise margin can be obtained because the dependable mode has two access transistor but four drive transistors in a memory cell.

3.4. Quality of a Bit

As mentioned above, we have three modes in the proposed 7T memory cell. Table 1 summarizes these modes. In the normal mode, one-bit datum is stored in one memory cell, which is the most area-efficient. In the high-speed mode and dependable mode, one-bit datum is stored in two memory cells although the quality of the information is different from the typical mode. The "higher-speed" or "more dependable" information can be obtained. We call this concept "quality of a bit (QoB)". The quality of the information is scalable in our proposed memory cell.

Table 1. Three modes in 7T memory cell.

	# of MCs comprising 1 bit	# of WL drives	/CTRL
Normal	1 (7T/bit)	1	"H" (off)
High-speed	2 (14T/bit)	2	"L" (on)
Dependable	2 (14T/bit)	1	"L" (on)

4. Dependability Simulation

In this section, we discuss the dependability of our proposed memory cell from view points of a cell current and a bit error rate.

4.1. Cell Current

Figure 4 exhibits the distributions of the cell currents in the conventional 6T cell and proposed high-speed cell (SS corner, VDD=1.0V, high temp.). The worst case cell current in the high-speed mode is increased by 133%. Statistically, it is very unlikely that two memory cells are both the worst, which is the reason why the cell current is more than double.

Figure 5 compares their worst-case bitline delays (SS corner, VDD = 1.0V, high temp.) in the read operation. The bitline delay is defined as a period from a time at which a WL rises to VDD/2 to a time at which a differential voltage between BL and /BL is expanded to 100 mV. The worst-case bitline delay time is improved by 53% in the high-speed mode.

Furthermore, the high-speed mode has higher tolerance of bitline leakage, thanks to the on current improvement [3-5].

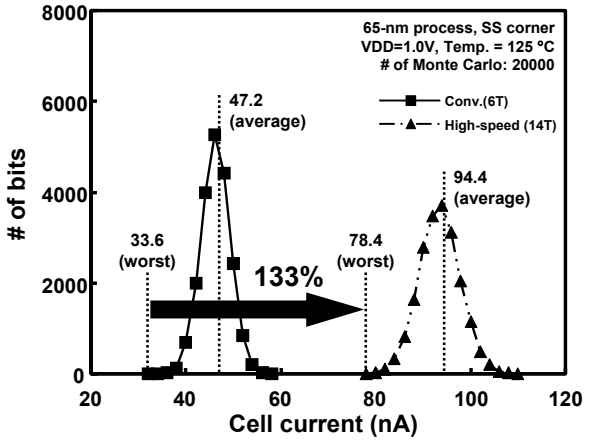


Figure 4. Cell current distributions in the conventional 6T cell and proposed high-speed cell.

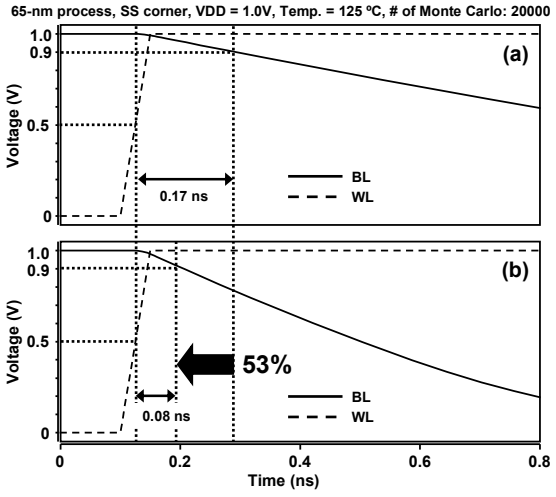


Figure 5. The worst case bitline delay simulation: (a) conventional and (b) proposed high-speed cell.

4.2. Bit Error Rate (BER)

Figure 6 shows comparisons of static noise margins (SNMs) and write trip points (WTPs) between the proposed and conventional memory cells [6-7]. The number of Monte Carlo simulation sample is 2,000. SNMs and WTPs of the proposed memory cells are in the dependable mode and the high-speed mode, respectively. The proposed memory cell has larger operating margins compared with the conventional one. Respective improvement of the worst case SNM and WTP are 40 mV and 60 mV.

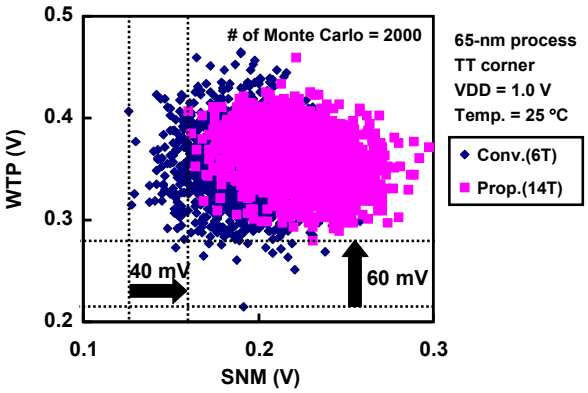


Figure 6. Static noise margins (SNMs) and Write trip points (WNMs).

Figure 7 (a) illustrates a bit error rate (BER) in the read operation [8-9]. The dependable mode works fine below 0.62 V with a BER of 10^{-8} kept even in the worst-case condition (FS corner, high temp.). The minimum operating voltage and BER are improved by 0.20 V and 4.0 x 10^{-5}, compared with the normal mode. The dependable mode is the most reliable in the read operation.

Figure 7 (b) is a BER in the write operation (worst-case condition: FS corner, low temp.). The dependable mode is inappropriate because the conductance of the access transistor is not sufficient. Instead, in the write operation, the high-speed mode should be exploited even in the usage of the dependable mode. In the high-speed mode, the conductance of the access transistors is doubled, and variation is suppressed. Thereby, the write margin becomes larger. The proposed memory cell functions at 0.69 V with a BER of 10^{-8} kept. The minimum operating voltage and BER are improved by 0.26 V and 3.2 x 10^{-3}, compared with the normal mode.

Furthermore, the high-speed and dependable modes sustain lower retention voltages, as illustrated in Figure 7 (c). This is because a retention voltage in a memory cell is averaged each other by the additional transistors.

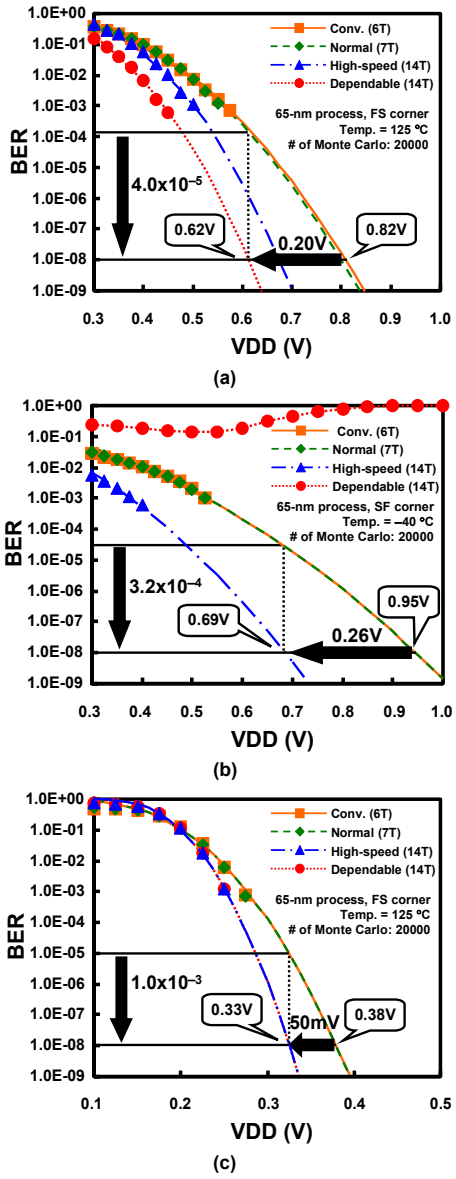


Figure 7. Bit error rates (BERs): (a) read operation, (b) write operation, and (c) retention.

Figure 8 compares the standby leakage powers between the conventional and proposed cells at the minimum operating voltages [10]. The 14T cell lowers the leakage power by 61% and 32% per cell in the typical and worst-case conditions, respectively. In particular, the gate leakage is decreased by more than 75%. This indicates that, even if one bit is stored in a memory cell pair, the 14T cell pair has a less leakage than one 6T cell in the typical condition. As well, the proposed memory cell mitigates the NBTI due to its low-voltage operation.

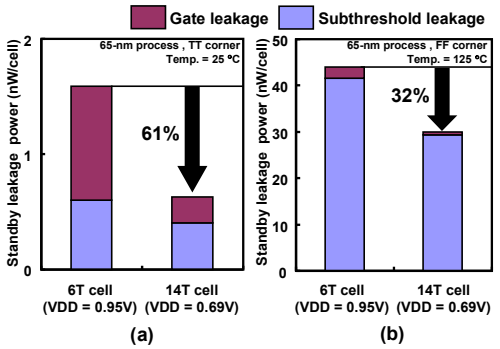


Figure 8. Standby leakage power per cell: (a) TT corner, 25 °Cand (b) FF corner, 125 °C.

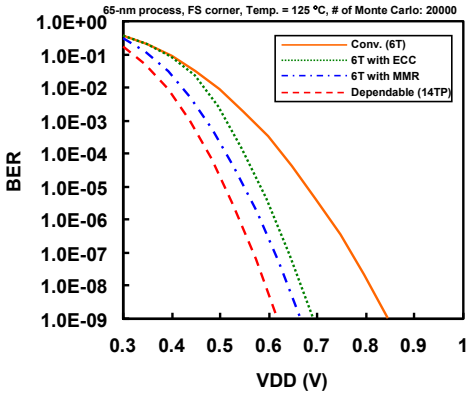


Figure 9. Comparison of BERs among the conventional methods and 14TP dependable mode.

Figure 9 illustrates the comparison of BERs among the error correction code (ECC), the multi-module redundancy (MMR) and proposed dependable mode, in the read operation. The proposed memory cell achieves the best BER.

In the proposed dependable SRAM, we can select an appropriate mode, in terms of area overhead, speed or dependability. The proposed SRAM is also suitable to fine-grain dynamic voltage scaling (DVS) for low-power operation because it works in a low-VDD region.

5. Design in 65-nm Process Technology

The proposed SRAM possibly incurs the half-selection problem [11]. Two wordlines in a memory cell pair have to be activated in the write operation, which might cause unexpected flips in unselected memory cells. Figure 10 portrays the situation. Although only the selected pair needs to be written in, the neighbors are half-selected. Unfortunately, the static noise margin in the half-selected pairs is as much as the high-speed mode, which is smaller than the dependable mode.

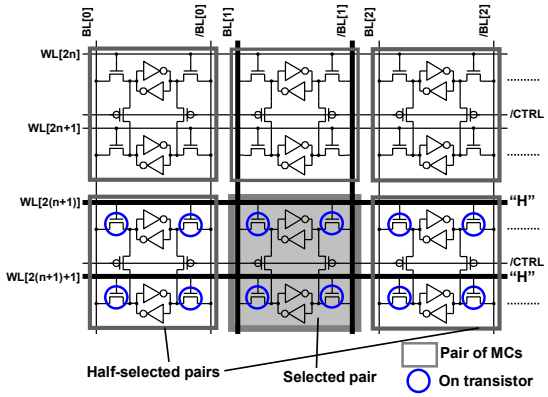
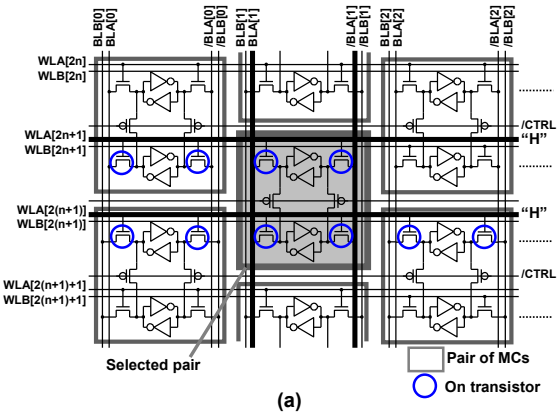


Figure 10. Conventional memory cell array structure with half-selection problem



	BL[0]	BL[1]	BL[2]	BL[3]	BL[4]	BL[5]	BL[6]	BL[7]
WL[2n]	WLA	WLA	WLA	WLA	WLB	WLB	WLB	WLB
WL[2n+1]	WLA	WLA	WLB	WLB	WLA	WLA	WLB	WLB
WL[2(n+1)]	WLA	WLA	WLA	WLA	WLB	WLB	WLB	WLB
WL[2(n+1)+1]	WLA	WLA	WLB	WLB	WLA	WLA	WLB	WLB

(b)

Figure 11. Proposed memory cell array: (a) memory cell array structure without half-selection problem, and (b) wordline mapping.

To solve this problem, we adopt a new array structure in Figure 11 (a). In every other column blocks, we shift a memory cell pair as much as a cell height, and introduce a wordline pair: WLA and WLB. There is no cell area overhead paid for the wordline pair since they are laid out on a metal-4 layer (see Figure 2 (b)). Figure 11 (b) is wordline mapping for WLA and WLB. When the hatched pair is selected, WLA[2n+1] and WLA[2(n+1)] have to be asserted. This memory cell array structure solves the half-select problem up to eight column blocks.

Fig. 12 is a micrograph of a dependable 64-kb SRAM test chip, designed and fabricated in a 65-nm CMOS process technology.

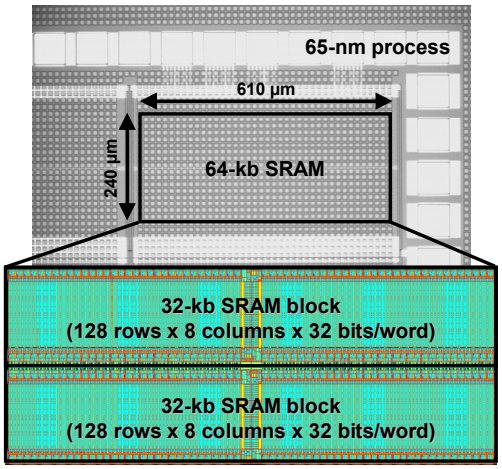


Fig. 12. Chip micrograph and layout

6. Conclusion

We designed a dependable SRAM with 7T/14T memory cells, which have three modes (normal mode, high-speed mode, and dependable mode), in a 65-nm process technology. The proposed SRAM can dynamically change its speed and dependability, based on the concept of "quality of a bit (QoB)". By running Monte Carlo simulation, we confirmed that the minimum voltages in read and write operations are improved by 0.20V and 0.26V, respectively, with a bit error rate of 10^{-8} kept. In addition, we proposed the new memory cell array structure to avoid the half-selection problem. The proposed SRAM will open up new memory allocation in an LSI system. Users can change its performance, depending on reliability, speed, supply voltage (dynamic voltage scaling: DVS), standby power, and/or application.

Acknowledgements

This work was supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Cadence Design Systems, Mentor Graphics and Synopsys, Inc.

References

[1] International Technology Roadmap for Semiconductors 2007 http://www.itrs.net/Links/2007ITRS/Home2007.htm.

[2] H. Pilo, J. Barwin, G. Braceras, C. Browning, S. Burns, J. Gabric, S. Lamphier, M. Miller, A. Roberts, F. Towler, "An SRAM Design in 65nm and 45nm Technology Nodes Featuring Read and Write-Assist Circuits to Expand Operating Voltage," 2006 Symposium on VLSI Circuits Digest of Technical Papers, pp. 15-16, June 2006.

[3] N. Verma, A. P. Chandrakasan, "A 65nm 8T Sub-Vt SRAM Employing Sense-Amplifier Redundancy," ISSCC 2007 Digest of Technical Paper, pp. 328-329, February 2007.

[4] T. H. Kim, J. Liu, J. Keane, C. H. Kim, "A High-Density Subthreshold SRAM with Data-Independent Bitline Leakage and Virtural Ground Replica Scheme," ISSCC 2007 Digest of Technical Papers, pp. 330-331, February 2007.

[5] I. J. Chang, J. J. Kim, S. P. Park, and K. Roy, "A 32kb 10T Subthreshold SRAM Array with Bit-Interleaving and Differential Read Scheme in 90nm CMOS," ISSCC 2008 Digest of Technical Papers, pp. 398-300, February 2008.

[6] E. Seevinck, F. J. List, and J. Lohstroh, "Static-Noise Margin Analysis of MOS SRAM Cells", IEEE JSSC, vol. 22, no. 5, pp. 748-754, October 1987.

[7] E. Grossar, M. Stucchi, K. Maex, and W. Dehaene, "Statistically Aware SRAM Memory Array Design," ISQED, pp. 25-30, March 2006.

[8] M. Yamaoka, N. Maeda, Y. Shinozaki, Y. Shimazaki, K. Nii, S. Shimada, K. Yanagisawa, And T. Kawahara, "90-nm process-variation adaptive embedded SRAM modules with power-line-floating write technique," IEEE J. Solid-State Circuits, vol. 41. no. 3, pp. 705-711, March 2006.

[9] Y. Morita, H. Fujiwara, H. Noguchi , K. Kawakami, J. Miyakoshi, S. Mikami, K. Nii, H. Kawaguchi, and M. Yoshimoto, "A Vth-Variation-Tolerant SRAM with 0.3-V Minimum Operation Voltage for Memory-Rich SoC under DVS Environment," 2006 Symposium on VLSI Circuits Digest of Technical Papers, pp. 16-17, June 2006.

[10] K. Nii, Y. Tsukamoto, T. Yoshizawa, S. Imaoka, Y. Yamagami, T. Suzuki, A. Shibayama, H. Makino, and S. Iwade, "A 90-nm low-power 32-kB embedded SRAM with gate leakage suppression circuit for mobile applications," IEEE J. Solid-State Circuits, vol. 39. no. 4, pp. 684-693, April 2004.

[11] H. Yamauchi, T. Suzuki, and Y. Yamagami, "A 1R/1W SRAM Cell Design to Keep Cell Current and Area Saving against Simultaneous Read/Write Disturbed Accesses," IEICE Trans. Electronics, vol. E90-C, no. 4, pp. 749-757, April 2007.

A 4Gbps 0.57pJ/bit Process-Voltage-Temperature Variation Tolerant All-Digital True Random Number Generator in 45nm CMOS

Suresh Srinivasan*, Sanu Mathew, Vasantha Erraguntla*, Ram Krishnamurthy
*Circuits Research Lab, Bangalore Design Lab, Intel Corporation, Bangalore, India
Circuits Research Lab, Hillsboro, OR 97124, USA
suresh.srinivasan@intel.com, sanu.k.mathew@intel.com, ram.krishnamurthy@intel.com

Abstract

This paper describes an all-digital on-die true random number generator implemented in 45nm CMOS technology, with random bit throughput of 4Gbps and total energy consumption of 0.57pJ/bit. A 2-step tuning mechanism enables robust operation in the presence of up to 20% fabrication-time process variation as well as immunity to run-time voltage and temperature fluctuation. The 100% use of digital components enables a compact layout occupying 1024μm^2 with high entropy/bit of 0.94, and scalable operation down to 0.5V, while passing all NIST RNG tests.

1. Introduction

High-entropy randomness is the foundation of data encryption, secure web communications and complex statistical analyses. Random number generators are therefore key blocks of processor platforms, responsible for generating secure keys in cryptography algorithms, session-IDs in secure internet protocols, mobile-internet-device IDs, Monte-Carlo simulations, and various operating system protocols. The quality of random numbers has a significant impact on the vulnerability of security algorithms and the accuracy of statistical simulations. Additionally, performance of secure mail/web servers and kernel scheduling routines is often limited by system stalls caused due to an insufficient pool of random numbers. Therefore there is a need for a high bit-rate high-entropy true random number generator that can be fabricated in a digital CMOS process with robust operation in the presence of process, voltage and temperature (PVT) variations. An all-digital design is motivated by the need for an on-die scalable energy-efficient implementation that can be easily ported to sub-32nm process technologies.

Software generated random numbers are pseudo-random in nature due to their dependence on events such as user-interface interrupts, page-faults, incoming TCP/IP requests and kernel system calls. Since these events are linked to user activity, they can be manipulated by malicious attackers, resulting in security loopholes [1]. True Random Number Generators (TRNGs) on the other hand, extract entropy from natural phenomena, such as device/thermal/flicker noise, or radioactive decay and are therefore immune to side-channel attacks. Among these natural sources, thermal noise is the most widely-used entropy source in TRNG designs. Thermal noise voltage is generated due to small fluctuations in channel current caused by the thermal vibrations of charge carriers in a conducting channel. This noise can be harvested by (i) amplifying the differential voltage across a pair of resistors using several stages of high-gain differential amplifiers and A/D converters [2][3][4], (ii) extracting edge-jitter from parallel chains of long ring-oscillators [5][6] (iii) sampling resolution state/time of meta-stable cross-coupled inverters [7][8] (iv) engineering SiN devices to increase the magnitude of thermal noise [9].

The resistor-amplifier-ADC based designs use analog circuit techniques that do not scale well to future digital process technologies and also require a very stable operating environment, free from deterministic sources of noise such as power-supply fluctuations, neighboring conductor coupling and temperature variations. The entropy of these designs may be disrupted due to random/systematic process variation or device drift due to aging. Mechanisms to counter device variation exist for ring-oscillator and meta-stability based schemes. These designs examine the generated bits to detect deterministic biases and introduce a counter-bias to offset the inherent mismatch. The counter-bias may be in the form of an analog bias voltage generated using a conditionally clocked capacitive charge-pump [7][8] or an external voltage source[5]. Once again, these bias generators are slow, do not scale easily to future process technologies and result in large area and power consumption. Robust operation of these designs in an environment of dynamic voltage scaling (1.1V down to 0.5V) is extremely challenging. As a result, hardware TRNGs are typically fabricated in older process technologies and housed in the chipset, where the communication between the off-chip RNG and processor becomes vulnerable to side-channel attacks and bus snooping.

In this work we propose the first all-digital on-die TRNG generating high-entropy random numbers with a throughput of 4Gbps in 45nm CMOS technology. The proposed TRNG design uses only digital components and is tolerant to both static and dynamic (aging based) process variations and also resilient to dynamic changes in voltage and temperature.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.69

2. Meta-stability Based TRNG

A cross-coupled inverter (Fig. 1) can be driven towards a meta-stable state by forcing input/output nodes *'a'* and *'b'* to identical values. This is achieved by using a pair of pre-charge devices to initialize both nodes to '1' when the *CLK*=0. At the rising edge of

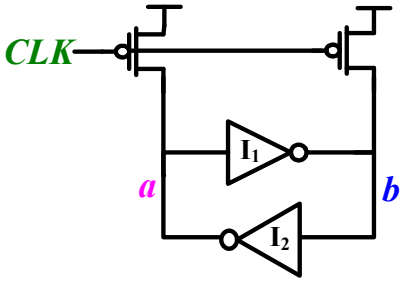


Figure 1: Bi-stable element: Entropy source

CLK, the bi-stable element enters a meta-stable state (Fig 2), where nodes 'a' and 'b' both settle to a value V_{meta}, which represents the intersection point of the VTCs of inverters I_1 and I_2. Resolution to either stable states of (*a=0,b=1*) or (*a=1,b=0*) depends on the magnitude of differential noise at *'a'* and *'b'* during the meta-stable period. A random noise source such as thermal noise at nodes *'a'* and *'b'* will result in a random resolution state during each evaluation phase of the clock. Thus a random bit is generated every cycle. This behavior of the bi-stable element forms the basis

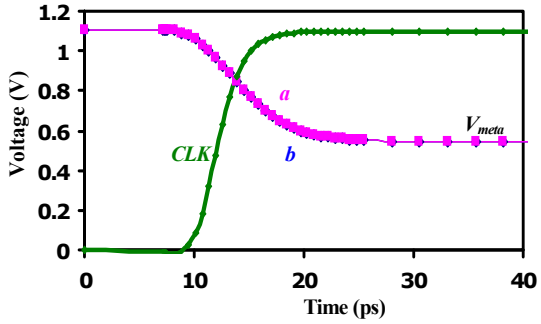


Figure 2: Bi-stable element in meta-stable state

of the proposed digital TRNG. The meta-stable state of the system can be disturbed by two events: (i) Noise on nodes *'a'* and '*b'* (ii) Device mismatch in the cross coupled inverters I_1 and I_2. Figure 3 shows the behavior of the bi-stable element in the presence of noise. Thermal noise magnitudes up to 3mV on nodes *'a'* and *'b'* quickly push the system out of meta-stability. As a result the element settles into the stable state of (*a=1, b=0*). A device mismatch in the cross-coupled inverters can introduce an intrinsic bias that will always push the

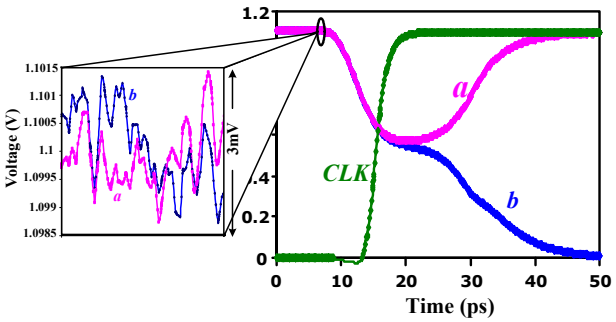


Figure 3: Bistable element behavior with thermal noise

bi-stable element in the direction of this bias, as it comes out of meta-stability. 1% mismatch in the PMOS device (Fig. 4) introduces a static bias that pushes inverter I_1 to resolve towards 1. As a result, even in the absence of thermal noise, the system will always resolve to the state (*a=0, b=1*), disrupting the random behavior of the system. Thermal noise magnitude will now have to be large enough to overcome the intrinsic bias caused due to device mismatch. This imbalance can significantly impact entropy of the generated bits. It should be noted that a systematic drift in PMOS/NMOS strengths may change the absolute values of inverter P/N skew, and impact the meta-stable point (V_{meta}). However, as long the relative P/N skews of I_1 and I_2 match each other, the behavior of the bi-stable element remains unchanged.

In a matched bi-stable element, a perturbation in the supply voltage presents itself as common-mode noise at nodes *'a'* and *'b'*. Since the meta-stable behavior of the

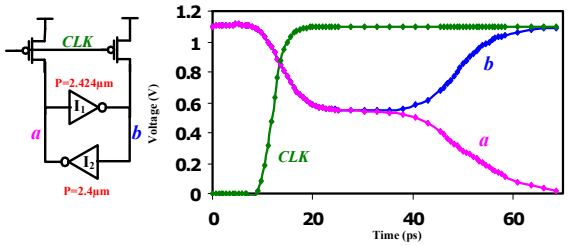


Figure 4: Bi-stable element behavior with 1% device mismatch, no thermal noise

cross-coupled system is affected only by differential noise, supply bounce should not affect the working of the system. Figure 5 shows the impact of 10% power supply noise injected into a matched system, while nodes 'a' and 'b' are at their meta-stable values. Supply noise affects the absolute value of V_{meta}, but does not tip the element out of meta-stability. However, in the presence of mismatch between the inverter devices, common-mode supply noise at V_{cc} will propagate differential components to nodes *'a'* and *'b'*, resulting in the disruption of the system from the meta-stable point. This undesired behavior can be avoided by

minimizing the mismatch between the cross-coupled inverters of the bi-stable element.

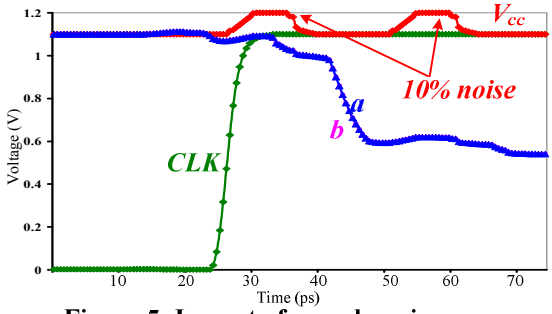


Figure 5: Impact of supply noise on a balanced meta-stable element

Thus we see that in order to accurately harness the entropy of thermal noise using a bi-stable element, we have to provide a mechanism for countering static and dynamic mismatches in the cross-coupled inverters. This would ensure that the resolution state of the system is solely governed by differential thermal noise at the two nodes, resulting in a random series of bits at nodes *'a'* and *'b'*.

3. PVT Tolerant All-Digital TRNG

A bi-stable cross-coupled inverter structure (Fig. 6) is used as the entropy harvester in the proposed all-digital TRNG. During pre-charge phase, a pair of pre-charge PMOS devices forces the system into a meta-stable state. A common NMOS transistor is used as a footer device to eliminate the short-circuit current that may exist due to the conducting NMOS devices during pre-charge. Successful generation of high-entropy random bits from such a system would require tuning mechanisms to counter mismatches between the inverters I_1 and I_2. Two techniques are introduced to handle mismatches that can disrupt the symmetry in the bi-stable element:

A. Coarse-grained Tuning using Programmable Inverters:

Inverters I_1 and I_2 are converted to programmable inverters with 8 digital configuration bits, *pconf[3:0]* and *nconf[3:0]* that control effective drive strengths of the PMOS and NMOS transistors respectively (Fig. 7). These scannable bits are used to conditionally turn ON parallel PMOS/NMOS legs, thereby controlling the P/N skew of the inverter. An increase in NMOS device

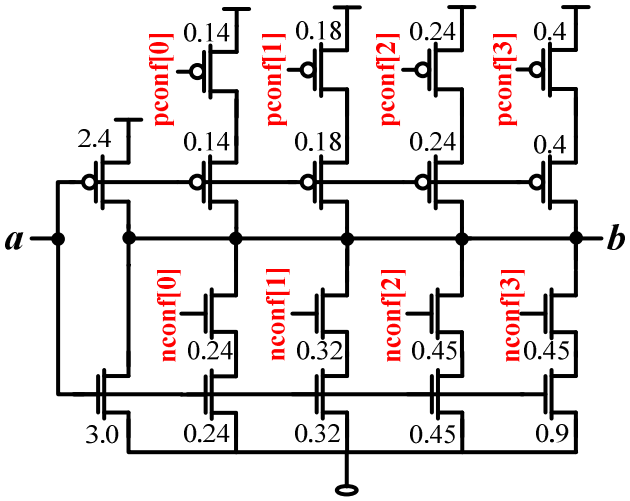


Figure 7: Programmable Inverter with 8 configuration bits

strength in inverter I_1 can be countered by turning ON an appropriate number of parallel NMOS legs in inverter I_2. The legs are weighted to tune out device mismatches in a range of 0-19% with a granularity of ~1% (Table I). This provides a coarse-grained knob for tuning out imbalances in the bi-stable element caused due to device width variation. The configurable inverters can also be programmed to tune out drive strength imbalances caused due to V_t mismatches. V_t variations result in a shift in inverter P/N skew from the

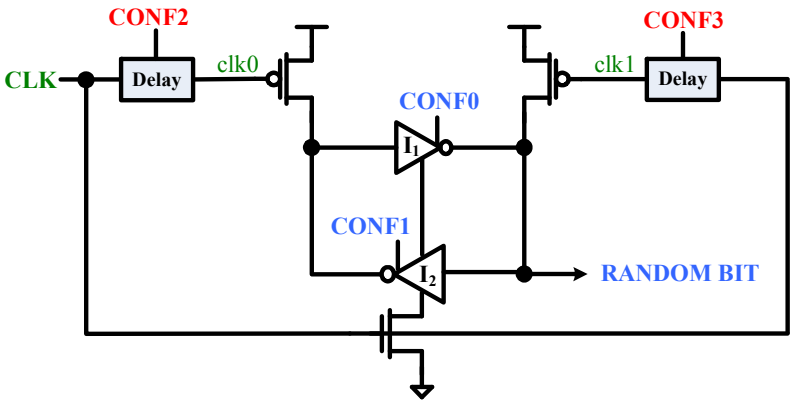


Figure 6: All-digital TRNG Organization

nominal designed value. This shift can be countered using either NMOS or PMOS configuration bits. Choosing the appropriate value of *nconf[3:0]* enables tuning out up to 158mV V_t mismatch (Table II). *Pconf[3:0]* can also be used to compensate for V_t

Table I: Tuning out device size mismatch

conf[3:0]	% NMOS size mismatch	% PMOS size mismatch
0000	0	19
0001	1	18
0010	3	17
0011	4	16
0100	5	14
0101	6	13
0110	7	12
0111	8	11
1000	10	8
1001	11	7
1010	12	6
1011	13	5
1100	14	3
1101	15	2
1110	16	1
1111	17	0

mismatches up to 24mV at a finer granularity than *nconf[3:0]*. Thus with the right combination of *pconf[3:0]* and *nconf[3:0]* a large range of process variation induced imbalances can be countered.

B. Fine-grained Tuning using delayed clocks:

A finer granularity of tuning can be achieved by separating out the pre-charge clocks and introducing a tunable skew between them using programmable delay generators (Fig. 8). By controlling the pre-charge release times for nodes 'a' and 'b', relative delay is introduced between the start of evaluation phase at each node, thereby injecting a directional bias on one side of the bi-stable element. This bias may be used to counter inherent voltage/temperature biases present in the system. Configuration bits *conf[3:0]* in the programmable clock delay element (Fig. 9) are scanned in through the scan-chain to control the PMOS drive strengths of the clock inverter, providing a tunable rising clock edge. This tuning mechanism provides finer granularity of control, enabling mismatch compensation as small as 0.05% in NMOS devices and 0.35% in PMOS devices. It will be demonstrated in section 4 that this level of granularity is sufficient to effectively tune out all residual mismatches, enhancing the sensitivity of the system to differential thermal noise.

Table II: Tuning out V_t mismatch using nconf

nconf	NMOS Vt mismatch (mV)	PMOS Vt mismatch (mV)
0001	1.5	7.8
0010	2.7	13.8
0011	3.1	23.8
0100	4.0	31.8
0101	6.0	41.8
0110	8.0	47.8
0111	10.0	57.8
1000	11.0	81.8
1001	12.0	91.8
1010	14.0	97.8
1011	14.5	107.8
1100	16.0	117.8
1101	18.0	137.8
1110	20.0	147.8
1111	21.0	157.8

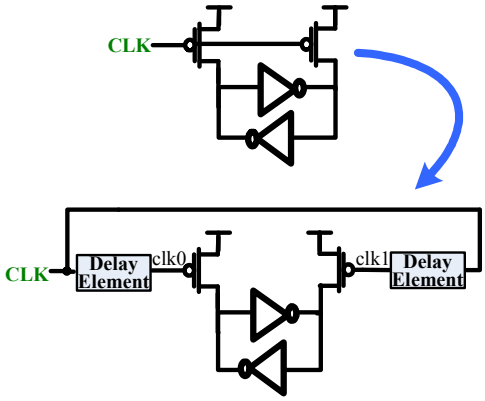


Figure 8: Separate pre-charge clocks for finer control

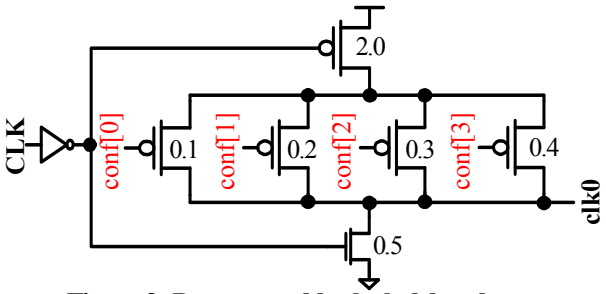


Figure 9: Programmable clock delay element

4. 45nm CMOS Implementation Results

In a 1.1V, 45nm CMOS process [10] the TRNG operates at 4GHz and generates 1 random bit each cycle, resulting in a throughput of 4Gbps, and total power consumption of 2.26mW, with leakage component of 0.3mW. RMS thermal noise at the inverter nodes is ~1.5mV. Six RNG tests (Table III) from the NIST suite [11] were applied to a bit-stream of 500 consecutive bits generated by the TRNG. Each test generates a *P-value*, an indicator of bit-stream entropy. A threshold of *P-value>0.01* is the essential pass criteria for a test. A frequency test *P-value*=1 indicates a bit-stream with equal number of 1's and 0's.

Table III: NIST Test Suite

Frequency test	Measures ratio of 1's vs. 0's in bitstream
Block frequency test	Frequency test at a block level
FFT	Examines periodicity of sequence
Long runs	Flags long runs of 0's or 1's
Cumulative sums	Computes sub-sequence partial sums
ACF Plot	Bitstream autocorrelation for lags 0-200

Table IV shows the *p-values* for the test suite in a perfectly balanced TRNG (0% mismatch) at 1.1V, 110°C. These results indicate that the magnitude and entropy of raw thermal noise at the inverter nodes, with the device sizes shown in Fig. 7, is high enough to generate a random bitstream that easily passes all NIST tests. As explained earlier, the presence of mismatch can reduce the sensitivity of the design to this thermal noise. A 2-step tuning mechanism is used to eliminate the effect of any process-variation-induced imbalance in the system. In the first step, the coarse-grained control is adjusted by scanning in 16 configuration bits (CONF0 and CONF1 in Fig. 6). Figure 10 shows the effect of coarse-grained tuning on the frequency test results in the presence of NMOS device mismatch ranging from 0% to 17%. Effective mismatches in V_t, transistor width/length, via/line resistance are simulated by varying the NMOS device size on inverter I_1 (or I_2) from its nominal value of 3µm to a max value of 3.51µm at a granularity of 0.05%. The programmable inverter I_2 (or I_1) is then configured to counter this mismatch. Coarse-grained tuning successfully counterbalances all mismatches that fall in the neighborhood of settings in Table I. This condition is represented by the peaks in *p-value* in Figure 10. It may also be observed that though the granularity of

Table IV: Test Results with 0% mismatch, 1.1V, 110°C

Frequency test	*Pvalue* = 1.00	**Pass**
Block frequency test	*Pvalue* = 0.608	**Pass**
FFT	*Pvalue* = 0.746	**Pass**
Long runs	*Pvalue* = 0.676	**Pass**
Cumulative sums	*Pvalue* = 0.629/0.38	**Pass**

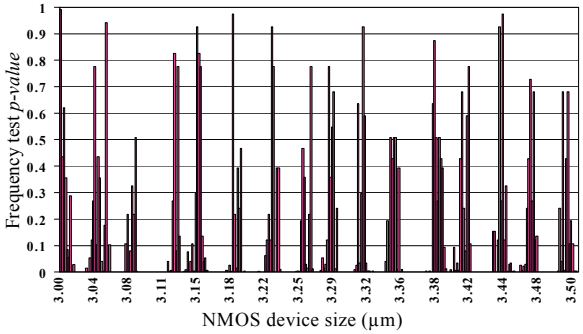


Fig 10: Coarse-grained tuning to counter device mismatch

adjustment in this step is ~1%, we obtain passing *p-values* for mismatches that fall outside this window. This is due to the fact that the magnitude of thermal noise is large enough to independently overcome up to 0.4% imbalance in device sizes. Mismatches greater than 0.4% result in failing *p-values*. These mismatches require the fine-grain control provided by the clock delay generator. In the 2nd step of tuning, the relative skews of pre-charge clocks *clk0* and *clk1* (Fig. 8) is adjusted. At the end of this step, devices can be balanced to within 0.05% of perfect balance, well within the range of thermal noise sensitivity. Mismatches that caused the frequency test to fail in the first step of tuning are handled by scanning in the appropriate clock configuration bits (CONF2 and CONF3 in Fig. 6). At the end of the second tuning step, mismatches have been minimized to the extent that random thermal noise injects sufficient entropy into the

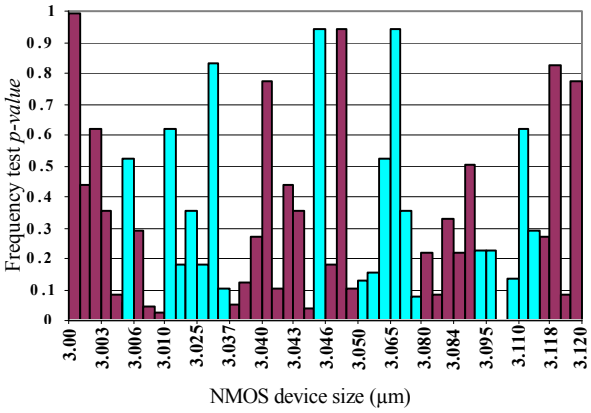


Fig 11: Fine-grained tuning using clock delay generator

Table V: NIST Test Results with 20% mismatch

Operating Conditions	**Vcc=1.1V, T=110°C**	
Configuration	**Nconf0=0000, Nconf1=1111 Pconf0=0000, Pconf1=0000 Clkconf0=0010, Clkconf1=1100**	
Frequency test	*Pvalue* = 0.33	**Pass**
Block Frequency test	*Pvalue* = 0.38	**Pass**
FFT test	*Pvalue* = 0.88	**Pass**
Long Runs test	*Pvalue* = 0.96	**Pass**
Cumulative Sums test	*Pvalue* = 0.48/ 0.30	**Pass**

bit-stream (Fig. 11). Similar results are obtained for all NIST tests on a run of 500 consecutive bits obtained from the TRNG with a worst-case mismatch of 20% on the NMOS device in inverter I_1. (Table V). The 2-step

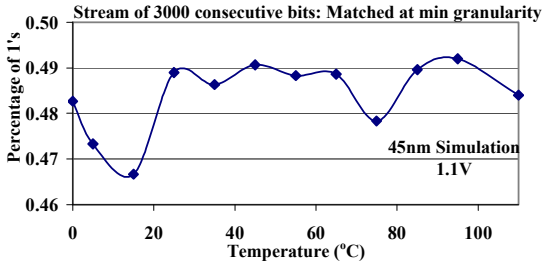


Fig. 12: Effect of temperature on randomness

tuning process minimizes effective device mismatch to a small enough magnitude such that any remnant imbalance is overcome by thermal noise. In this condition, the system is also immune to common mode noise events like power supply noise or temperature variations (Fig. 12). The all-digital nature of the TRNG results in good V_{cc} scaling behavior. Sensitivity of the RNG to thermal noise increases at low voltage,

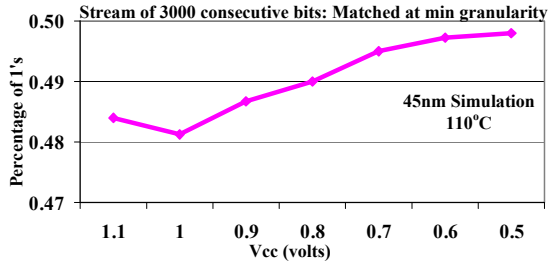


Fig. 13: Impact of V_{cc} on randomness

resulting in probabilities approaching ideal values as supply approaches 0.5V (Fig. 13). NIST autocorrelation tests at different voltages and temperatures with 20% mismatch show zero correlation with 95% confidence for up to 200 lags (Fig. 14). This confirms the

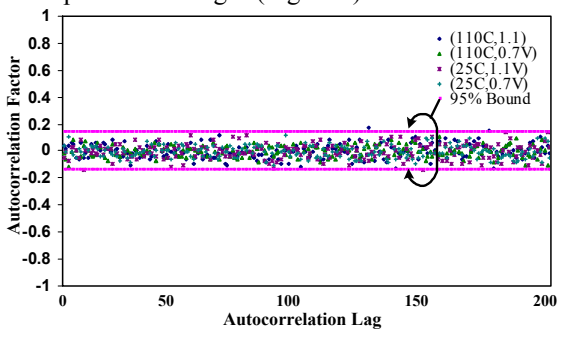


Fig. 14: Effect of Voltage/Temperature on Autocorrelation

robustness of this design to common-mode events like voltage/temperature variations.

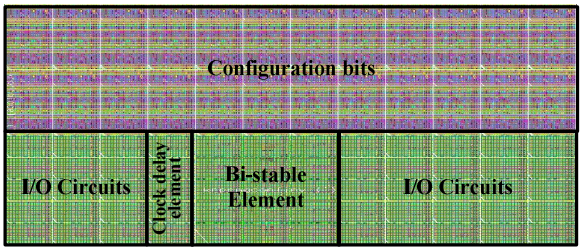


Fig 15: 45nm All-digital TRNG layout

5. Summary and Conclusion

An all-digital TRNG targeted for an on-die encryption engine with high tolerance to PVT variations is implemented in 45nm CMOS technology. A 2-step coarse/fine tuning mechanism provides tolerance to 20% mismatch in device characteristics, enabling 4Gbps throughput, generating an output bit-stream that passes NIST RNG tests with entropy/bit of 0.94. The use of 100% digital components ensures a compact layout measuring 1024μm^2 (Fig. 15) with a low energy consumption of 0.57pJ/bit. The design shows good V_{cc} scaling behavior with entropies approaching ideal values at supply voltages around 0.5V and good robustness to temperature and voltage variations.

6. Acknowledgments

The authors thank C. Dike for discussions and valuable feedback and M. Haycock, G. Taylor, S. Borkar and J. Schutz for encouragement and support.

7. References

[1] Z.Gutterman et al, "Analysis of the Linux RNG", *IEEE Symposium on Security and Privacy*, pp 371-385, May 2006.

[2] R. Brederlow, et al, "A low-power TRNG using random telegraph noise of single oxide-traps", *ISSCC Dig. Tech. Papers*, pp. 536-537,Feb., 2006.

[3] B. Jun and P. Krocher, "The Intel RNG", White Paper, *http://www.cryptography.com/intelRNG.pdf*, 1999

[4] C. Petrie and J. Connelly, "Noise-based IC RNG for applications in cryptography," *IEEE Trans. Circuits & Systems-I*, vol. 47, no. 5, pp.615-621, May 2000

[5] M. Bucci, L. Germani, R. Luzzi et al., "A high-speed oscillator-based truly random number source for cryptographic applications on smart card IC," *IEEE Trans. on Computers*, vol. 52, pp. 403-409, April 2003.

[6] S. Fujita, et al., "Si nanodevices for RNG circuits for cryptographic security", *ISSCC Dig. Tech. Papers*, pp. 294-295, Feb. 2004.

[7] D. Kinniment and E. Chester, "Design of an on-chip random number generator using metastability," *Proc. ESSCIRC.*, pp. 595-598, Sep., 2002.

[8] C. Tokunaga, et al, "TRNG with a metastability-based quality control," *ISSCC Dig. Tech. Papers*, pp. 404-405, Feb. 2007.

[9] Mari Matsumoto et al, "1200μm^2 Physical RNG based on SiN MOSFET for Secure Smart-Card Application", *ISSCC Dig. Tech. Papers*, pp. 414-415, Feb. 2008.

[10] K. Mistry et al., A 45nm Logic Technology with High-k+Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193nm Dry Patterning, and 100% Pb-free Packaging, *Proc. IEEE IEDM*, Dec. 2007, pp. 247-250.

[11] National Institute of Standards and Technology, "A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications," *Pub 800-22*, 2001

Single Ended Static Random Access Memory for Low-V_{dd}, High-Speed Embedded Systems

Jawar Singh, Jimson Mathew, Saraju P. Mohanty* and Dhiraj K. Pradhan
Department of Computer Science, University of Bristol, UK.
Department of Computer Science and Engineering, University of North Texas, USA.*
email-ID: jawar@cs.bris.ac.uk, saraju.mohanty@unt.edu

***Abstract*— Single-ended static random access memory (SE-SRAM) is well known for their tremendous potential of low active power and leakage dissipations. In this paper, we present a novel six-transistor (6T) SE-SRAM bitcell for low-V_{dd} and high-speed embedded applications with significant improvement in their power, performance and stability under process variations. The proposed design has a strong $2.65\times$ *worst case* read static noise margin (SNM) compared to a standard 6T SRAM. A strong write-ability of logic 'one' is achieved, which is problematic in SE-SRAM cells even at lower voltage. The proposed bitcell design is mainly targeted for word-organized SRAMs. A $16 \times 16 \times 32$ bit SRAM with proposed and standard 6T bitcells is simulated (including parasitics) for $65nm$ CMOS technology to evaluate and compare the different performance parameters, such as, read SNM, write-ability, access delay and power. The dynamic and leakage power dissipation in the proposed 6T design is reduced by 28% and 21%, respectively, as compared to standard 6T design.**

I. Introduction

Embedded systems particularly targeted towards low duty-cycles and portable applications such as mobile phone or PDAs require extremely low energy consumption as they are often battery powered. In such systems, a significant amount of power is consumed during memory accesses which determines the battery life. Hence, efficient active and leakage power saving SRAM designs need to be explored for higher reliable and longer operation of battery powered applications. There are mainly two areas with strong potential of active power saving: (a) reduction in charging capacitance or static current by partial activation of multi-divided word and bit lines and (b) lowering operating voltage resulting from external power supply reduction and half-V_{dd} precharging [10]. In [13], 30% to 70% of the total active power is dissipated in bit lines charging and discharging during read and write operation. Hence, reduction in charging capacitance or static current has strong prospect of active power saving. *In the proposed design we have exploited this fact to reduce the active power despite of full-V_{dd} precharging of the bitline*. The precharging of bitline to full-V_{dd} is mainly to achieve strong write-ability of logic 'one' into the bitcell which makes easier to operate the SRAM at lower V_{dd}.

Lowering supply voltage to reduce power (energy) consumption is one of the first choice of designers for ultra-low-power applications. However, ultra-low-power design of high-density SRAMs in which the operating voltage is below the transistor threshold voltage is extremely challenging. This is due to reduced static noise margin (SNM) and increased variability in design and process parameters in the nanoscale CMOS (nano-CMOS) technology. As we move from $130nm$ to $65nm$ technology node, the area occupied by the memory increases from 71% to 82% [1]. In modern system on chips (SoCs) when total power and total area is dominated by the SRAM, reduction in V_{dd} for SRAMs can save both active energy and leakage power [9]. Also for system integration, SRAM must be compatible with subthreshold combinational logic operating at ultra-low voltages [14]. However, this leads to increase in sensitivity of design and process parameter variability. This problem will worsen in nanometer technologies with ultra-low voltage operation and makes SRAM design and stability analysis more challenging. These practical challenges limit standard 6T SRAM bitcells and architectures to higher V_{dd}. A standard 6T SRAM bitcell in $65nm$ CMOS technology is shown in Fig. 1 (a) [11]. The data storage node Q and QB in standard 6T bitcell are most vulnerable to capacitive coupling noise due to bitlines (BL and BLB) and voltage division effect between access transistors and pulldown transistors. A proper sizing of these transistors is important to maintain data stability and functionality as shown in Fig. 1(a).

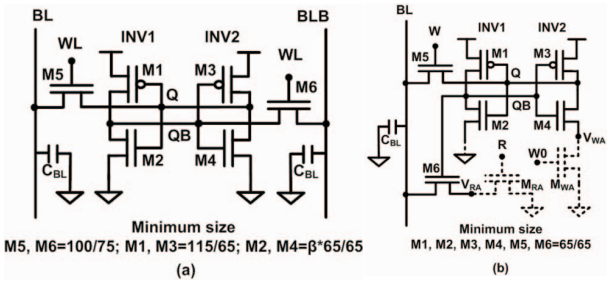


Fig. 1. The proposed single-ended 6T SRAM cell with dotted read and write assist transistors shown in (b) with respect to standard 6T SRAM cell shown in (a).

This paper introduces a 6T bitcell and its word-organization for robust and high density SRAMs in the subthreshold regime. In proposed 6T SEIO bitcell: 1) read current path is isolated from the data storage node Q and QB hence, less vulnerable to noise; 2) isolation of read current path improves the read SNM $2\times$ compared to standard 6T bitcell with $\beta = 2$ and at

[0]This research is supported in part by NSF award number 0702361.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.38

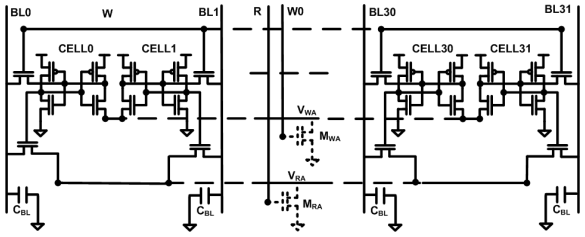


Fig. 2. A 32-bit word organization of the proposed 6T SE-SRAM cell with dotted read and write assist transistors.

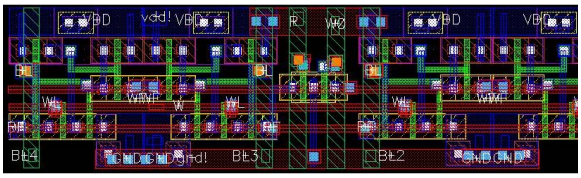


Fig. 3. Layout of the proposed word-organized 6T SRAM bitcell with four bitcells and read/write assist transistors in the middle.

V_{dd} = $0.2V$ and $1.0V$; 3) process variation degrade the read SNM of proposed 6T and standard 6T SRAM bitcells by up to 13% and 50% respectively thereby, $2.65\times$ tolerance to process variability. A model for determining the size of the read/write assist transistors is developed for estimation of read access delay with accuracy of up 95%. The dynamic and leakage power dissipation in the proposed 6T design is reduced by 28% and 21%, respectively, as compared to its counterpart design.

The rest of the paper is organized as follows: Section II introduces the proposed bitcell and word-organized SRAM design. In Section III, statistical analysis of parametric failures is presented. Read and write assist transistors sizing issues are discussed in Section IV. In Section V, dynamic and leakage power of the standard and proposed designs are compared. Section VI provides a summery of the key conclusions.

II. A PROPOSED SEIO 6T SRAM BITCELL DESIGN

Fig. 1 (b) shows the proposed single ended input/output 6T SRAM bitcell schematic with minimum feature sized transistors for a 65nm CMOS technology. The proposed 6T SRAM bitcell consists of a cross coupled inverter pair (INV1 and INV2) connected to a bitline (BL) using access transistor (M5) and a storage node isolation transistors (M6). The dotted transistors in the figure (M_{WA} and M_{RA}) represent read and write assist transistors, respectively, for a memory word. A memory word can be 8, 16, or 32 bit. Three control signals W, its complement W0 and R are used for controlling the write and read operations. The write operation is controlled by W and $W0$. These signals are respectively connected to M5 and M_{WA}. While read operation is controlled by R which is connected to M_{RA}.

In the following, we illustrate the word-organized SARM design architecture with proposed bitcell. Let, n be the number of cells in a word-organized memory which contains more than 1-bit per word, that is, $n \geq 2$. For instance, the word-organization of the proposed 6T SRAM bitcell for $n = 32$, is shown in Fig. 2. Since read and write operations access the n bits of a word simultaneously, one could share the read/write assist transistors of a bitcell as shown dotted in Fig. 1(b). Therefore, we need only one read/write assist transistor per word. Consequently, each bitcell in a word consists of six transistors with two additional dotted transistors per word (Fig. 2). Sizing issues of these shared (dotted) transistors are explained in Section IV.

Fig. 3 shows the layout of the proposed word-organized 6T SRAM bitcell with four bitcells and read/write assist transistors. We present only four cells for clarity. The propped bitcell layout area is $0.68\mu m^2$ ($0.55\mu m \times 1.22\mu m$), which is 8% higher (because of additional contacts) than the standard 6T SRAM bitcell for $\beta = 2$. While, read/write assist transistors occupies merely half of the bitcell area per word. We have used three metal layers (M1, M2 and M3). Metal layer M1 is used for routing the supply rails (V_{dd} and G_{nd}), M2 is used for routing the shared contacts among bitcells, read and write signals. While, M3 is used for routing the bitlines. The design has been successfully laid-out for different word sizes. Parasitic were extracted and included in a SPICE deck for simulation results presented in this paper.

A. Read Operation

Information read out from the proposed SRAM bitcell is carried out via single ended bitline (data-line). Prior to read operation, BL is precharged to V_{dd} and the read signal (R) is asserted high (W is low) to turn on the M_{RA}, which is essentially applicable for reading '0'. For reading '1', BL has to remains at precharged level ($\sim V_{dd}$) because transistor M6 is turned off. It is important to notice that only the read '0', high to low transition is affected by the insertion of the M_{RA} and that the read '1', low to high transition will not be affected. As a result, reading '1' is directly sensed from the precharged BL. In both the cases either reading '1' or '0', storage nodes are isolated from the read current path. It results reduced capacitive coupling noise due to BL and hence, significantly enhancing the data stability during read and hold state. Also compared to standard 6T bitcell the read current path has equal number (two) of series connected transistors with minimum feature size resulting in better performance of proposed 6T bitcell.

Read static noise margin (SNM) of the proposed 6T and standard 6T SRAM bitcells are shown in Fig. 4 for a comparative perspective. The proposed 6T bitcell has an SNM of $0.302V$, while the standard 6T bitcell SNM is $0.152V$ at a supply voltage of $1.0V$ and $\beta = 2$ (Fig. 4(a)). The SNM of the proposed 6T bitcell at a supply voltage of $0.3V$ is equal to that of the standard 6T bitcell at $0.5V$ and $\beta = 4$ (Fig. 4(b)). However, the SNM normalized to supply voltage for different

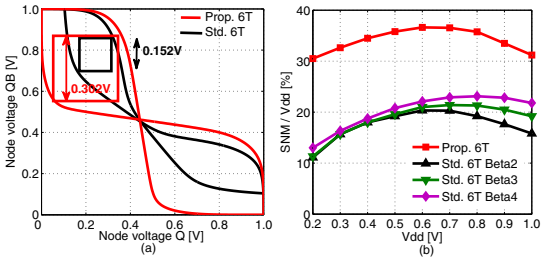


Fig. 4. SNM comparison of standard SRAM and proposed SRAM cell during a read operation at $V_{dd} = 1V$ in Fig. (a). SNM normalized to supply voltage for different cell ratio ($\beta = 2$, 3 and 4) is shown in Fig. (b).

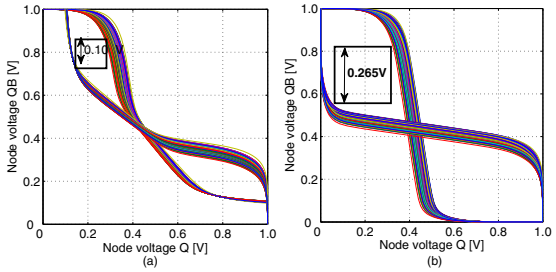


Fig. 5. Monte Carlo simulation of voltage transfer characteristics (VTCs) shown with worst case SNM during read operation under process variations: (a) for standard SRAM and (b) for proposed SRAM bitcell.

cell ratio (β= 2, 3 and 4) in Fig. 4(b) shows that the variation of SNM in the proposed 6T bitcell (for minimum feature size) is smaller than that of the standard 6T bitcell, which is mainly because of reduced capacitive coupling noise due to BL and isolation of read current path from the storage node Q and QB.

B. Write operation

It is well known that the write operation in single ended SRAM cell is difficult because of strongly cross coupled inverters. A write assist transistor M_{WA} is used to alleviate this problem, which is controlled by W0 for a successful write operation. The usage of M_{WA} is to weaken the cross coupling of proposed 6T SRAM bitcell inverters during write access time.

Initially assume that the node Q= 0 and QB= 1, we need to change these node states. In write mode, write signal (W) is asserted high to turn on the write access transistor M5 that connects the precharged bit line to node Q. As both the inverters (INV1 and INV2) are strongly cross coupled so forcing the node Q to '1' is difficult through an NMOS (M5) pass device. Hence, we weaken the pull down strength of INV2 by inserting a series transistor M_{WA}, which is controlled by a complement of write signal W0 to turned off during write operation. In other words, M_{WA} is used to weaken the strongly cross coupled inverters.

The timing waveforms of read and write control signals (R and W), input and output data (Data-Write and Data read), and bitcell node Q are shown in Fig. 6. While the timing

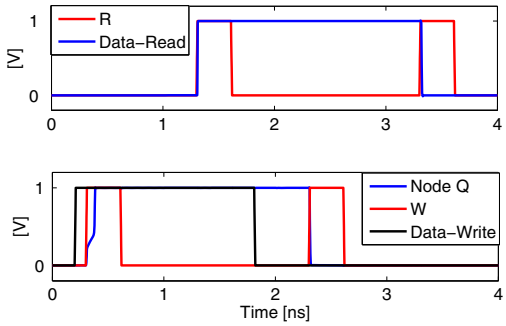


Fig. 6. Timing simulation waveforms for write and read operations of proposed 6T bitcell.

waveforms of clock, decode, precharge, and sense stage signals are not shown. One can observed that the information has been effectively written and readout from the proposed word-organized 6T SRAM bitcell design.

III. Statistical Analysis of Parametric Failures

The variations in threshold voltage of an SRAM cell transistors due to random dopant fluctuations is the principal reason for parametric failures [4]. Parametric failures in standard 6T SRAM bitcell can occur due to (a) destructive read (cell may flip when access for read), (b) un-successful write i.e., bitcell cannot be written within the write access time, which is measured in terms of trip voltage of an inverter, and (c) read access failure i.e., incorrect read operation, which is a strong determinant of performance and power of the SRAM. For parametric failure analysis, we assume a 15% variation in V_{th} with 3σ as an independent random variable for all the transistors in SRAM cell with a Gaussian distribution.

A. Destructive read

Data retention of the 6T SRAM bitcell during the read and hold operation is an important functional constraint, which is measured in terms of read and hold SNM. The SNM is a widely used metric for stability analysis of an SRAM bitcell usually defined as the maximum value of dc noise voltage (V_n) that can be tolerated by the SRAM cell without flipping the node states. During the read operation, voltage at node QB ($= 0$) is most vulnerable to noise due to potential divider action in read current path of M5 and M2 to a positive value of V_n. If V_n is higher than the trip voltage of the INV2, then the cell flips resulting destructive read failure. In the proposed 6T SRAM bitcell the nodes (Q and QB) are is isolated from the read current path to circumvent the noise vulnerability. Process variations in V_{th} degrade the read SNM of standard 6T and proposed 6T SRAM cell by up to 50% and 13% respectively compared to nominal design corner as shown in Fig. 5. The proposed 6T SRAM bitcell provide $2.65X$ higher worst-case read SNM as compared to the standard 6T SRAM bitcell under same process variations. *Thus, the proposed 6T bitcell has better noise margin, worst-case read stability and process variation tolerant.*

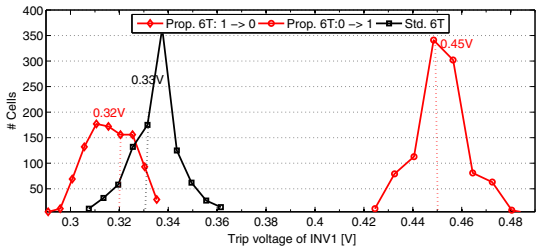


Fig. 7. Monte Carlo simulation of write trip voltage of the standard and proposed 6T SRAM bitcell.

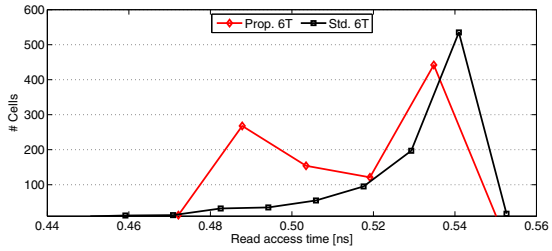


Fig. 8. Monte Carlo simulated read access time of the standard and proposed 6T SRAM bitcells.

B. Un-successful write

Write ability of a standard 6T SRAM bitcell is best characterize using write trip voltage which is defined as the maximum voltage on the bitline needed to flip the bitcell content [6]. Due to asymmetric nature of the proposed 6T SRAM bitcell, we need to analyzed both the state write '1' and '0'. In order to write '1' (Q= 1 and QB = 0) to a cell storing '0' (Q = 0 and QB = 1), low internal node Q of the cell is pulled up above the trip voltage of the INV1. Since, pull down strength of the INV2 has been weaken during write access time due to stacked transistor M_{WA}, which makes pulling up of low internal node Q above the trip voltage easier. Similarly, writing '0' (Q = 0 and QB = 1) to a cell storing '1' (Q = 1 and QB = 0), high internal node Q of the cell has to discharge via bitline (BL) well below the trip voltage of the INV1 so that the cross-coupled inverter pair starts working and the cell content gets flipped. To guarantee that a correct write operation will occur, it is important that the node Q should be pulled up (down) above (below) the trip voltage of INV1 within the write access time when W is high otherwise a write failure will occur. Under process variation, statistical analysis of write-ability shows that the mean value of the write trip voltage for writing $1 \rightarrow 0$ is $0.32V$, whereas for writing $0 \rightarrow 1$ is $0.45V$. However, mean value of write trip voltage for writing '1/0' of standard 6T bitcell is $0.33V$. The write trip voltage standard deviation due to process variations in standard and proposed 6T bitcells are almost equal of about $10mV$, as shown in Fig.7. *Thus, the write ability of the proposed bitcell has not degraded under process variation*

C. Read access failure

The bitcell read access time or critical path in SRAM memories typically determines the memory performance and ensures the correct read operation. For a successful read operation, read access time is defined as the time required to produce a pre-specified voltage difference between two bit lines of a standard 6T SRAM bitcell [3], [12]. In proposed 6T SRAM bitcell the critical read access time correspond to reading '0', which determines the performance of the proposed bitcell. Since '1' is directly sensed from the precharged bitline. The read access time (for '0') of the proposed bitcell is defined as the time required to produce a pre-specified voltage difference between reference and single bitline voltage. Statistical read access time distribution of standard and proposed 6T SRAM bitcells are shown in Fig. 8. Under process variation, mean value of the read access time of standard 6T bitcell is $0.53ns$, which is 4% higher ($0.51ns$) than the proposed 6T bitcell. While, standard deviation in read access time of standard 6T bitcell ($0.020ns$) which is 14% higher ($0.017ns$) than the proposed 6T bitcell. *Thus, the proposed cell has better process variation acceptance than the standard 6T bitcell.*

IV. Sizing of Read and Write Assist Transistors

Proper sizing of read/write assist transistor is very crucial because whole functioning and performance of a memory block depends on these transistors. If we overestimate their size, then there is a wastage of valuable silicon area and increase of switching power dissipation because of larger loading. Similarly, if we underestimate the size, then the read and write operations would be too slow because significant delay due to the increased resistance to ground. Usage of both the transistors is fundamentally different because one (read assist) transistor has to provide low resistive path to read current during read operation. On the other hand (write assist) transistor has to provide high resistance path for successful write operation to weaken the cross coupling of bitcell inverters. As both read and write requirements are conflicting in nature, so we need to analyze the sizing issues separately for read and write assist transistors.

A. Sizing of read assist transistor

As we have seen in Section III, the read assist transistor forms the critical path, essentially when reading '0' from the proposed bitcell. Hence, performance of the proposed SRAM is determined by the '0' read access time, which is mainly dependent on the size of M_{RA}. Consequently, size of the M_{RA} in word-organized SRAM design when a word has common read assist transistor (M_{RA}) is critical for proper functioning of SRAM. We have developed a simple model to determine the minimum size of M_{RA} and corresponding '0' read access delay for a single cell, which is extended for proposed word-organized SRAM design. The proposed model is inspired by well-established power gating techniques in which sleep transistor is used to gate the power supply [7]. In the literature [7], [8], it was shown that the sleep transistor can be approximated as a linear resistor to create a virtual ground because $V_{ds} < (V_{gs} - V_{th})$ of sleep transistor. Here,

this sleep transistor is referred as read assist transistor (M_{RA}). The amount of current flowing through the linearly-operating M_{RA} transistor can be approximated as [5]:

$$I_{RA} \approx \mu_n C_{ox} \left(\frac{W}{L}\right)_{RA} (V_{dd} - V_{th}) V_{RA}, \quad (1)$$

where μ_n is the mobility of electrons, C_{ox} is the oxide capacitance and V_{th} is the threshold voltage. Since, the M_{RA} is approximated as linear resistor and operating in a linear region, then the M_{RA} resistance $R_{RA} \approx \frac{V_{RA}}{I_{RA}}$. Thus, the size of the read assist transistor can be expressed as:

$$\left(\frac{W}{L}\right)_{RA} = \frac{1}{R_{RA}\mu_n C_{ox}(V_{dd} - V_{th})}. \quad (2)$$

If R_{RA} is known, then the size of the read assist transistor $(W/L)_{RA}$ can be determined by using the above expression 2. The M_{RA} affects only high to low transition or reading '0' to discharge the precharged bitline. Since, bitline capacitance C_{BL} is discharging, and neglecting the node V_{RA} parasitic capacitance, any charge flowing out of the source of M6 will flow through the read assist resistor R_{RA} of M_{RA}. This phenomenon is modeled as a *R-C circuit*, which comprises of series resistor R_{RA} and bit line capacitance C_{BL} charged at voltage V_{dd}. Thus, the relationship among these parameters can be expressed as follows:

$$V_{RA} = V_{dd} \times exp\left(\frac{-t}{\tau}\right). \quad (3)$$

Where τ is the time constant, the read sensing circuitry will detect the transition high to low i.e. read '0' only when the bit line is discharged to about 36.8% of the V_{dd} after a certain amount of delay from the assertion of read control signal, which is defined as a read access delay. Under this condition the read access delay τ_d is equal to time constant (τ):

$$\tau_d = R_{RA} C_{BL}. \quad (4)$$

In the word-organized SRAM array shown in Fig. 2, let the word is n-bit wide i.e. there are n-bitcells in each word and all are having individual M_{RA}. These individual M_{RA} of n-bitcells in a word are replaced by an equivalent M_{RA} to reduce the transistor count and silicon area overhead. The size of M_{RA} in worst case pattern (i.e. when all the n-cells having '0' at node Q) determines the read access delay or operating frequency of the SRAM. As we have approximated the M_{RA} of a cell as a linear resistor, then all the n-bitcells M_{RA} will form a parallel combination of n-linear resistors in worst case pattern. In this case, the M_{RA} resistance will be equivalent to M_{RA}/n. Similarly, n-precharged bitlines capacitance (neglecting the node capacitance) will be replaced by an equivalent capacitance nC_{BL} because of parallel combination they form. Once we have an equivalent resistance, capacitance and target read access delay then from eqns. 2- 4, we can determine the size of the M_{RA} for any word size. The SPICE simulation and estimated results for read assist transistor size (W/L) and read access delay for different word sizes (n =8, 16, 32 and 64) of the proposed word-organized SRAM designs are shown in Fig. 9. One can observe that the proposed model archives up to 95% accuracy in estimation of read access delay for different word sizes.

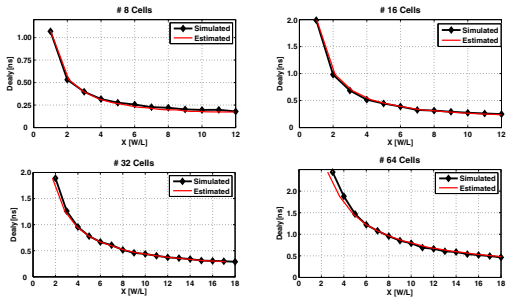


Fig. 9. Estimation of read access delay for different read assist transistor size (W/L).

B. Sizing of write assist transistor

In the proposed word-organized SRAM array, all individual SRAM bitcell's M_{WA} transistors are replaced by a single equivalent transistor (M_{WA}). Thus, M_{WA} should be sized properly so that all the cells in that word written correctly. In worst case scenario, that can be either writing '1' or '0' in all the cells. The M_{WA} has to weaken the cross coupled inverters by floating the INV2 of all the bitcells in that word. Weakening of the loop doesn't matter whether we are intended to write '1' or '0' in all or fewer cells in that word. The weakening of the loop of a single bitcell or all the bitcells in a word is equivalent because V_{ds} of M_{WA} is always higher than the '0', when V_{GS} of M_{WA} is zero. Thus, a minimum sized transistor would be well suited for this purpose. Also after the write access time M_{WA} has to provide a ground to node V_{RA} of all the bitcells. For providing a ground to node V_{RA}, M_{WA} has to provide only the leakage current path to all the bitcells either they are having '0' or '1' at node Q. Since, the transistor M_3 (when node Q at '0') and transistor M_4 (when node Q at '1') are in cutoff mode, therefore, there is only leakage current has to flow through M_{WA}. As M_{WA} has to provide only the leakage current path to all the bitcells of a word which will always less than the dynamic current of a transistor even when all the cells are writing either '1' or '0' simultaneously. Also, for minimum leakage and data retention it is recommended to use minimum size of transistor. The SPICE simulation for different word size of SRAM reveals that there is no significant improvement in the write-ability of the SRAM with increasing the size of M_{WA}.

V. Power Consumption

A $16 \times 16 \times 32$ bit SRAM memory with 32 bitcells in a word using standard and proposed 6T bitcell designs was simulated in SPICE, operated at a clock speed of $1GHz$ and $V_{dd} = 1V$. The simulation results are based on the BPTM of $65nm$-technology node [2]. The dynamic power consumption of a standard and proposed bitcells under different read and

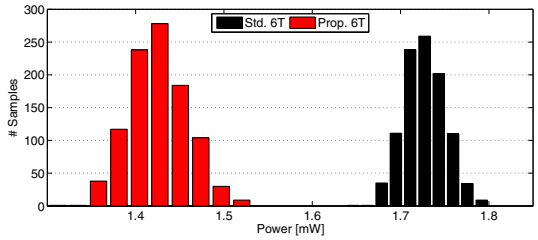


Fig. 10. Statistical distribution of leakage power for the proposed and standard SRAM.

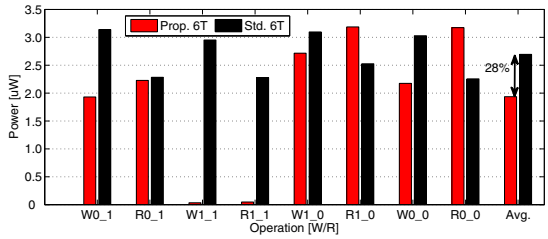


Fig. 11. Dynamic power pattern for different read/write operations of proposed and standard 6T SRAM bitcells.

write operations is shown in Fig. 11. Because proposed bitcell is asymmetric, its dynamic power consumption pattern is also asymmetric. In Fig. 11, operation W0_1 stands for writing '1' into the cell while its original content is '0'. Similarly, R1_0 stands for reading '0' from the cell, while previous output was '1'. The dynamic power consumption of the proposed bitcell under diffract combinations are quite different because of asymmetric nature. For operations W1_1 and R1_1, the dynamic power of proposed 6T bitcell is very low as compared to standard 6T bitcell, because both the operations are performed without discharging the bitline of the proposed bitcell. Under such operations precharged bitline can be used for future read/write operation. Alternatively, in standard bitcell one bitline has to discharge during these operations. However, the dynamic power for operations R1_0 and R0_0 in proposed 6T bitcell is 21% and 29% higher than the standard 6T bitcell. The average dynamic power under different read/write operations of the proposed 6T SRAM bitcell is 28% lower than the standard 6T bitcell [Fig. 11].

In 16X16X32 bit SRAM memory using proposed bitcells, reading a word "1110 1110....1110" consumes an average power of only 31% $(3.86mW)$ of the standard 6T SRAM memory array because of the reuse of the charged bitline. While, reading a word "0001 0001....0001" consumes 128% $(15.94mW)$ of the standard 6T SRAM memory. Reading a word with alternating values "1010 1010....1010" uses 68% $(8.47mW)$ of the standard 6T SRAM memory array power.

The leakage contribution pattern of the proposed bitcell is also asymmetric. When node Q$=0$, it leaks more as compared to Q$=1$ because the read current path transistor M6 is turned on. However, average leakage contribution in the proposed cell is 37% less than the standard bitcell. For total leakage in $16\times16\times32$ bit SRAM memory array (using proposed bitcells) in standby mode, when all the bitlines are charged to V_{dd}, access transistors (M5) of a word are cutoff and control signal read and write are clamped at '0'. Similarly, for standard 6T memory array bitlines are charged to V_{dd}, and control signals are clamped at '0'. The leakage power distribution under process variation for the proposed and standard SRAM array is shown in Fig. 10. The average leakage power consumption of the proposed SRAM array is $1.4mW$, which is 21% lower than the counterpart SRAM array. The standard deviation in leakage power of the proposed SRAM array is 42% higher $(32\mu W)$ than the standard SRAM array $(23\mu W)$.

VI. Conclusion

A SEIO 6T bitcell design and its word-organization for robust and high density SRAMs is presented. The immunity to process variations (robustness) and high density in the proposed design is achieved by isolating the read current path and using minimum feature size transistors. The improved read and write-ability (data stability), reduced dynamic and leakage power dissipation compared to standard 6T, makes the new approach attractive for nanoscale technology regime in which process variation is a major design constraint. Experimental results shows that the proposed design has tremendous potential for nano-CMOS SRAM design.

References

[1] International technology road map for semiconductors, test and test equipments. http://public.itrs.net/, 2006.

[2] BPTM, http://www.device.eecs.berkeley.edu/ ptm/download.html/, 2008.

[3] K. Agarwal and S. Nassif. Statistical analysis of sram cell stability. In *Proc. 43rd annual conf. Design automation*, pages 57–62, 2006.

[4] A.J.Bhavnagarwala, X. Tang, and M. J.D. The impact of intrinsic device fluctuations on cmos sram cell stability. *IEEE Journal of Solid-State Circuits*, 36:658–665, Apr 2001.

[5] M. Anis, S. Areibi, and M. Elmasry. Design and optimization of multithreshold cmos (mtcmos) circuits. *IEEE Trans. CAD of Integrated Circuits and Systems*, 22(10):1324–1342, Oct. 2003.

[6] E. Grossar, M. Stucchi, K. Maex, and W. Dehaene. Read stability and write-ability analysis of sram cells for nanometer technologies. *IEEE Journal of Solid-State Circuits*, 41(11):2577–2588, Nov 2006.

[7] J. Kao, A. Chandrakasan, and D. Antoniadis. Transistor sizing issues and tool for multi-threshold cmos technology. *Proceedings of the 34th Design Automation Conference*, pages 409–414, Jun 1997.

[8] J. Kao, S. Narendra, and A. Chandrakasan. Mtcmos hierarchical sizing based on mutual exclusive discharge patterns. In *Proceedings of the 35th annual conference on Design automation*, pages 495–500, 1998.

[9] N. S. Kim, K. Flautner, D. Blaauw, and T. Mudge. Circuit and microarchitectural techniques for reducing cache leakage power. *IEEE Trans. Very Large Scale Integr. Syst.*, 12(2):167–184, 2004.

[10] I. Kiyoo, S. Katsuro, and N. Yoshinobu. Trends in low-power ram circuit technologies. In *Proc. of the IEEE, vol. 83*, pp. 524–543, April 1995.

[11] Z. Liu and V. Kursun. Characterization of a novel nine-transistor sram cell. *IEEE Trans. VLSI Systems*, 16(4):488–492, April 2008.

[12] S. Mukhopadhyay, H. Mahmoodi, and K. Roy. Modeling and estimation of failure probability due to parameter variations in nanoscale srams for yield enhancement. In *Proc. VLSI Circuits Symposium*, pp. 64–67, 2004.

[13] L. Villa, M. Zhang, and K. Asanovic. Dynamic zero compression for cache energy reduction. In *International Symposium on Microarchitecture*, pages 214–220, 2000.

[14] A. Wang and A. Chandrakasan. A 180 mv fft processor using subthreshold circuit techniques. In *Proc.IEEE ISSCC Dig. Tech. Papers*, pages 229–293, 2004.

Session 5B

Secure VLSI Design

Encoding of Floorplans through Deterministic Perturbation

Debasri Saha and Susmita Sur-Kolay
A.C.M. Unit, Indian Statistical Institute, Kolkata, India. {debasri_r, ssk}@isical.ac.in

Abstract

Recent trends in VLSI design involve rapid growth of design reuse and electronic Intellectual Property (IP) commerce. For VLSI physical design, the risk of misappropriation of design IP stored in design repositories, or the threat of hacking the same during its web-based transmission, mandates design file encryption. However, encryption of GDSII/OASIS design files, too large in size and complex in format, is troublesome, time consuming and also prone to typical cryptanalysis. The idea of an alternate efficient approach of encoding by deterministic perturbation of design IP resulting in a degraded design of negligible IP value, is proposed here to ensure security during design storage or transmission. From the highly degraded design only authorized person can quickly regenerate the optimized design. In this paper, the technique for design encoding through perturbation is applied for floorplanning stage. Encoding moves for various floorplan representations are analyzed and a novel technique for encoding tree-based representations is proposed. Experimental results on floorplan perturbation for MCNC benchmarks are encouraging.

Keywords: Direct intellectual property protection, VLSI physical design, floorplan representations, encoding moves.

1 Introduction

With the advent of DSM VLSI technology, rapid increase in design complexity of integrated circuits, increasing market demands on shorter design cycle time encourage design reuse. The circuit components in a design, available in electronic form and known as virtual components, signify intellectual property (IP) of the design. However, infringement of design IP is quite likely. Hence intellectual property protection (IPP) of VLSI design is a prime concern.

Techniques to protect IP of VLSI design can be of two types – direct and indirect. Direct IP Protection includes proper protection of design repositories, secure web transmission of design IP during electronic design commerce between two parties, whereas embedding of watermarks and fingerprints into the design for future establishment of identity of legal owner and buyer of design is covered under indirect IPP. In this paper, we concentrate only on direct IP protection of VLSI physical design. An idea of protection through deterministic degradation of the design, based on a certain key K is proposed so that only an authorized person can recover the original optimized design from the perturbed one using the same private key K. For a design developed through multi-objective combinatorial optimization, the perturbed version is of negligible IP value, and if intercepted, is of no use. Generating optimized design from the degraded one is equivalent to starting from scratch. Here, a novel technique for encoding floorplans through perturbation is developed.

This paper is organized as follows. Previous works are outlined in Section 2. Section 3 analyzes existing floorplan representations and various floorplan moves for design IP encoding. A novel binary tree encoding scheme is given in Section 4. The proposed technique of floorplan encoding through degradation and its analysis are presented in Section 5. Experimental results on benchmarks appear in Section 6 and concluding remarks in Section 7.

2 Previous Works and Motivation

The IPP technique, discussed in [1], emphasizes on cryptographic encryption for ensuring direct IPP. In [2], an encryption protocol for secure transmission of FPGA design is proposed but it is hardware-supported and applicable to FPGAs only. A design layout file either in GDSII or OASIS format contains repetitions and is large in size (tens of GB) leading to high encryption time and huge transmission overhead. So GDSII compression using GDSIIzip or GDSII optimization using Bantam software (timing requirement 1min/GB) is essential prior to its encryption. Moreover, physical design in any intermediate stage cannot be protected using standard encryption techniques unless it is converted to binary/ASCII format compatible for encryption tools. Binary format is not directly generated by the CAD tools and has the limitation of fixed field size whereas ASCII format suffers from much slower parsing. These issues motivate us towards secure yet efficient design protection through deterministic design degradation. Our ap-

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.49

Table 1. Characteristics of Floorplan Representations

Floorplan representation	Solution space	Packing time	Space complexity	Type of floorplan	Encoding friendly
SP	$O((n!)^2)$	$O(n^2)$	$2nlgn$	General	Medium
Fast-SP [4]	$O((n!)^2)$	$O(nlglgn)$	$2nlgn$	General	High
BSG	$n!C(n^2, n)$	$O(nlgn)$	$O(nlgn)$	General	Medium
TCG	$O((n!)^2)$	$O(n^2)$	$O(n^2)$	General	Medium
TCG-S	$O((n!)^2)$	$O(nlgn)$	$O(n^2)$	General	Medium
Q-Seq	F(n)	$O(n)$	$3nlg3n$	General	Low
O-tree	$O(n!2^{2n}/n^{1.5})$	$O(n)$	$nlgn + 2n$	Compact	Medium
B* tree [5]	$O(n!2^{2n}/n^{1.5})$	$O(n)$	$nlgn + 2n$	Compact	High
CBL [6]	$O(n!2^{3n}/n^{1.5})$	$O(n)$	$nlgn + 3n$	Mosaic	High
ECBL	$O(C^n_{\lfloor \lambda n \rfloor} n! 2^{3\lfloor \lambda n \rfloor - 4} / \lfloor \lambda n \rfloor^{1.5}) + 3\lfloor \lambda n \rfloor - 4$	$O(\lfloor \lambda n \rfloor)$	$\lfloor \lambda n \rfloor ln \lfloor \lambda n \rfloor$	Extended to General	Medium
TBS	$O(n!2^{3n}/n^{1.5})$	$O(n)$	$nlgn + 3n$	Mosaic extended to General	Low

proach can directly protect any intermediate as well as the complete physical design. In [3] and [11], some preliminary techniques for design perturbation are addressed. A more efficient and reliable technique is proposed here.

3 Encoding of VLSI Floorplans

Of all the stages in VLSI physical design, the output of the partitioning stage is not significantly value-added, whereas the IP value of the routed design greatly depends on the quality of its floorplan. Encoding of floorplan is attempted as it contributes significant IP value to the design.

3.1 Issues in Encoding of Floorplans

Here we justify the encoding of a floorplan to another valid floorplan as opposed to an invalid one analyzing various kind of attacks against security. In case of encoding to a valid floorplan, in addition to time for checking validity, time for checking floorplan quality is also required by the hacker to realize that the hacked file is not directly useful. It reduces the probability of active attack, so valid-to-valid encoding is preferable.

The next issue is choosing a floorplan representation scheme suitable for encoding. A floorplan representation scheme with fewer redundancies (the closer is the representation space to the possible number of floorplans of a particular type) is desirable for floorplan encoding as otherwise it requires relatively longer encoding time to ensure sufficient perturbation and may even lead to an invalid floorplan. Low packing time and space complexity are the other two desirable properties. A floorplan representation scheme is said to be *encoding-friendly* if it satisfies the above three properties and there exists a good floorplan encoding technique for it, satisfying the following properties:

P1: If floorplan $\mathcal{F}_o$ is encoded to $\mathcal{F}_e$ with the key K, $\mathcal{F}_o$ can be recovered uniquely from $\mathcal{F}_e$ using the same key K.

P2: The perturbation of $\mathcal{F}_e$ from $\mathcal{F}_o$ in terms of alteration in floorplan topology and adjacency of modules should be high, guaranteeing sufficient degradation.

P3: Total time for encoding and decoding, key bits requirement for encoding and space/ transmission overhead (other than key bits for correct recovery) should be low.

3.2 Analysis of Floorplan Representations and Encoding Techniques

For a floorplan with n modules, properties of several popular floorplan representations [8] are shown in Table 1. For general floorplans, the encoding friendliness of different floorplan representations are ranked depending on their solution space, packing time and space complexity. Unlike SP and Fast-SP, TCG and TCG-S, capable of supporting incremental update for cost evaluation reduce the robustness of an encoding scheme against the attempt of generating an optimized floorplan from the perturbed one. Hence, Fast-SP, being faster than SP, is chosen for encoding general floorplan. The encoding friendliness of the representations for compact and mosaic floorplans cannot be ranked based on the properties of Table 1. For that purpose, several popular encoding moves are described below.

$M1$: Swapping of two modules. It changes the adjacencies of the modules but has no effect on the topology.

$M2$: Deletion of a module with its subsequent insertion to a different site. It alters the adjacencies of modules and topologies at each site.

$M3$: Single rotation on a floorplan-based tree. Adjacency of the module and its local topology are changed.

$M4$: Swapping two subtrees of a floorplan-based tree. Both the adjacencies and topologies along the boundary of corresponding subsets of the floorplan are altered.

$M5$: Alteration of directive relations of the modules. It has global impact on adjacencies and topologies.

The time and space/transmission requirements of these encoding moves for several floorplan representations are shown in Table 2. For TBS representation, $'\times'$ indicates an invalid move which destroys the twin property of the two binary trees. B* tree for encoding compact floorplan and

Table 2. Complexity of floorplan encoding moves for various floorplan representation schemes

Floorplan rep.	<Time complexity, Preprocessing time, Key bits required, Space/transmission overhead>				
	M1	**M2**	**M3**	**M4**	**M5**
Fast-SP	<O(1),-,O(lgn), ->	<O(1), O(n), O(lgn), ->	NA	NA	<O(1), O(n), O(lgn), ->
O-tree	<O(1),-,O(lgn), ->	<O(n), -, O(lgn), O(1)>	<O(1), -, variable, O(lgn)>	<O(n),-, O(lgn), ->	NA
B* tree	<O(1),-,O(lgn), ->	<O(n), -, O(lgn), O(1)>	<O(1), -, O(lgn), O(1)>	<O(n),-, O(lgn), ->	NA
CBL	<O(1),-,O(lgn), ->	<O(n), -, O(lgn), O(1)>	<O(1), O(n), O(lgn), O(1)>	<O(n),-, O(lgn), ->	NA
TBS	<O(1),-,O(lgn), ->	×	<O(1), O(n), O(lgn), O(1)>	×	NA

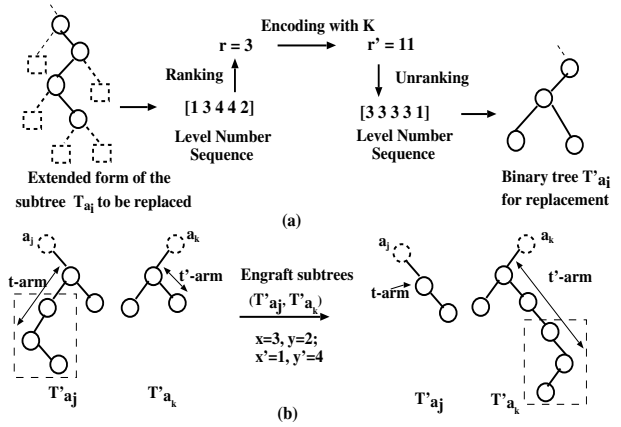


Figure 1. (a) Subtree Replacement (b) Subtree Grafting

CBL for mosaic floorplan are selected considering their behavior towards popular encoding moves in Table 2.

Table 2 reflects that, for Fast-SP representation, there are encoding moves with constant time complexity and zero space/transmission overhead. But for the tree-based representations, a topology-changing encoding move has either $O(n)$ time and zero space overhead or $O(1)$ time and space overhead each. Such move, applied for $O(n)$ times to ensure sufficient floorplan perturbation, results in either $O(n^2)$ timing requirement or $O(n)$ space/ transmission overhead. Hence, a new encoding move $Tree\_Encode$ to encode a binary tree is described next and employed in an efficient encoding scheme for tree-based representations. Binary tree encoding using $Tree\_Encode$ has overall $O(n)$ timing requirement and zero space overhead.

4 A Novel Binary Tree Encoding

The following definitions and illustrations are needed to explain the proposed scheme $Tree\_Encode$.

Left-arm(B) (or Right-arm(B)) is the path from leftmost (or rightmost) leaf to the root of the binary tree B. Left-arm(B) appended with Right-arm(B) constitutes arm(B).

In-subtree (a_i) is the right-subtree for node $a_i \in$ left-arm(B) and vice versa.

Here we use level number sequence representation of binary trees. The level numbers of the leaves from left to right of the extended binary tree of B having n vertices, constitute a sequence of $(n+1)$ integers which uniquely represents B.

Algorithm Tree_Encode(B)
Input: A binary tree B, symmetric key K
Output: An encoded binary tree
1. *for* each node $a_i \in$ arm(B)
  if k, the size(in-subtree(a_i))> s_{Th}
    Tree_Encode(in-subtree(a_i));
  else { Find r = Rank(in-subtree(a_i), k);
    Encode r to r' using K;
    Compute unrank(r', k) to generate subtree T'_{a_i};
    Assign T'_{a_i} to in-subtree(a_i); }
2. *for* $i = 1$ to $m/2$ do /* $m = |$arm(B)$|$*/
  Select distinct nodes a_j, a_k on arm(B) and arms t and t';
  Let x=|t-arm(in-subtree(a_j))|, y=|t′-arm (in-subtree(a_k))|
  Select integer $x' \in [1, x+y)$, Compute $y' = x + y - x'$;
  if $x' > x$, let a_z=$(y-y')^{th}$ node of t′-arm(in-subtree(a_k))
    else, $a_z = (x-x')^{th}$ node of t-arm(in-subtree(a_j))
  if $t \neq t'$, Reflect(a_z); /*swap subtrees of a_z*/
  Separate T_g, the subtree rooted at a_z from rest of the tree;
  if $x' > x$, Append T_g to t-arm(in-subtree(a_j));
    else, Append to t′-arm(in-subtree(a_k));
3. Let σ be the sequence of the next m bits of K;
  for $i = 1$ to m do
    if σ_i = 1, Reflect(a_i); /* a_i is i^{th} node in arm(B)*/
4. Compute r, the position of root a_r on arm(B);
  Encrypt r with K to find new root position q;
  Apply series of rotations about root to obtain new root a_q;

This representation [9] of a binary tree lends it to efficient generation of a lexicographic listing of all the binary trees of same size, i.e., number of vertices, also known as Catalan family $\mathcal{C}_n$ having $C_n = \frac{1}{n+1}\binom{2n}{n}$ members [10].

Rank(B, k***)*** of B with k vertices is the number of binary trees that precede it in the lexicographic ordering of $\mathcal{C}_k$.

Unrank(p, k***)*** generates the binary tree with k vertices, having integer p as its rank in the lexicographic ordering of $\mathcal{C}_k$.

The key idea of step 1 of $Tree\_Encode(B)$ is to replace an in-subtree T_{a_i} by another binary tree T'_{a_i} of same size. First, the level number sequence of T_{a_i} is determined and its rank r is computed in the lexicographic ordering of $\mathcal{C}_k$. Based on the key K, r is then encoded to another distinct rank r', followed by replacing the current in-subtree T_{a_i} by the binary tree T'_{a_i} with rank r' [Figure 1(a)]. In step 2, the nodes in arm(B) are paired up. Let nodes

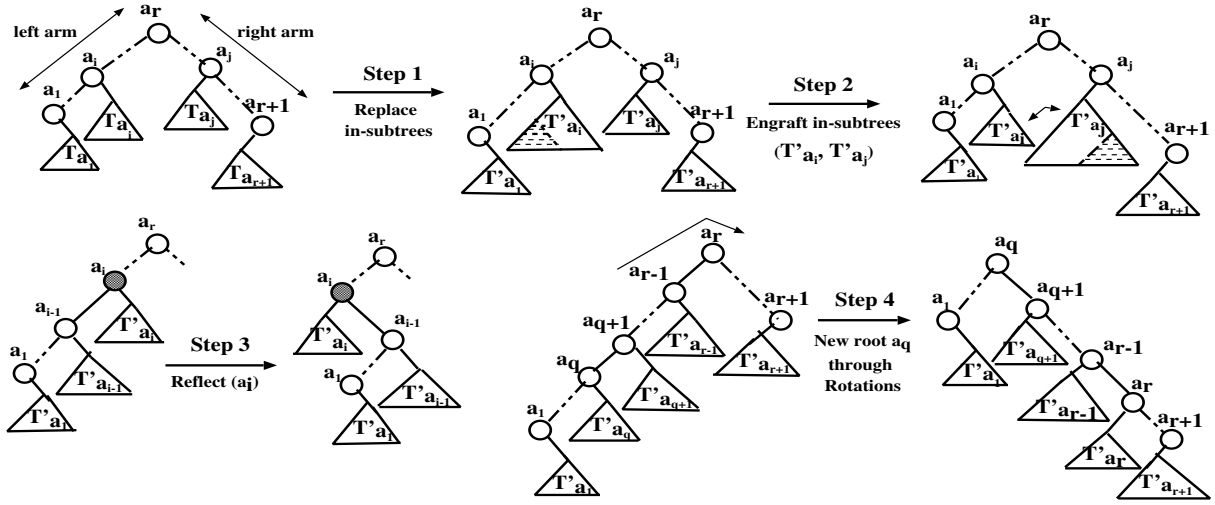


Figure 2. Steps of $Tree\_Encode$

a_j, a_k be such an ordered pair, and x and y be the sizes of t-arm(in-subtree(a_j)) and t′-arm (in-subtree(a_k)) where t, $t'\in$ {left, right}. Grafting a portion of in-subtree(a_j) upon in-subtree(a_k) or vice-versa is performed through integer partitioning of $(x + y)$ into x' and y'. If $x'>x$ ($x'<x$), subtree rooted at $(y - y')^{th}$ ($(x - x')^{th}$) node of t′-arm(in-subtree(a_k)) (t-arm(in-subtree(a_j))) is separated and subsequently prepended to t-arm(in-subtree(a_j)) (t′-arm(in-subtree(a_k)). If $t\neq t'$, subtree is reflected before prepending [Figure 1(b)]. Figure 2 shows all the steps of $Tree\_Encode$.

Time Complexity of Tree_Encode(B): The binary tree generation algorithm and the ranking-unranking algorithm have linear time complexities [10]. Hence time complexity of step 1 is $O(n)$, n=# nodes. For step 2, it is $O(n)$ and for step 3 and step 4, it is $O(m)$ each, m=$|arm(B)|$, $m\leq n$.

Average Perturbation Distance between Original and Perturbed Floorplan: The average distance d_s (min. number of rotations required to transform one to the other) of all pairs of Catalan family binary trees $\mathcal{C}_s$ of size s is greater than $s/2$. For large s, d_s is close to $(s - 1)$. For replacement of m in-subtrees, each of average size s, average distance between the encoded and the original tree is greater than $ms/2$. For both B* tree and CBL, a rotation, resulting at least one change in topology and one loss of adjacency of module, contributes unit perturbation distance. Hence perturbation distance due to step 1 is greater than $ms/2$.
Relocating a subtree of size s i.e. a subset of s modules for B* tree changes at least adjacency of $4s^{1/2}$ modules placed on the perimeter of the subset (considering all modules are of nearly equal size). So, step 2 and step 3 alter on average, at least adjacencies of $2ms'^{1/2}$ and $4ms^{1/2}$ modules respectively, s' = size of the subtree relocated in step 2. Relocating a subtree of size s of the unlabeled binary tree, drawn from the T sequence of CBL, changes T junction coverage, hence the adjacency of all s modules. So, for CBL, step 2 and step 3 alter on average at least $\frac{1}{2}ms'$ and ms adjacencies of modules respectively. Hence, perturbation distance between original and perturbed floorplan, causing degradation of the design, is $O(ms)$ i.e. $O(n)$, $n =$ # modules.

5 The Proposed Floorplan Degradation

5.1 Basics of Floorplan Degradation

Let $\mathcal{F}_o$ be an optimized floorplan with n original modules and K be the symmetric key, known to the IP owner. In case of floorplan transmission, K is encrypted using public-key-encryption RSA [7] and transmitted through ultra-secure channel from owner to buyer. Now, for protection, a floorplan can be degraded in following two ways:

1. n' ($2 \leq n' \leq lgn$) dummy modules are inserted on the given floorplan. $lglgn$ bits of K determine the value of n'. Each dummy module contains selective part of the circuits of two or more strongly interconnected original modules such that entire circuit of certain original modules are duplicated in dummy modules. So simple deletion of the modules with entirely duplicated parts of the circuit fails to retain all the original modules. Dummy modules are placed in the floorplan in a specified manner so that those can be identified later and deleted [3]. The value of n' is restricted to lgn to limit the transmission overhead. This step protects the valuable partitioning information of the design.

2. Encoding of the floorplan with total $N = n + n'$ modules is achieved through altering the adjacency relationships of all the modules in the floorplan.

Benefits: Design is degraded due to (i) increased area of perturbed floorplan (because of dummy modules and huge dead spaces between the modules) (ii) increase in total wire-

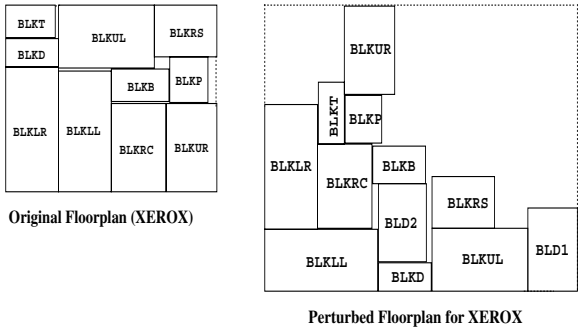


Figure 3. Perturbation of Floorplan Xerox

length and critical path delay.
Figure 3 shows the perturbation of Xerox floorplan.

5.2 Encoding based on Floorplan Type

5.2.1 General floorplan as Fast-SP

Fast-SP consists of an ordered pair of module name sequences $S = (\Gamma_+, \Gamma_-)$, where $\Gamma_+(\Gamma_-)$ represents the order of positive (negative) loci from top-left to bottom-right (from bottom-left to top-right).
1. Apply move $M1$ by selecting module names $\mathcal{M}_1$, $\mathcal{M}_2$ based on K and swapping $\mathcal{M}_1$, $\mathcal{M}_2$ in both Γ_+ and Γ_-.
Repeat move $M1$ for N times.
2. Apply move $M2$ by choosing the three module names $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ based on K and placing the module just after $\mathcal{M}_1$ in Γ_+, after $\mathcal{M}_2$ in Γ_+ and after $\mathcal{M}_3$ in Γ_-.
Repeat move $M2$ for N times.
3. Move $M5$: Select a particular sequence, either Γ_+ or Γ_-, and a subsequence of it based on K. Reverse the subsequence within that selected sequence.

5.2.2 Compact floorplan as B*-tree

1. Permute S, the inorder sequence of B*-tree B, using move $M1$ N times to generate module name sequence S'.
2. Apply $Tree\_Encode(B)$ to encode B to B_e.
3. Generate the encoded B* tree B' with the topology of B_e and inorder sequence S'.

5.2.3 Mosaic floorplan as CBL

CBL represents a mosaic floorplan with three-tuple $<S, L, T>$. S is a permutation of module names, L consists of $(N-1)$ orientation bits and T has no more than $(2N-2)$ bits representing T junction coverage. Append $'0'$(s), if required, to T to generate the sequence T_e of length $2N-2$ so that T_e corresponds to the bit-string representation of an unique binary tree [10].
1. Apply move $M1$ for N times to permute S to S'.
2. Bitwise XOR L with $(N-1)$ key-bits to generate L'.
3. Construct binary tree B_t from the sequence T_e, encode B_t to B_t' using $Tree\_Encode(B)$ and regenerate bit-string representation T' of B_t'.
$< S', L', T' >$ represents the encoded floorplan.

5.3 Analysis of Proposed technique

Time Complexity: Each of the steps of floorplan encoding techniques for three different floorplan representations is of $O(N)$ time, so the time complexities of the techniques are linear in total number of modules of the floorplan.

Strength of Proposed Techniques: The strength of the floorplan encoding schemes against brute-force attack [7] for three chosen floorplan representations are as follows.

Fast-SP: The number of bits, required to define move $M1$, $M2$, $M5$ are $2lgN$, $lgN + 2lg(N-1)$ and $lgN + lg(N-1)+1$ respectively. So the robustness of step 1, step 2 and step 3 are $O(N^2)$, $O(N(N-1)^2)$ and $O(2N(N-1))$ respectively. Hence the robustness of the technique against brute-force attack is $O(2N^{3N+1}(N-1)^{2N+1})$. Packing time of Fast-SP is $O(NlglgN)$ and computation of HPWL of the floorplan requires $O(E)$ times (E = # net terminals). As Fast-SP does not support incremental updates of packing, $O(2N^{3N+1}(N-1)^{2N+1}(NlglgN+E))$ time is required to certainly obtain the desired high quality floorplan.

B* tree: Algorithm $Tree\_Encode(B)$ provides robustness of $O(C_s^m)$, $O(m!m'^{m/2})$, $O(2^m)$ and $O(m)$ in step 1, step 2, step 3 and 4 respectively, where m=|arm(B)|, $C_s = \frac{1}{s+1}\binom{2s}{s}$, s = the average size of in-subtrees and m'=|arm(in-subtree(a)|, $a \in$arm(B). So the robustness of $Tree\_Encode(B)$ is $O(mm!(2m'^{\frac{1}{2}}C_s)^m)$. Hence for B* tree, perturbed floorplan can be generated from any of $O(N^{2N}mm!(2m'^{\frac{1}{2}}C_s)^m)$ possible floorplans. Considering time for packing and computation of HPWL, $O(N^{2N}mm!(2m'^{\frac{1}{2}}C_s)^m(N+E))$ time is required to certainly obtain the desired optimized compact floorplan.

CBL: Robustness of steps 1, 2 and 3 of the floorplan encoding technique with CBL representation are $O(N^{2N})$, $O(2^{(N-1)})$ and $O(mm!(2m'^{\frac{1}{2}}C_s)^m)$ respectively. Hence, $O(N^{2N}2^{(N-1)}mm!(2m'^{\frac{1}{2}}C_s)^m(N+E))$ time is required to obtain desired mosaic floorplan.

The perturbation of the floorplan estimates the strength the technique against any attempt of generating the desired floorplan from the perturbed one.

6 Experimental Results

The proposed technique of floorplan perturbation was implemented in C on 1.2 GHz Sun Blade 2000 machine using Sun OS 9 and was tested on MCNC floorplan benchmarks. We started with two sets of floorplan results, shown in Table 3 – floorplan set I: a set of compact floorplans as Fast-SP and B* tree representations, floorplan set II: a set of mosaic flooplans represented as CBL, which are to be

Table 3. Benchmark Floorplans before Encoding

Benchmark Floorplan	# Blocks	# Nets	Floorplan Set I (for encoding with Fast-SP, B* Tree)		Floorplan Set II (for encoding with CBL)	
			Area (mm^2)	**HPWL** (mm)	**Area** (mm^2)	**HPWL** (mm)
Apte	9	97	47.31	363	47.30	363
Xerox	10	203	20.02	366	20.70	374
Hp	11	83	9.27	143	9.27	153
Ami33	33	103	1.19	75	1.20	83
Ami49	49	408	37.13	892	37.30	953

Table 4. Area and HPWL Degradation due to Proposed Encoding Technique

Benchmark Floorplan	Fast-SP		B* tree		CBL	
	% increase in area	% increase in HPWL	% increase in area	% increase in HPWL	% increase in area	% increase in HPWL
Apte	183.5	208.3	181.0	195.0	185.1	189.7
Xerox	176.1	267.2	142.8	253.3	306.4	285.3
Hp	202.7	264.1	218.8	262.6	321.9	293.9
Ami33	235.0	226.2	174.2	211.8	250.9	236.2
Ami49	294.9	245.8	172.1	202.7	246.8	241.4

perturbed. The degradation in high quality flooplans due to insertion of dummy modules and alteration of adjacencies of the modules is measured in terms of increase in area and HPWL and shown in Table 4. This degradation is more compared to that in [11]. The perturbation distance between

Table 5. Effect on Adjacencies of Modules and Transmission Overhead

Benchmark Floorplan	% change in adjacency of modules			% overhead in transmission
	Fast-SP	B* tree	CBL	
Apte	87.5	85.5	81.2	22.22
Xerox	86.7	86.4	85.0	20.00
Hp	87.3	81.2	88.1	18.18
Ami33	94.4	94.1	92.1	12.12
Ami49	96.0	95.7	94.4	8.16

input floorplan and perturbed floorplan [Table 5] is measured by % change in the adjacencies of modules averaged on all the modules. For a particular module say $\mathcal{M}$, it is $|A - B|/|A|$, where A=$\{x|x \in adjacent(\mathcal{M})$ before perturbation$\}$, B=$\{y|y \in adjacent(\mathcal{M})$ after perturbation$\}$. % overhead in transmission, due to insertion of dummy modules in the floorplan, decreases with size of the floorplan.

7 Conclusion and Future Scope

An effective approach of encoding a design IP through perturbation instead of design file encryption is proposed to protect VLSI physical design in repositories and during web transmission of the design. Popular floorplan representations are analyzed for various encoding moves and a novel way of encoding tree-based floorplan representations is suggested. Experimental results reflecting huge degradation in the floorplan quality ensure robustness of the technique.

The technique for floorplan encoding can be extended for placed and routed design in future.

References

[1] E. Charbon and I. H. Torunoglu, On Intellectual Property Protection, *Proc. of the IEEE Custom Integrated circuit Conference*, 2000, pp. 517-522.

[2] W. Adi, R. Ernst, B. Soudan, A. Hanoun, VLSI Design Exchange with Intellectual Property Protection in FPGA Environment Using both Secret and Public-Key Cryptography, *Proc. of Int. Symposium on VLSI*, 2006.

[3] D. Saha, P. Dasgupta, S. Sur-Kolay, S. Sen-Sarma, A Novel Scheme for Encoding and Watermark Embedding for IP Protection of VLSI Physical Design, *Proc. of Int. Conf. on Computation: Theory and Application*, 2007, pp. 111-116.

[4] X. Tang and D. F. Wong, Fast SP: A Fast Algorithm for Block Placement Based on Sequence Pair, *Proc. of ASP Design Automation Conference*, 2001, pp. 521-526.

[5] Y.-C. Chang, Y.-W. Chang, G.-M. Wu and S.-W. Wu, B*-trees: A New Representation for Non-Slicing Floorplans, *Proc. of Design Automation Conference*, 2000, pp. 458-463.

[6] X. Hong, S. Dong, G. Huang, Y. Cai, C. Cheng, Corner Block List Representation and Its Application to Floorplan Optimization, *IEEE Trans on CAS*, Vol. 51, No. 5, 2004.

[7] A. Menezes, P. V. Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[8] H. H. Chan, S. N. Adya and I. L. Markov, Are Floorplan Representations Important In Digital Design?, *Proc. of Int. Symposium Physical Design*, 2005.

[9] F. Ruskey, T. C. Hu, Generating Binary Trees Lexicographically, *SIAM J. of Computing*, Vol. 6, 1977, pp. 745-758.

[10] J. Lucas, D. Baronaigien, F. Ruskey, On Rotations and the Generation of Binary Trees, *J. of Algorithms*, 1993, pp. 1-25.

[11] D. Saha, S. Sur-Kolay, An Analytical approach for Direct IP Protection of VLSI Floorplans, *Proc. of ICIIS*, 2008.

Design Optimization and Automation for Secure Cryptographic Circuits

Kuan Jen Lin[+], Yi Tang Chiu and Shan Chien Fang
Department of Electronic Engineering, Fu Jen Catholic University, Taiwan
[+]kjlin @mail.fju.edu.tw

Abstract

Various logic design styles have been proposed to counteract DPA (Differential Power Analysis) attacks for secure cryptographic IC design. However, only a couple of papers addressed the automatic synthesis and optimization for these secure logic circuits. This paper attempts to identify common optimization issues in typical masking-based countermeasures. They include (1) constrained Reed-Muller (RM) logic minimization, (2) minimum decomposition of multi-input AND gates and (3) minimum number of mask bits used to randomize power consumption. An OFDD-based heuristic method is proposed to minimize the RM logic with emphasis on literal number. The latter two optimization problems are formulated as zero-one integer linear programming and graph coloring problems respectively. Based on these formulations and optimizations, an automated design flow for secure cryptographic IC design was implemented in C language.

1. Introduction

Securing information systems can be divided into several kinds of different layers. The algorithm layer such as AES (Advanced Encryption Standard) has been designed to be secure against computational cryptanalysis. Thus, as long as the key is kept hidden from an attacker, the encrypted information can be considered secure. However, the physical implementation of cryptographic algorithms leaks the so-called side channel information such as power consumption and electro-magnetic emanation during the computation of cryptographic algorithms, and that can be exploited by an attacker to find the secret key.

In 1998, Kocher et al. first reported that the power consumption of a smart card could reveal the secret key of the cryptographic algorithm [7]. They found that the power consumption of a device executing a cryptographic algorithm is correlated with the intermediate data and an attacker can use statistical analysis to find the key from power traces of data processing. The attack called *Differential Power Analysis* (DPA) has been considered as the most dangerous attack to the security of cryptographic devices. Hereafter, there is a lot of research conducted on corresponding countermeasures against the DPA attack. An excellent survey can be found in the book written by S. Mangard et al [11]. They classified the countermeasures into two kinds: hiding and masking. A popular method of hiding is to make the power consumption constant and regardless of what intermediate result being produced [e.g. 9, 18]. However, it is hard to implement the approach using standard semi-custom design flow. The masking method attempts to randomize the intermediate computed data and make the power consumption unpredictable. Many logic styles which mask data at the gate level and mostly can be realized with common standard cell libraries have been proposed [e.g. 5, 10, 14]. Although secure masked cells or logic structure may be different among them, the design flows for these masking approaches basically are similar.

Only a couple of papers addressed the automatic synthesis and optimization for securing cryptographic IC design against DPA attacks [6, 8, 18], which all are about hiding-based methods. This paper attempts to identify common optimization issues in typical masking-based countermeasures. They include (1) constrained Reed-Muller logic minimization, (2) minimum decomposition of multi-input AND gates and (3) minimum number of mask bits used to randomize power consumption. An OFDD-based heuristic method is proposed to minimize the RM logic with emphasis on literal number. The latter two optimization problems are formulated as zero-one integer linear programming and graph coloring problems respectively. Based on these formulations and optimizations, an automated design flow for secure cryptographic IC design was implemented in C language..

In the next section, a typical design flow for secure cryptographic IC design and common optimization issues will be described. Three key optimizations will be discussed in the following three sections. Finally, experimental results will be given.

2. Design Flow for Secure Cryptographic Circuits

One of popular countermeasures is to mask computed

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.57

data to make each probably attacked signal unpredictable. It is achieved by exclusive-oring (XOR) a signal *b* with an uniformly distributed random variable m_b (i.e. $p(m_b=0)=p(m_b=1)=1/2$) and is independent of b) to get $b_m=b \oplus m_b$. The masking approach replaces each cell that is to be protected with a masked cell, as shown in Fig. 1. For example, a 2-input XOR function $g=a \oplus b$ can be replaced with $g_m= a_m \oplus b_m$ and $m_g= m_a \oplus m_b$. The m_g is called as a correction mask. Note that the value of m_a must differ from that of m_b. A masked cell for 2-input AND *g=ab* proposed in [5] is derived as follows:

$$g_m = \overline{m}_a \overline{a}_m m_b + m_a \overline{a}_m b_m + \overline{m}_a a_m b_m + m_a a_m m_b .$$

Its corresponding correction mask is m_b. In MDPL style [14], it allows that $m_a=m_b$ and all masks even use the same mask. However, a recent paper has shown that single-bit masking can be broken by filtering of the masked probability density function [16]. Therefore, for security, the mask values for two input signals of a 2-input masked AND should be different.

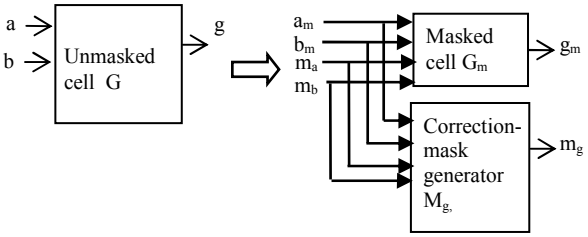


Fig 1: A 2-input unmasked cell G and a corresponding masked cell G_m and correction-mask generator $M_{g,}$ where $a_m=a \oplus m_a$, $b_m= b \oplus m_b$, and $g_m=g \oplus m_g$.

Another important characteristic of cryptographic circuits is the use of XOR gates. Specifically, its circuit structure often is AND-XOR style, i.e. Reed Muller (RM) form, rather than AND-OR style. It is well-known that RM logic style usually can save much area from AND-OR style for cryptographic applications.

We now summarize a typical design flow for masking-based cryptographic circuits against DPA attacks, as described in Fig. 2. The first step is to transform a general circuit to be the RM form. The second step is to replace unmasked cells with masked cells. Up to now, secure masked AND cells all have fan-in constraint. Namely, only 2-input masked AND cells are provided. Therefore, the main task of this step is to decompose multi-input AND gates into a network of 2-input AND gates. Furthermore, a masked AND cell is much more expensive than a masked XOR cell. For example, to have resistance against a DPA attack exploiting glitch and early propagation effect, a masked 2-input AND cell needs more than 5 times of area of a unmasked 2-input AND cell in a recent design iMDPL [15]. Based on these observations, we draw two optimization issues:

Optimization issue 1*: The logic minimization of RM expression should put emphasis on the number of literals rather than the number of product.*

Optimization issue 2*: Multi-input AND gates must be decomposed to a network of 2-input AND gates. The total number of 2-input AND gates should be minimized without violating timing constraint.*

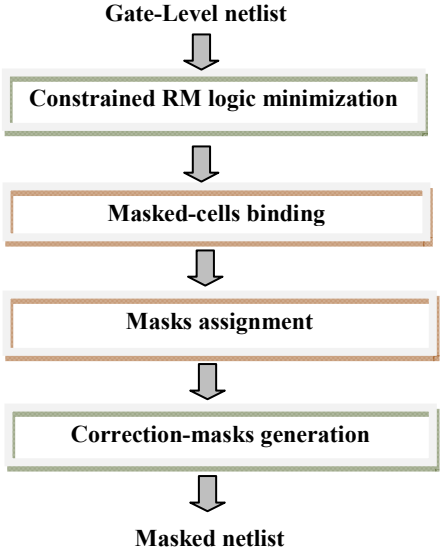


Fig. 2: Design flow for masked cryptographic circuits.

The third step is to assign random masks to all attackable signals in each round. The assignment must satisfy the constraint: the masks used for two inputs of a masked cell need to be independent of each other. A true RNG (Random Number Generator) is used to produce these masks. To save the complexity of the RNG design, we have the following issue:

Optimization issue 3*: The mask bits should be minimized without violating the independency constraint.*

After the masked circuit netlist is obtained, some additional schemes such as pre-charge or dual-rail coding can be added. They are not considered in this paper.

3. Constrained RM Logic Minimization

3.1 Fixed Polarity Reed-Muller Expansion

An n-variable Boolean function can be expressed as exclusive sum-of-product form, i.e. AND-XOR form, which is known as Reed-Muller Form. If every variable

appears either true or complemented, but not in both form, the expression is defined as Fixed Polarity Reed-Muller Form (FPRM).The advantage of the FPRM is that the resulting circuit needs at most n inputs in contrast to up to 2n inputs in Mixed Polarity (MP) RM representation. Although the MPRM usually needs less number of gates, the FMRM can save wring area. The RM expression under our consideration will be the FMRM form. A polarity is specified by an integer p, $0 \le p \le 2^n-1$, which can be written as a binary n-tuple $p=(p_n p_{n-1} \ldots p_1)$. Given a polarity p, an n-variable Boolean function can be expressed by the FPRM form as:

$$f(x_n x_{n-1} \ldots x_1) = \oplus \sum_{i=0}^{2^n-1} b_i \pi_i$$

where the integer subscript i, $0 \le i \le 2^n-1$, can be written as a binary n-tuple $i=(i_{n-1} i_{n-2} \ldots i_0)$, '$\Sigma$'is the XOR operation, $b_i \in \{0,1\}$ denotes whether the π_i exists, the pi-term π_i can be represented as

$\pi_i = \dot{x}_n \dot{x}_{n-1} \ldots \dot{x}_1$:

$$\dot{x}_j = \begin{cases} 1 & i_j = 0 \\ x_j & i_j = 1, p_j = 0 \\ \bar{x}_j & i_j = 1, p_j = 1 \end{cases}$$

The number of product terms largely depends on the polarity. The problem to find the best polarity with a least cost is computationally extensive in both space and time. A lot of research has been conducted to the minimization of FPRM expression. Tabular methods and Matrix transformations [e.g. 12, 17] need at least $O(2^n)$ complexities for both space and computation time. Several works have shown that the use of OFDD (Ordered Function Decision Diagram) to represent an FPRM expression generally needs less memory and less computation time than tabular and matrix methods [e.g. 1, 2, 4].

3.2 OFDD-Based FPRM Expansion

The goal of most previous works [e.g. 1, 2, 4] is to minimize the number of product terms.The proposed minimization will put emphasis on the number of literals rather than the number of product (see optimization issue 1). For mixed polarity RM forms, there are several works that addressed the issue [e.g. 13]. However, the key idea behind them is to search products with k-distance to be combined to single product. It is not applicable for FPRM expressions. In this paper, we proposed an OFDD-based approach. The OFDD is a kind of ordered decision diagram, which is constructed by Davio expansion [3]. For an n-input Boolean function $f(X) = f(x_n, x_{n-1}, \cdots, x_i, \cdots, x_2, x_1)$, the positive and negative Davio expansions about index variable x_i are given as follows:

Positive Davio expansion: $f = f_0 \oplus x_i f_2$;

Negative Davio expansion: $f = f_1 \oplus \bar{x}_i f_2$.

Where $f_0 = f(x_n, x_{n-1}, \cdots, 0, \cdots, x_2, x_1)$ is *f(X)* with x_i replaced by 0, $f_1 = f(x_n, x_{n-1}, \cdots, 1, \cdots, x_2, x_1)$, x_i replaced by 1, and the Boolean difference f_2, $f_2 = f_0 \oplus f_1$ [3].

A simple example of OFDD representing the function $f(x_3, x_2, x_1) = x_3 x_2 \bar{x}_1 + \bar{x}_3 x_2 x_1 + \bar{x}_2 x_1$ is shown in Fig. 3. The variable order from root to leaf is x_1 -> x_2 -> x_3. The polarity selection is $\bar{x}_3 x_2 \bar{x}_1$.

Given a polarity assignment, only one decomposition type of Davio expansion (either positive or negative) about the same index variable is carries out at all nodes on the same tree level. The decomposition is performed until all non-decomposed nodes being a 1- or 0- terminal, denoted by a square box, whose function equals 1 and 0 respectively. The resultant OFDD is called as FPRM tree in this paper. The corresponding FPRM expression is derived as follows:

$f(X) = \sum_{v \in U} \otimes$ (NPP(v)), where U is the set of all 1-terminals and the NPP(v) (Node Path Product) is the product of all the literals on the path from the root to the node v. For example, the NPP(V_6) is $x_2 \bar{x}_3$. By finding the NPP for all 1-terminals, we get $f = 1 \oplus \bar{x}_1 \oplus x_2 \bar{x}_3$ from Fig. 3.

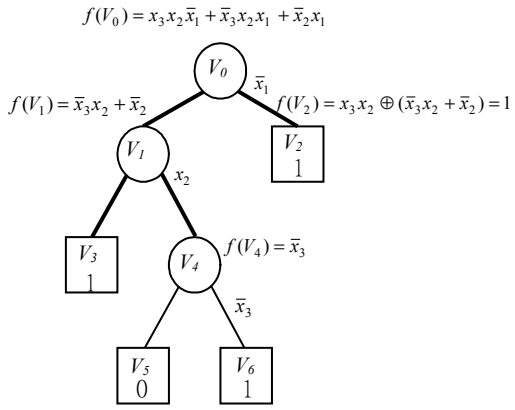


Fig. 3: An OFDD for $f(x_3, x_2, x_1) = x_3 x_2 \bar{x}_1 + \bar{x}_3 x_2 x_1 + \bar{x}_2 x_1$, where the selected polarity is $\bar{x}_3 x_2 \bar{x}_1$.

3.3 Cost Estimates

The following cost estimates are proposed to determine the polarity during the tree decomposition and evaluate the cost of an FPRM tree.

- *weight (n_i)*: equals the number of literals in the path from root to the node *i*. The followings can be

easily derived:

weight(n_i ->low_successor)=weight(n_i);
weight(n_i ->high_successor)=weight(n_i)+1;

- *Pnumber(f)*: For a Boolean function *f*, if *f=1 (f=0)*, *Pnumber(f)=1(0)*. Otherwise, *Pnumber(f)* equals the number of its product terms while not counting any one-literal term.

Let S_j be the node set with the same decision variable x_j.

- $C0_{x_j} = \sum_{n_i \in S_j} weight(n_i) \times Pnumber(f_0(n_i))$
- $C1_{x_j} = \sum_{n_i \in S_j} weight(n_i) \times Pnumber(f_1(n_i))$

For a 1-terminal node, the weight value equals the number of literals of its node-path product. The total Literal Cost (LC) of a PFRM tree is defined as follows:

- $LC = \sum_{n_i \in V \wedge weight(n_i)>1} weight(n_i)$

where *V* is the set of 1-terminal nodes.

3.4 Algorithms

The proposed heuristic algorithm attempts to build an FMRM tree with a minimum *LC*. It consists of three main steps:

(1) Transform a circuit description from SOP form to DSOP (Disjoint SOP) form: The task is performed by running ESPRESSO [19]. The DSOP simplifies the XOR operation between two expressions.

(2) Construct an initial FPRM tree from the DSOP representation: The variable order is derived according to the number of literals occurring in the DSOP expression. During the tree construction, the cost estimates $C0_{x_j}$ and $C1_{x_j}$ are used to determine the polarity for variable x_j.

(3) Refine the FPRM tree iteratively: The initial FPRM tree is reconstructed by changing the polarity for one of variables. For an n-input function, there are *n* trees reconstructed. If the result with the least *LC* among the *n* FPRM trees has less *LC* than that of the initial FPRM tree, its polarity assignment is chosen. Otherwise, the refinement stops. If there are two trees to have the same *LC*, the one with fewer product terms is chosen. One variable at most changes its polarity once. Once a better assignment is obtained, the same scenario is repeated for all those variables having not yet polarity changed. In the worst case, this step will construct at most $O(n^2)$ trees.

4. Decomposition of Multi-Input AND Gates

Because only 2-input masked AND cells are provided, multi-input AND gates must be decomposed into a network of 2-input AND gates. It is not trivial to find a decomposition that uses the minimum number of 2-input AND gates when simultaneously considering a set of multi-input AND gates. We formulate the minimization as a problem of zero-one integer linear programming by using a binary decision variable: $X=\{x_{i,j};\ i=1, 2,\ldots, n;\ j=1, 2,\ldots, n\}$, where *n* is the number of variables. In an FMRM expression, this means that there are at most *n* different literals. We let them be $v_1, v_2,\ldots, v_n$. There are at most $(n^2-n)/2$ different 2-input terms and *n* different one-literal term in the decomposition. The $x_{i,j} = 1$ (0) means that the product term v_iv_j is (not) selected. If *i=j*, the term has only one literal. For each multi-input AND gate, we derive a set of conditions. For example, the term $v_1v_3v_4$ implies the following conditions must be satisfied

$$x_{13} + x_{14} + x_{11} \geq 1$$
$$x_{13} + x_{34} + x_{33} \geq 1$$
$$x_{14} + x_{34} + x_{44} \geq 1$$

The three conditions ensure that each variable will be contained in a selected term. Subject to such conditions derived for all multi-input AND gates, the problem is to minimize

$$\sum_{\substack{i=1,\ldots,n \\ j=1,\ldots,n}} B_{ij} x_{ij}$$

where $i \leq j$ and if $i \neq j$, $B_{ij}=1$,otherwise $B_{ij}=n^2$.

The solution will use one-literal terms as less as possible. A solution of the integer linear programming problem defines a set of x_{ij} =1. Each multi-input AND gate then can be decomposed into several terms from this set. Finding a minimum decomposition (i.e. with minimum terms) corresponds to a minimum covering problem. Since the input number is often small, exact solution can be easily derived. The above derived solution is the first level decomposition. Each selected 2-literal term will be replaced with a new variable. Then, the same formulation is applied to drive the second level decomposition. The same approach is repeated to get the output.

5. Mask Assignment

In mask assignment, we attempt to use the minimum number of bits to satisfy the *independency constraint*: no any two inputs of a masked cell carry the same random mask. The procedure of mask assignment is shown in the Fig. 4. It traverses the circuit from primary outputs to its primary inputs. Each wire will be labelled with a string, which will be used to differentiate the assignment. Fig. 5 shows an example

of how the procedure *mask_assignment()* to label the string on each wire. The three primary outputs O_0, O_1 and O_2 are labelled with a, b and c. Then the strings are backward passed to cells' inputs. If a wire has more than one branch, the strings of its branches are concatenated to become its label string.

After all primary inputs are labelled, we construct a conflict graph, in which a node represents a primary input and an edge (v, u) exists if LS(v) ∩ LS(u) ≠∅, which means that the label strings of v and u contain at least one common literal. According to the *independency constraint*, they must be assigned with different masks. Fig. 6 shows such a conflict graph for the example in Fig. 5. Since only one of inputs affects the mask of the masked AND cell's output, we can let I_3 and I_5 to share the same mask. This is why they are combined to be one supernode in Fig. 6. Based on the above formulation, the minimization of mask assignment can be transformed to a *graph colouring* problem. In this example, it needs 4 mask bits.

```
mask_assignment( Circuit C) {
  //LS(v) is the label string of a wire v
   Traverse backward the circuit from POs to PIs:
       Label different literals to each PO
       For each gate,
          copy the output's LS to all inputs as their LS
          If a wire v forks, LS(v)=∪ LS(u_i), where u_i is a
             branch of v.
    Construct a conflict graph
    Assign masks to each PI by graph colouring :

}
```

Fig. 4: The procedure of mask assignment.

6. Experimental Results

The section mainly reports the current result of our heuristic approach to derive FMRM logic. The proposed FMRM logic synthesis that tries to minimize the number of literals was implemented in C language and run under a PC that has 2.01 GHz AMD dual-core Athlon processor. A set of MCNC benchmark circuits was used to evaluate the method. Table 1 shows the results obtained. The solutions with a minimum *LC* are derived by exhaustive searching. However, due to the computation time exploding (over 72hr), the minimum for several cases were not determined, which are indicated by "-". Note that there are five examples whose FPRM expression has a minimum *LC* but not a minimum number of product terms. The column "Ours" shows the results derived by our approach. Compared to those determined minimums, our results need additional 1% cost only. It is noticeable that the OFDD-based works in [1, 4] need more 10% product terms than those minimums. As for the computation time, the circuit apex3 and apex5 take about 25 and 35 minutes respectively. Other circuits synthesized can be obtained within one minute.

We used the *lp-solve* to derive optimal decomposition for multi-input AND gates. The circuits with *LC* less than 5000 can be solved within one hour. However, other circuits may take unacceptable time. For example, we cannot get the optimal solution for *table3* even running 48 hours. Similarly, it is also impossible to derive exact minimization for mask assignment for larger circuit using graph coloring method. Hence, heuristic techniques must be exploited.

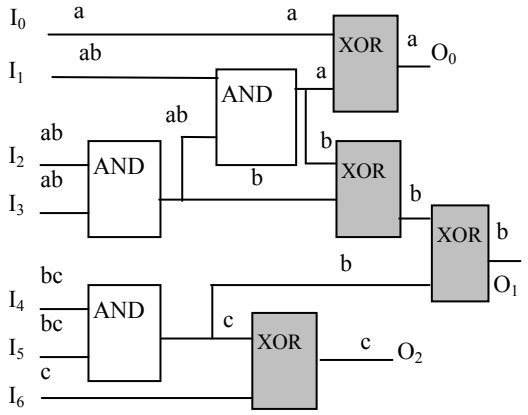


Fig. 5: An example of *mask-assignment()* that labels all wires from POs to PIs.

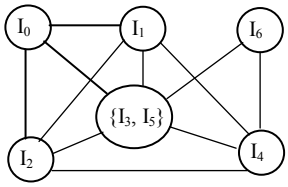


Fig. 6: A conflict graph for the example in Fig. 5.

7. Conclusion

In this paper, common optimization issues in typical masking-based countermeasures have been identified, which include (1) constrained Reed-Muller logic minimization, (2) (2) minimum decomposition of multi-input AND gates and (3) minimum number of mask bits used to randomize power consumption. An OFDD-based heuristic method has been proposed to minimize the RM logic with emphasis on the literal number. The latter two optimization problems have been formulated as zero-one integer linear programming and graph coloring problems respectively. Based on these formulations and optimizations, an automated design flow for secure cryptographic IC design was implemented in C language.

8. Acknowledgments

The authors would like to thank the anonymous reviewers for their comments to improve the quality of this paper. This work was supported by Taiwan NSC under Contract No. NSC 96-2221-E-030-015-MY2.

9. References

[1] S. Aborhey, "Reed-Muller Tree-based Minimization of Fixed Polarity Reed-Muller Expansions," *IEE. Proc., Comput. and Digit. Tech.*, pp. 63-70, 2001.

[2] B. Becker, and R. Dreschler, "OFDD based Minimization of Fixed Polarity Reed-Muller Expressions Using Hybrid Genetic Algorithms," *IEE International Conference on Computer design*, pp. 106-110, 1994.

[3] M. Davio, J.P. Deschamps, and A. Thayse, "*Discrete and Switching Functions*," McGraw-Hill Int'l, 1978.

[4] R. Drechsler, M. Theobald, and B. Becker, "Fast OFDD-based Minimization of Fixed Polarity Reed-Muller Expansions," *IEEE Trans. Comput.*, pp. 1294-1299, 1996.

[5] J. D. Golić and R. Menicocci, "Universal Masking on Logic Gate Level," *Electronics Letters* 40(9), pp. 526–527, 2004.

[6] S. Guilley et al., "Secured CAD Back-End Flow for Power-Analysis-Resistant Cryptoprocessors," *IEEE Design and Test of Computers*, pp. 546-555, No. 6, 2007.

[7] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Advances in Cryptology – CRYPTO '99, LNCS,* vol. 1666, pp. 388-397, 1999.

[8] K. J. Kulikowski, A. Smirnov and A. Taubin "Automated Design of Cryptographic Devices Resistant to Multiple side-Channel attacks," *2006, LNCS*, vol. 4249, pp. 399-413, 2006.

[9] K. J. Kulikowski, M. Su, A. B. Smirnov, A. Taubin, M. G. Karpovsky and D. MacDonald, "Delay Insensitive Encoding and Power Analysis: A Balancing Act," ASYNC 2005, pp. 116-125, 2005.

[10] K. J. Lin, S. C. Fang, S. H. Yang and C. C. Lo, "Overcoming Glitches and Dissipation Timing Skews in Design of DPA-Resistant Cryptographic Hardware," *IEEE/ACM DATE*, pp. 1265-1270, 2007.

[11] S. Mangard, E. Oswald and T. Popp, *Power Analysis Attacks Revealing the Secrets of Smart Cards*, Springer, 2007.

[12] J. F. Miller, and P. Thomson, "Highly Efficient Exhaustive Search Algorithm for Optimizing Canonical Reed-Muller Expansions of Boolean Functions," *Int. J. Electron.*, pp. 37-56, 1994.

[13] A. Mishchenko and M. Perkowski, "Fast Heuristic Minimization of Exclusive-Sums-of-Products," *Proc. Reed-Muller Workshop*, pp. 242-250, 2001.

[14] T. Popp and S. Mangard, "Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints," *CHES 2005*, pp.172-186, 2005.

[15] T. Popp, M. Kirschbaum, T. Zefferer and S. Mangard2, "Evaluation of the Masked Logic Style MDPL on a Prototype Chip," *CHES*, *LNCS 4727*, pp.81-94, 2007.

[16] P. Schaumont1 and K. Tiri, "Masking and Dual-Rail Logic Don't Add Up," *CHES*, *LNCS 4727*, pp. 95-106, 2007.

[17] E.C. Tan, H. Yang, "Fast Tabular Technique for Fixed-Polarity Reed-Muller Logic with Inherent Parallel Process," *Int. J. Electron.*, pp. 511-520, 1998.

[18] K. Tiri, D. Hwang, A. Hodjat, B. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "A Digital Design Flow for Secure Integrated Circuitsg", IEEE Trans. On CAD, pp. 1197~1208, vol. 25, no. 7, 2006.

[19] Espresso, http://embedded.eecs.berkeley.edu/.

Table 1: Experimental results.

Circuits	Inputs	Outputs	Ours (LC)	Minimum (LC)
5xp1	7	10	164	164
9sym (1)	9	1	464	462
alu4	14	8	21945	21945
apex3	54	50	7091	-
apex4	9	19	1664	1661
apex5	117	88	6741	-
b12	15	9	224	199
bw	5	28	43	43
clip	9	5	790	790
con1	7	2	32	32
cps	24	109	2756	2756
duke2	22	29	1755	1755
e64	64	64	2064	-
ex1010 (1)	10	10	4078	4078
ex5	8	63	343	343
f51m (1)	8	8	138	138
inc	7	9	131	131
misex1	8	7	48	48
misex2	25	18	377	377
misex3	14	14	24729	24729
misex3c	14	14	11707	11672
pdc	16	40	11583	11583
rd53	5	3	25	25
rd73	7	3	126	126
rd84	8	4	245	245
sao2	10	4	634	607
seq	41	35	17378	-
spla	16	46	11781	11781
squar5	5	8	33	33
t481	16	1	28	28
table3	14	14	14706	14706
table5	17	15	22009	22009
vg2	25	8	58280	-
xor5	5	1	0	0
z5xp1	7	10	164	164
z9sym (1)	9	1	462	462
Total (2)			**133184**	**133092**

(1) The FPRM expression has a minimum LC but not a minimum number of product terms.
(2) The total excludes those undetermined minimums.

2009 22nd International Conference on VLSI Design

A Novel Sustained Vector Technique for the Detection of Hardware Trojans

Mainak Banga and Michael S. Hsiao
Bradley Department of Electrical and Computer Engineering
Virginia Tech, Blacksburg, Virginia - 24061
Email: {banga, mhsiao}@vt.edu

***Abstract*—Intentional tampering in the internal circuit structure by implanting Trojans can result in disastrous operational consequences. While a faulty manufacturing leads to a nonfunctional device, effect of an external implant can be far more detrimental. Therefore, effective detection and diagnosis of such maligned ICs in the post silicon testing phase is imperative, if the parts are intended to be used in mission critical applications. We propose a novel sustained vector methodology that proves to be very effective in detecting the presence of a Trojan in an IC. Each vector is repeated multiple times at the input of both the genuine and the Trojan circuits that ensures the reduction of extraneous toggles within the genuine circuit. Regions showing wide variations in the power behavior are analyzed to isolate the infected gate(s). Experimental results on ISCAS benchmark circuits show that this approach can magnify the behavioral difference between a genuine and infected IC up to thirty times as compared to the previous approaches.** [1]

I. Introduction

With the decreasing per component cost for silicon ICs, companies are searching for new avenues to reduce the manufacturing cost. This has led to the outsourcing of the fabrication process. Consequently, the question of security and integrity of the embedded product comes forth as a prime concern. Thus, the design company has to ensure that no subtle intentional alterations had been subjected to the original logic during fabrication. The tiny circuits that are implanted to the original design to make it work contrary to the expected in certain rare and critical situations are called as *Trojans*.

Trojans have been common in the software domain and are commonly referred to as *virus*. Although *viruses* and *Trojans* are not exactly the same, their end consequences are similar. *Viruses* are necessarily malicious and interfere with the normal operation of the host on which they reside, whereas *Trojans* are passive monitors for most part of their operational life cycle until they are triggered. Solutions to counteract *virus* attacks exist in the form of *anti-virus* softwares. But currently there are no such remedies for Trojan attacks in hardware.

Trojans have distinguishing features that make them unique. They are stealthy in nature, which implies that they do not manifest their presence in normal operational conditions of the IC. This also suggests that they are not associated with internal gates that are either highly controllable or highly observable. They are very small in size, occupying only a small fraction of chip area which enables the third party vendor to accommodate them in the same die without altering the physical dimensions of the chip. Although test patters generated from an ATPG can detect most of the manufacturing faults, no such scheme is available to uncover the Trojans because they have an unknown triggering scenario which is difficult to assess and occurs rarely. Additionally, Trojans can have varying spatial locations on the IC and different logical behaviors (counter-based Trojans, sequence-detector Trojans etc.) [6] which complicate the detection mechanism. Finally, Trojans may not affect any of the primary outputs even if they are triggered. As a result, irrespective of whether a Trojan resides in an IC or not, there may not be a difference in the circuit's output behavior. Trojans can be selectively implanted and its absence in one IC does not guarantee its absence on any other. So destructive testing is also not a viable option. On one hand, destructive testing incurs a yield loss where a chip that has been cut open for analysis has to be discarded, while on the other it cannot guarantee genuineness of the other parts not subjected to such testing.

Moreover, on-chip testing structures like *Built In Self Test* (BIST) are also common to check on chip defects and reduce the test time [7], [8], [9]. However, similar to the problems with scan-based testing, BIST patterns may not be able to trigger the embedded Trojan which requires a specific sequence of input data. Hardware security based on *cryptography* and *public and private key* has been prevalent in industry. *Physically Unclonable Functions* (PUF) based structures have been proposed recently [11], [12] to characterize individual IC security key. While PUF based schemes are very effective in preventing external attacks to extract out the internal information from the ICs, Trojan attacks are on-chip intrusions and hence require a different approach. The intelligent nature of Trojans make them immune to such conventional checking procedures. This has directed the researchers to search for newer methods to detect the presence of Trojans. Since one has a limited access to the logical behavior of the device (only at the inputs and the outputs), researchers have identified the use of physical characteristics such as power, circuit delay or radiation behavior to act as a signature for the IC. Methods based on side channel signal analysis have been used in [1], [2] where the authors have used a random sequence of test patterns to differentiate between the actual and the Trojan circuits. However, the magnitude in the difference between the circuit under test (CUT) could be very small and may not be detectable considering process variations. With the process geometries sinking down in the nanometer regions, leakage and process variations continue to increase. Thus it is of utmost priority that the discrepancies in the genuine and maligned ICs

[1] Supported in part by NSF grant 0840936

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.22

are highlighted as much as possible so that their probability of detection increases.

In [3], [4], the authors have proposed partition mechanisms to isolate parts of the circuitry that might account for the Trojan. Results show that these methods are helpful in filtering out the infected regions. However, in those papers the Trojans are assumed to be associated with flip-flops of the circuit. This is a restricted assumption since embedded Trojans can monitor any signals in the circuit. Moreover, in [4], the regions under consideration are filtered out based on the flip-counts. A Trojan based on gate inputs may be missed using these techniques.

In this work, we propose a sustained vector methodology that magnifies the power consumption differences between the actual and the Trojan circuitry to a value which is much higher than the process variation. In certain cases the power differential can vary by more than an order of magnitude. Each vector is repeated multiple times to both the genuine and the Trojan-embedded circuits that ensures the reduction of toggles within the genuine circuit. This is needed so that the power dissipation outside of the Trojan will not drown out the extra power from the Trojan. In addition, we propose a scheme to suggest the locations susceptible to Trojan implantations. Regions showing wide variations in the power behavior are analyzed to isolate the infected gate(s). Our method is generalized for detecting Trojans that may be connected to any gate(s) in the circuit. There is no pre-silicon on-chip processing requirement which means that the methodology has no silicon overhead.

The rest of the paper is organized as follows. Section II gives the background on side channel analysis and kinds of Trojans. Section III details our approach. Section IV describes the Trojans inserted in our experimental setup. Section V discusses the results, and Section VI concludes the paper.

II. Preliminaries

In this section we present a brief overview of the concepts and terms that we will use throughout this paper.

A. Side Channel Analysis

With the improvement in cryptographic algorithms it is becoming increasingly difficult for the attackers to infringe the system dynamics in the conventional manner. This has forced them to explore other methods to access internal information. Physical parameters such as timing information, power consumption or electromagnetic leaks (side channel signals) have been shown to provide valuable information about the internal operations going on inside the system. Since side channel signals contain useful information which can reveal the internal functionality of the operating device, it is employed as a powerful non-destructive method of system analysis. The process of extracting internal data from a concealed system by tracking one or more side channel signals is called as *side channel analysis*.

In our work we have used power consumption as our side channel signal. The total dynamic power consumed in an IC is proportional to the operating frequency *f*, the switching capacitance *C* and the supply voltage *V* and are related by the following equation [5]:

$$\mathbf{P = CV^2f} \tag{1}$$

As the supply voltage *V* and the operating frequency *f* remain constant for an IC, the parameter for analysis is the switching capacitance *C*. This depends on the number of gates that undergoes toggle(s) in a system for any given vector pair.

B. Trojan Types

Combinational Trojan: An embedded combinational circuit that monitors static signal conditions with no memory of the signal's previous conditions.

Sequential Trojan: An embedded finite state machine (FSM) that is triggered upon the appearance of a specific sequence(s) of internal signal conditions.

The Trojans in our experiments are sequential Trojans and have been further categorized according to the toggling frequency of the signals from which they receive their inputs. More details will be given later in the paper.

C. Power Profile

A power profile represents the pattern of power consumption in a system. Power consumption for any pair of vectors is dependent on the total number of gates that switch which accounts for the switching capacitance (other factors in Equation 1 remaining the same). In our paper we have used the terms activity profile or power profile interchangeably because number of gate switches in a circuit is directly proportional to the dynamic power consumed by it. Also, a *gate* refers to any combinational or sequential element in the CUT.

III. Our Approach

There are two steps in our approach. The first step aims to detect the presence of a Trojan while the second tries to isolate the region within the circuit that may contain it. We call the first step as *Toggle Minimization* and the second as *Infected Region Isolation* respectively.

A. Step 1: Toggle Minimization

In the context of earlier discussion, power consumed in a circuit depends on the amount of signal switchings for any given vector pair. Since Trojans are minuscule circuits relative to the entire circuit, it is intuitive that the power consumed by the Trojan will also be very small. To observe the extra power that is contributed by the Trojan circuit over the genuine circuit, it is essential that the overall power consumption in the genuine circuit should be minimized. This would highlight the power contributed from the Trojan circuit which is the key to detect its presence. In the process, we also need to ensure that there is at least some kind of activity going on inside the circuit. In other words, the circuit should not be allowed to enter some *sleep mode*, which may also make the Trojan dormant. Sophisticated tools to measure very low on-chip currents are available which can sample the input power pins of the chip and measure the current [13].

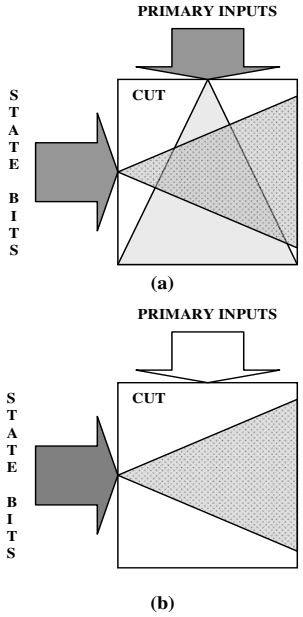


Fig. 1. Concept of activity minimization. In (a) circuit activity is created by both flip-flops and PIs whereas in (b) only by flip-flops

Circuit activity within the combinational frame of the circuit is induced in two ways: (1) with the changing inputs and (2) with the changing state. While primary inputs are fully controllable, the state variables are not. In order to limit the switching activity within the circuit, we can restrict the input variations to an extent such that the state variables are the only factor for inducing toggles. This is achievable by sustaining the same vector at the input pins over multiple clock cycles. Statistically in a purely random scenario each new vector will have at least half of the input bits toggled from the previous vector. These toggles will propagate through the transitive fanout cones of the respective inputs to generate further toggles in the circuit. If we ensure not to create any toggles at the input itself, it helps us reduce the circuit activity to a good extent because in such situation the state bits are the only factor for generating activity in the circuit. Moreover, we prefer scenarios where fewer state bits change as we keep the input vector at a stable value. Additionally, it also helps in reducing the synergistic transitions. That is, there are gates in the circuit which derives its inputs from the state-bits as well as from the inputs and they transition when more than a single input changes. If the changing state is the only dynamic variable during the operational mode, chances are less that such synergistic transitions will occur as compared with the random scenarios. Naturally these help in minimizing the overall circuit activity and keep the power consumption of the overall circuit low, which is our primary objective. We note that without sustaining a vector, the power consumption generally is much higher, closer to consuming an average power level of the circuit. The concept of *Toggle Minimization* by sustaining a vector is shown in Figures 1 (a) and (b).

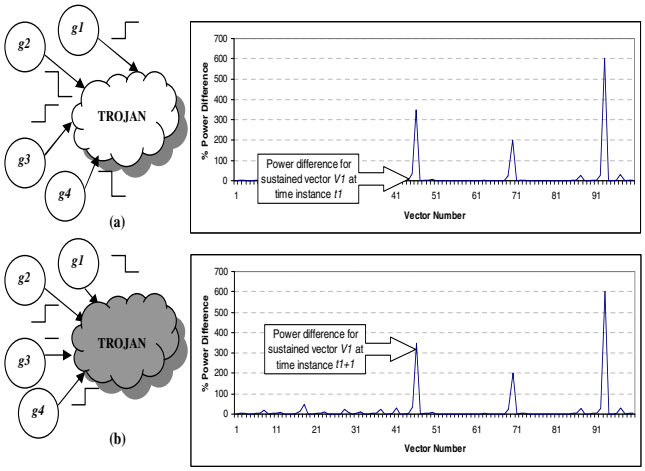


Fig. 2. Power differential measurement between vectors to isolate the gates connected to the Trojan

In our experiments, we generated a set of 1000 random input vectors, each of which is sustained to a maximum of 25 cycles. For a vector *V*, after sustaining it k times ($k < 25$), if we find that the system has reached a stable state where no further change in the state variables occurs, we move on to the next vector. Thus, a vector set for any particular circuit contains a maximum of 25000 input sequences. While holding the same vector at its input helps the circuit to traverse a local state space, changing the input vector to sustain next serves as a jump to explore some other regions of the state space. We apply this test sequence to both the genuine and the Trojan circuits in our experiments and obtain the *differential power* numbers (expressed in %) between them. The resultant plot is the *Differential Power Profile Plot* for the CUT.

B. Step 2: Infected Region Isolation

We use the *Differential Power Profile Plot* information from Step 1 to identify the region(s) of the circuit that are likely to be insertion points of the Trojan. We focus on vector pairs that produced high *differential power* as starting points.

Let us consider a sustained vector V_1 with which the CUTs shows a noticeable difference in the power profile in simulation cycles *t* and *t+1*. This is shown in Figure 2. Let g_1 to g_4 be the four internal gates in the circuit that are actually connected to the Trojan. Since the Trojan derives its inputs from the gates in its transitive fanin cone, any activity produced in the Trojan implies activity in its inputs. From the lower portion of the figure it shows that as the gates g_1, g_2, g_3 and g_4 underwent transition from 1010 to 0111, there is an observable difference in the *differential power* in the circuit. Thus a transition in gates g_1, g_2 and g_4 gives rise to a significant increase in the differential power ratio between the Trojan circuit and the genuine circuit, marking these gates as a potential suspect for the Trojan implantation.

We keep two counters for each gate: *TrojanCount* and *NonTrojanCount*. Each time a gate toggles by a vector pair

Algorithm 1 Isolate gate(s) accountable for Trojan activity

Require: *GenuineCkt*, *TrojanCkt*, *InputVector*

Ensure: Plot of *gate weights* corresponding to their probability of *Trojan association*

1: $PowerDifferentialThreshold \Leftarrow 5.0$
2: $TrojanCount \Leftarrow 0$
3: $NonTrojanCount \Leftarrow 0$
4: $ToggledGateList \Leftarrow Simulate(\textbf{\textit{GenuineCkt}}, \textbf{\textit{InputVector}})$
5: $PowerNumbers(\textbf{\textit{GenuineCkt}}) = PowerSimulate(\textbf{\textit{GenuineCkt}}, \textbf{\textit{InputVector}})$
6: $PowerNumbers(\textbf{\textit{TrojanCkt}}) = PowerSimulate(\textbf{\textit{TrojanCkt}}, (InputVector))$
7: **for all** $(V_i, V_{i+1}) \epsilon$ ***InputVector*** **do**
8: $\%PowerDifferential \Leftarrow (\textbf{abs}\frac{(PowerNumbers(\textit{TrojanCkt},(V_i,V_{i+1}) - PowerNumbers(\textit{GenuineCkt},(Vi,V_{i+1}))))}{PowerNumbers((GenuineCkt),(V_i,V_{i+1}))} * 100)$
9: **if** $\%PowerDifferential > PowerDifferentialThreshold$ **then**
10: $IncrementWeight(TrojanCount, ToggledGateList(V_i, Vi+1))$
11: **else**
12: $IncrementWeight(NonTrojanCount, ToggledGateList(V_i, Vi+1)$
13: **end if**
14: **end for**
15: $BuildPowerProfilePlots(PowerNumbers(\textbf{\textit{GenuineCkt}}), PowerNumbers(\textbf{\textit{TrojanCkt}}))$
16: **for all** $g_i \epsilon$ ***GenuineCkt*** **do**
17: $GateWeight = \frac{TrojanCount(g_i)}{NonTrojanCount(g_i}$
18: **end for**

TABLE I
FUNCTIONS OF ALGORITHM 1

Function	Purpose
Simulate(*Ckt*, *Vector*)	Simulate *Vector* set on given *Ckt*
PowerSimulate(*Ckt*, *Vector*)	Simulate *vector* set on given *Ckt* to compute the power numbers
BuildPowerProfilePlots(PowerNumbers(*Ckt1*), PowerNumbers(*Ckt2*))	Plot the differential power between *Ckt1* and *Ckt2*
IncrementWeight(*CountArray*, ToggledGateList(V_i, V_{i+1})	Increment weights of all gates that toggled between V_i and V_{i+1} in *CountArray*

that shows *differential power* greater than the *PowerDifferentialThreshold* (which is set to 5%, a typical value of process variation [1]), its *TrojanCount* is incremented and vice versa. After this analysis is over we compute the ratio *TrojanCount*/*NonTrojanCount* which is called as *gate weight*. A high value of the *gate weight* indicates that the gate(s) are most likely to be associated with the Trojan.

The entire procedure is outlined in Algorithm 1 and the functions used in the algorithm are explained in Table I. *PowerDifferentialThreshold* is the maximum differential power accounted for process variation. The two counters *TrojanCount* and *NonTrojanCount* are as described above. Evidently, a highly active gate which is not associated with the Trojan will most likely toggle between vectors when power differential is above the threshold. In that case its *TrojanCount* will increase as per our algorithm. To compensate for such spurious increments, we keep the *NonTrojanCount* which increments the count of a gate which toggles when the *Differential Power* is below *PowerDifferentialThreshold*. Whenever the number of points above the threshold is less than the points below the threshold for a given gate, the ratio of these two parameters would turn out to be small, thereby filtering out highly active signals that may not be associated with the Trojan.

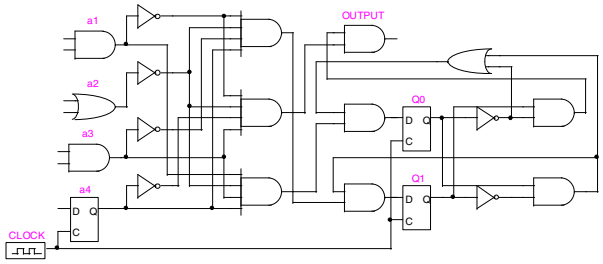


Fig. 3. Example of a Trojan circuit

IV. TROJAN DESCRIPTION

Typical Trojan structures used in our experimental setup is shown in Figure 3, and its size is less than 1% of the original circuit for large circuits. For small circuits, we keep it less than 3%. The sequence that the Trojan attempts to detect is **1011, 0001 and 0010** in this order. The Trojan inputs $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$ and $\mathbf{a_4}$ as well as the Trojan output referred as **OUTPUT** is connected to internal nets. Table II shows the sizes of the Trojans in terms of percentage gate counts used in our experiments.

We also insert the Trojan to different locations in the original circuit. We classify the gates in the circuit into three different classes depending on their activity. To compute the

TABLE II
TROJAN SIZE (% OF TOTAL GATE COUNT)

Circuit	High-activity	Medium-activity	Low-activity
s1196	2.61	2.96	2.61
s5378	0.66	0.56	0.56
s9234	0.29	0.32	0.36
s15850	0.19	0.20	0.18
s38584	0.09	0.09	0.10

TABLE III
AVERAGE % ACTIVITY OF GATES ASSOCIATED WITH TROJAN

Circuit	High-activity	Medium-activity	Low-activity
s1196	40.4	18.8	5.3
s5378	48.2	37.4	10.6
s9234	49.7	25.5	11.6
s15850	37.7	27.7	9.9
s38584	41.8	31.7	0.4

activity of a gate in the circuit we apply a sequence of 10000 non-sustained random vectors to the circuit and then compute the toggle count of each gate. Each signal is classified as High-activity, Medium-activity or Low-activity. After classifying the signals, we embed Trojans with the groups identified. The inputs to a Trojan can come from any of the three categories. Table III shows the average percentage activity at the inputs of the Trojans for our setup.

In order to make sure that the Trojan is really stealthy we choose a triggering sequence which is very rarely occurring within the set of signals chosen. To make it undetectable we connect the Trojan output(s) to an internal gate for which the non-triggered value at the output(s) of the Trojan is non-controlling for the gate to which it is input. For instance, if the Trojan produces an output of 0 under normal operating conditions, this net is connected to an OR/NOR gate so that it does not produce any effect unless the Trojan is triggered. Further, in Step 1 of the algorithm, we check that even when the Trojan is triggered, the effect does not reach any primary output, which otherwise would be a functionally detectable Trojan. This is possible because any controlling value in the propagation path of the gate affected by the Trojan output would mask its effect.

V. EXPERIMENTAL RESULTS

The results are presented in two parts corresponding to the two steps in our methodology. Table IV shows the maximum *percentage power differential* between the genuine and Trojan circuits obtained from both random vectors and vectors generated by our approach. We embedded Trojans three different ways as discussed before: Trojans monitoring high-activity signals, medium-activity signals, and low-activity signals, respectively. *Rnd* and *Ours* used in the table stand for *Random Approach* and *Our Approach* respectively. It is evident that for nearly all cases our approach enhances the *percentage differential power*. For example, for both s1196 and s5378, more than an order of magnitude improvement was achieved using our approach when compared with random vectors. The enhanced power differential helps us to isolate the region where the Trojan may be embedded, which is discussed next.

TABLE IV
COMPARISON OF % POWER DIFFERENTIAL BETWEEN GENUINE AND TROJAN CIRCUITS ACHIEVED BY RANDOM AND OUR APPROACH

Trojan Type	High-activity		Medium-activity		Low-activity	
Circuit	Rnd	Ours	Rnd	Ours	Rnd	Ours
s1196	10.71	**300**	27.08	**650**	18.75	**600**
s5378	4.18	**133.33**	2.24	**18.18**	2.79	**7.46**
s9234	30.77	**50**	30.77	**50**	22.22	**66.66**
s15850	13.04	**38.7**	**10.07**	9.46	3.28	**5.68**
s38584	0.74	**4.62**	0.8	**1.66**	0.68	**6.52**

The results of the second part of our analysis is shown in Figures 4, 5, 6, 7 and 8. The *x-axis* represents the gate numbers and the *y-axis* represents the computed *gate weight*. Recall that relative *GateWeight* is a direct indication of the probability that the gate is connected to the Trojan. In larger circuits, gates have higher count value in the *NonTrojanCount* because *PowerDifferentialThreshold* is exceeded less frequently during the application of the sustained vectors. As a result, the fraction *GateWeight* generally decreases as the circuit size increases. Nevertheless, it is the relative weights that count. As we can see from Figure 8, only a fraction of Total gates of the circuit is actually assigned a weight which means that the algorithm sieves out most of the gates from the larger circuits. The dotted circle in the figures correspond to the gates that are indeed connected to the Trojan in the actual circuit. As evident, in most cases the Trojan gates weigh out to a high value relative to other gates.

There are a few cases in which this method was not able to make a distinction. Our analysis on such cases led to two reasons for such anomalies:

1) Not all signals connected to the Trojan undergo transition when the Trojan is switching. As an example, for signals g_1, g_2, g_3 and g_4 connected to the Trojan if the triggering sequence is 1010 followed by 1111, then corresponding to every such sequence occurring during simulation g_2 and g_4 will have higher count in *TrojanCount* than g_1 and g_3. So during *GateWeight* computation, g_2 and g_4 will get more weight than g_1 and g_3, and this effect will show up in the *GateWeights* plot.
2) The FSM nature of the Trojan may be such that the internal state of the Trojan may change even if the inputs to the Trojan are stable. In such a case, if the overall circuit activity happens to be low, the *differential power* between the CUTs will be highlighted. But the toggling signals corresponding to such high differentials need not necessarily be connected to the Trojan.

We note that for nearly all the test cases, our approach was able to identify the signals responsible for the Trojan, and the anomalies mentioned above occur infrequently.

VI. CONCLUSION

We have presented a novel sustained vector methodology that is very effective in detecting the presence of a Trojan.

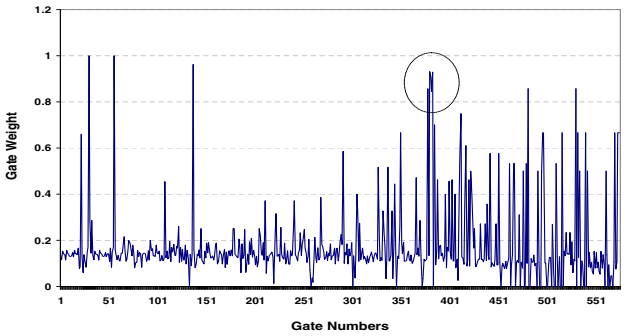


Fig. 4. Plot of Gate Weights for s1196 Medium-active Trojan

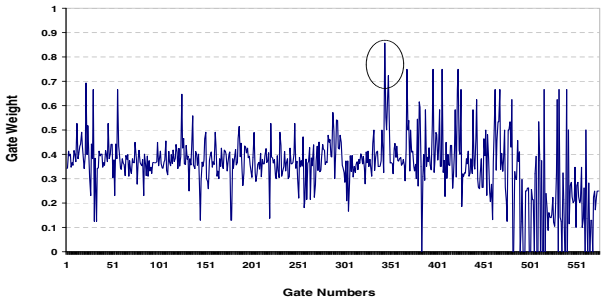


Fig. 5. Plot of Gate Weights for s1196 High-active Trojan

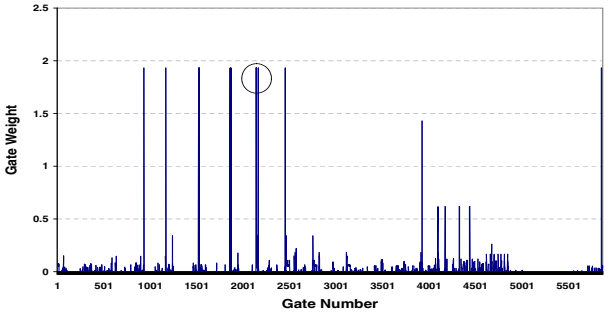


Fig. 6. Plot of Gate Weights for s9234 Low-active Trojan

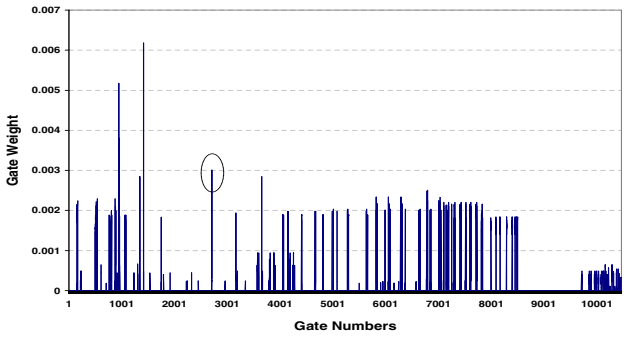


Fig. 7. Plot of Gate Weights for s15850 Low-active Trojan

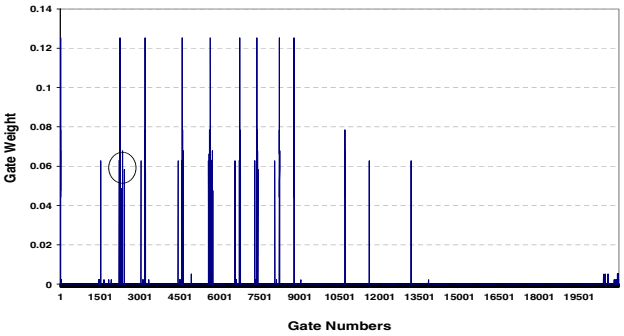


Fig. 8. Plot of Gate Weights for s38584 High-active Trojan

Even if the Trojan constitutes only a tiny fraction of the chip area, our experimental results show that this technique enhances the power profile difference between the genuine and Trojan circuits by up to more than one order of magnitude as compared with the random vectors. Furthermore, this method is effective irrespective of the activity behavior of the gates which is evident from the fact that it is successful in detecting most of the High-activity, Medium-activity and Low-activity Trojans in the benchmark circuits. Finally, in many cases, our approach was able to pinpoint the actual location of the Trojan in the circuit.

REFERENCES

[1] D. Agarwal, S. Baktir, D. Karakoy, P. Rohatgi and B. Sunar; *Trojan Detection using IC Fingerprinting*, IBM Research Report, 2006.

[2] K. Nowaka, G. Carpenter, F. Gebara, J. Schaub, D. Agarwal, P. Rohatgi, W. E. Hall, S. Baktir, D. Karakoyunlu and B. Sunar; *IC Fingerprinting and Stable IS Sensors for Enhanced IC Trust*,2006.

[3] M. Banga, M. Chandrasekar, L. Fang and M. Hsiao; *Guided Test Generation for Isolation and Detection of Embedded Trojans in ICs*, ACM Great Lake Symp. on Very Large Scale Integration, 2008, pp. 363-366.

[4] M. Banga and M. Hsiao; *A Region Based Approach for the Detection of Hardware Trojans*, IEEE Int. Wkshop on Hardware-Oriented Security and Trust, 2008, pp 43-50.

[5] S. Pilli and S. Sapatnekar; *Power estimation considering statistical IC parametric variations*, ISCAS 1997, pp. 1524 - 1527, vol.3.

[6] X. Wang, M. Tehranipoor and J. Plusquellic; *Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions*, IEEE Int. Wkshop on Hardware-Oriented Security and Trust, Jun 2008, pp 15-22.

[7] C. Fagot, O. Gascuel, P. Girard and C. Landrault; *On Calculating Efficient LFSR Seeds for Built-In Self Test*, Proc. Of European Test Wkshop, 1999, pp 7-14.

[8] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski; *Logic BIST for large industrial designs: real issues and case studies*, ITC, 1999, pp. 358-367.

[9] W. T. Cheng, M. Sharma, T. Rinderknecht and C. Hill; *Signature Based Diagnosis for Logic BIST*, ITC 2006, Oct. 2006, pp. 1 - 9.

[10] C. H. Kim and J. J. Quisquater, *How can we overcome both side channel analysis and fault attacks on RSA-CRT?*, Wkshop on Fault Diagnosis and Tolerance in Cryptography, 2007, pp. 21 - 29.

[11] J. Guajardo, S. S. Kumar, G.-J. Schrijen and P. Tuyls, *Physical Unclonable Functions and Public-Key Crypto for FPGA IP Protection*, Int. Conf. on Field Programmable Logic and Applications, Aug. 2007, pp. 189 - 195.

[12] B. Gassend, D. Clarke, M. van Dijk and S. Devadas, *Controlled physical random functions*, 18th Annual Proceedings of Computer Security Applications Conf., Dec. 2002, pp. 149 - 160.

[13] http://cp.literature.agilent.com/litweb/pdf/5988-0687EN.pdf

Session 5C

Embedded Systems II

2009 22nd International Conference on VLSI Design

Efficient Placement of Compressed Code for Parallel Decompression

Xiaoke Qin and Prabhat Mishra

Department of Computer and Information Science and Engineering
University of Florida, Gainesville FL 32611-6120, USA
{*xqin, prabhat*}*@cise.ufl.edu*

Abstract

Code compression is important in embedded systems design since it reduces the code size (memory requirement) and thereby improves overall area, power and performance. Existing researches in this field have explored two directions: efficient compression with slow decompression, or fast decompression at the cost of compression efficiency. This paper combines the advantages of both approaches by introducing a novel bitstream placement method. The contribution of this paper is a novel code placement technique to enable parallel decompression without sacrificing the compression efficiency. The proposed technique splits a single bitstream (instruction binary) fetched from memory into multiple bitstreams, which are then fed into different decoders. As a result, multiple slow-decoders can work simultaneously to produce the effect of high decode bandwidth. Our experimental results demonstrate that our approach can improve decode bandwidth up to four times with minor impact (less than 1%) on compression efficiency.

1 Introduction

Memory is one of the most constrained resources in an embedded system, because a larger memory implies increased area (cost) and higher power/energy requirements. Due to dramatic complexity growth of embedded applications, it is necessary to use larger memories in today's embedded systems to store application binaries. Code compression techniques address this problem by reducing the storage requirement of applications by compressing the application binaries. The compressed binaries are loaded into the main memory, then decoded by a decompression hardware before it's execution in a processor. Compression ratio is widely used as a metric of the efficiency of code compression. It is defined as the ratio (CR) between the compressed program size (CS) and the original program size (OS) i.e., CR = CS / OS. Therefore, a smaller compression ratio implies a better compression technique. There are two major challenges in code compression: i) how to compress the code as much as possible; and ii) how to efficiently decompress the code without affecting the processor performance.

The research in this area can be divided into two categories based on whether it primarily addresses the compression or decompression challenges. The first category tries to improve code compression efficiency using the state-of-the-art coding methods such as Huffman coding [1] and arithmetic coding [2]. Theoretically, they can decrease the compression ratio to its lower bound governed by the intrinsic entropy of code, although their decode bandwidth usually is limited to 6-8 bits per cycle. These sophisticated methods are suitable when the decompression unit is placed between the main memory and cache (pre-cache). However, recent research [3] suggests that it is more profitable to place the decompression unit between the cache and the processor (post-cache). In this way the cache retains data still in a compressed form, increasing cache hits, therefore achieving potential performance gain. Unfortunately, this post-cache decompression unit actually demands much more decode bandwidth than what the first category of techniques can offer. This leads to the second category of research that focuses on higher decompression bandwidth by using relatively simple coding methods to ensure fast decoding. However, the efficiency of the compression result is compromised. The variable-to-fixed coding techniques [12] are suitable for parallel decompression but it sacrifices the compression efficiency due to fixed encoding.

In this paper, we combine the advantages of both approaches by developing a novel bitstream placement technique which enables parallel decompression without sacrificing the compression efficiency. This paper makes two important contributions. First, it is capable of increasing the decode bandwidth by using multiple decoders to work simultaneously to decode a single/adjacent instruction(s). Second, our methodology allows designers to use any existing compression algorithms including variable-length encodings with little or no impact on compression efficiency.

The rest of the paper is organized as follows. Section 2 introduces related work addressing code compression for embedded systems. Section 3 describes our code compression and bitstream placement methods. Section 4 presents our experimental results. Finally, Section 5 concludes the paper.

2 Related Work

A great deal of work has been done in the area of code compression for embedded systems. The basic idea is to take one or more instruction as a symbol and use common coding methods to compress the code. Wolfe and Chanin [1] first proposed the Huffman-coding based code compression approach. A Line Address Table (LAT) is used to handle the addressing of branching within compressed code. Lin et al. [4] uses LZW-based code compression by applying it to variable-sized blocks of VLIW codes. Liao [5] explored dictionary-based compression techniques. Lekatsas et al. [2] constructed SAMC using arithmetic coding based compression. These approaches significantly re-

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.71

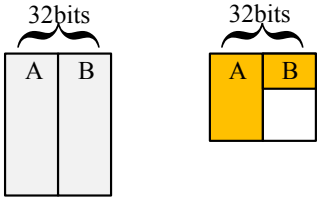
duces the code size but their decode (decompression) bandwidth is limited.

To speed up the decode process, Prakash et al. [6] and Ros et al. [7] improved conventional dictionary based techniques by considering bit changes of a 16-bit or 32-bit vectors. Seong et al. [8] further improved these approaches using bitmask based code compression. These techniques enable fast decompression but they achieve inferior compression efficiency compared to those based on well established coding theory.

Instead of treating each instruction as a single symbol, some researchers observed that the number of different opcodes and operands are quite smaller than that of entire instructions. Therefore, a division of a single instruction into different parts may lead to more effective compression. Nam et al. [9] and Lekatsas et al. [10] broke instructions into several fields then employed different dictionary to encode them. CodePack [11] divided each MIPS instruction at the center, applied two prefix-dictionary to each of them, then combined the encoding results together to create the finial result. However, in their compressed code, all these fields are simply stored one after another (in a serial fashion). The variable-to-fixed coding technique [12] is suitable for parallel decompression but it sacrifices the compression efficiency due to fixed encoding. The variable size encodings (fixed-to-variable and variable-to-variable) can achieve the best possible compression. However, it is impossible to use multiple decoders to decode each part of the same instruction simultaneously, when variable length coding is used. The reason is that the beginning of next field is unknown until the decode of the current field ends. As a result, the decode bandwidth cannot benefit very much from such an instruction division. Our approach allows variable length encoding for efficient compression and proposes a novel placement of compressed code to enable parallel decompression.

3 Efficient Placement of Compressed Binaries

Our work is motivated by previous variable length coding approaches based on instruction partitioning [9, 10, 11] to enable parallel compression of the same instruction. The only obstacle preventing us from decoding all fields of the same instruction simultaneously is that the beginning of each compressed field is unknown unless we decompress all previous fields.



(a) Uncompressed code (b) Compressed code

Figure 1. Intuitive placement for parallel decompression.

One intuitive way to solve this problem, as shown in Figure 1, is to separate the entire code into two parts, compress each of them separately, then place them separately. Using such a placement, the different parts of the same instruction can be decoded simultaneously using two pointers. However, if one part of the code (part B) is more effectively compressed than the other one (part A), the remaining unused space for part B will be wasted. Therefore, the overall compression ratio will be hampered remarkably. Furthermore, the identification of branch targets will also be a problem due to the unequal compression. As mentioned earlier, fixed length encoding methods are suitable for parallel decompression but it sacrifices the compression efficiency due to fixed encoding. The focus of our research is to enable parallel decompression for binaries compressed with variable length encoding methods.

The basic idea of our approach to handle this problem is to develop an efficient bitstream placement method. Our method enables the compression algorithm to make maximum usage of the space automatically. At the same time, the decompression mechanism will be able to determine which part of the newly fetched 32 bits should be sent to which decoder. In this way, we exploit the benefits of instruction division in both compression efficiency and decode bandwidth.

3.1 Overview of Our Approach

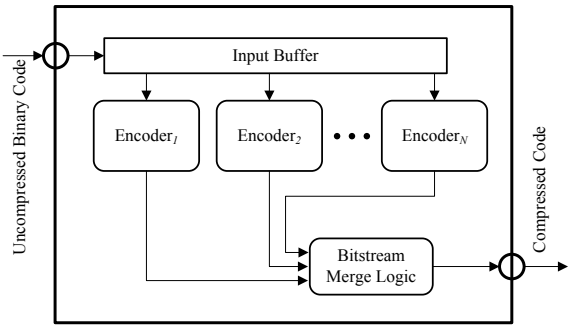
In our approach, we use branch blocks[1] [4] as the basic unit of compression. In other words, our placement technique is applied to each branch blocks in the application. Figure 2 shows the block diagram of our proposed compression framework. It consists of four main stages: compression (encode), bitstream merge, bitstream split and decompression (decode).

During compression (Figure 2a), we first break every input storage block (containing one or more instructions) into several fields, then apply specific encoders to each one of them. The resultant compressed streams are combined together by a bitstream merge logic based on a carefully designed bitstream placement algorithm. Note that the bitstream placement cannot rely on any information invisible to the decompression unit. In other words, the bitstream merge logic should merge streams based on only the binary code itself and the intermediate results produced during the encoding process.

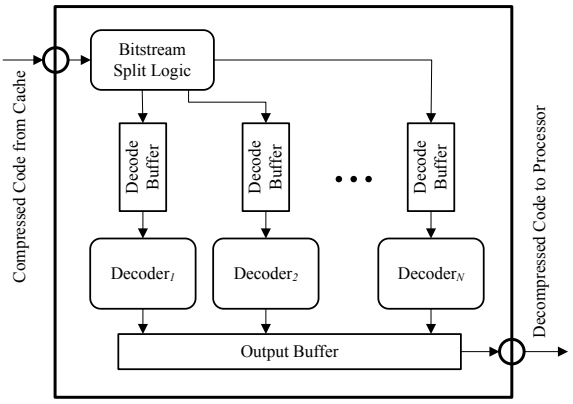
During decompression (Figure 2b), the scenario is exactly the opposite of compression. Every word fetched from the cache is first split into several parts, each of which belongs to a compressed bitstream produced by some encoder. Then the split logic dispatches them to the buffers of correct decoders, according to the bitstream placement algorithm. These decoders decode each bitstream and generate the uncompressed instruction fields. After combining these fields together, we obtain the final decompression result, which should be identical to the corresponding original input storage block (containing one or more instructions).

From the viewpoint of overall performance, the compression algorithm affects the compression ratio and decompression speed in an obvious way. Nevertheless, the bitstream placement actually governs whether multiple decoders are capable to work in parallel. In previous works, researchers tend to use a very simple placement technique: they appended the compressed code for each symbol one after the other. When variable length coding is used, symbols must be decoded in order. In this paper,

[1] The instructions between two consecutive branch targets.



(a) Compression technique



(b) Decompression mechanism

Figure 2. Proposed code compression framework.

we demonstrate how a novel bitstream placement enables parallel decoding and boosts the overall decode performance. The remainder of this section describes the four important stages in our framework: compression, bitstream merge, bitstream split and decompression.

3.2 Compression Algorithm

In our current implementation, we use Huffman coding as the compression algorithm of each single encoder (Encoder$_1$ - Encoder$_N$ in Figure 2 (a)), because Huffman coding is optimal for a symbol-by-symbol coding with a known input probability distribution. To improve its performance on code compression, we modify the basic Huffman coding method [1] in two ways: i) instruction division and ii) selective compression. As mentioned earlier, any compression technique can be used in our framework.

Similar to previous works [9, 10, 11], we believe that compressing different parts of a single instruction separately is profitable, because the number of distinct opcodes and operands is far less than the number of different instructions. We have observed that for most applications it is profitable to divide the instruction at the center. In the rest of this paper, we will use this division pattern, if not stated otherwise.

Selective compression is a common choice in many compression techniques [8]. Since the alphabet for binary code compression is usually very large, Huffman coding may produce many dictionary entries with quite long keywords. This is harmful to the overall compression ratio, because the size of the dictionary entry must also be taken into account. Instead of using bounded Huffman coding, we address this problem using selective compression. First, we create the conventional Huffman coding table. Then we remove any entry e which does not satisfy Equation 1.

$$(Length(Symbol_e) - Length(Key_e)) * Time_e > Size_e, \quad (1)$$

Here, $Symbol_e$ is the uncompressed symbol (one part of an instruction), Key_e is the key of $Symbol_e$ created by Huffman coding, $Time_e$ is the total time for which $Symbol_e$ occurs in the uncompressed code, and $Size_e$ is the space required to store this entry. For example, two unprofitable entries from Dictionary II (Figure 3) are removed.

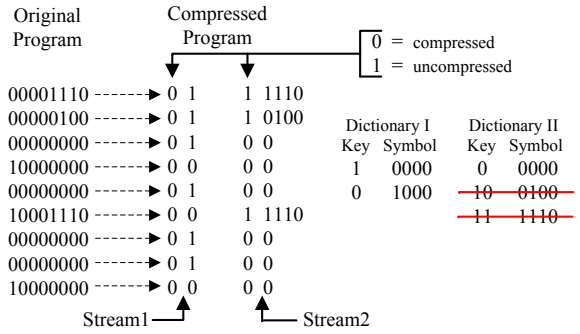


Figure 3. Code compression using modified Huffman coding

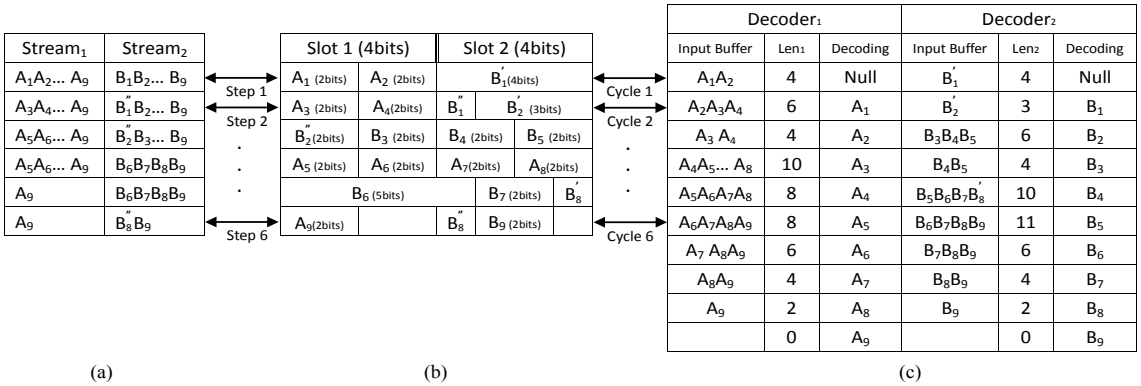
Once the unprofitable entries are removed, we use remaining entries as the dictionary for both compression and decompression. Figure 3 shows an illustrative example of our compression algorithm. For the simplicity of illustration, we use 8-bit binaries instead of 32 bits used in real applications. We divide each instruction in half and use two dictionaries, one for each part. The final compressed program is reduced from 72 bits to 45 bits. The dictionary requires 15 bits. The compression ratio for this example is 83.3%. The two compressed bitstreams (Stream1 and Stream2) are also shown in Figure 4.

Stream1		Stream2	
Symbol	Value	Symbol	Value
A_1	01	B_1	11110
A_2	01	B_2	10100
A_3	01	B_3	00
A_4	00	B_4	00
A_5	01	B_5	00
A_6	00	B_6	11110
A_7	01	B_7	00
A_8	01	B_8	00
A_9	00	B_9	00

Figure 4. Two compressed bitstreams from the code compression example in Figure 3

3.3 Bitstream Merge

The bitstream merge logic merges multiple compressed bitstreams into a single bitstream for storage. We first explain



(a) (b) (c)

Figure 5. Bitstream placement using two bitstreams in Figure 4. (a) Unplaced data remaining in the input buffer of merge logic, (b) Bitstream placement result, (c) Data within Decoder$_1$ and Decoder$_2$ when current storage block is decompressed[2].

some basic models and terms which we will use in the following discussion. Next, we describe the working principle of our bitstream merge logic.

Definition 1: **Storage block** is a block of memory space, which is used as the basic input and output unit of our merge and split logic. Informally, a storage block contains one or more consecutive instructions in a branch block. Figure 6 illustrates the structure of a storage block. We divide it into several slots. Each of them contains adjacent bits extracted from the same compressed bitstream. In our current implementation, all slots within a storage block have the same size.

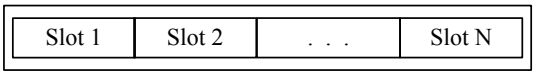


Figure 6. Storage block structure

Definition 2: **Sufficient decode length (SDL)** is the minimum number of bits required to ensure that at least one compressed symbol is in the decode buffer. In our implementation, this number equals one plus the length of an uncompressed instruction field.

Our bitstream merge logic performs two tasks to produce each output storage block filled with compressed bits from multiple bitstreams: i) use the given **bitstream placement algorithm** (BPA) to determine the bitstream placement within current storage block; ii) count the numbers of bits left in each buffer as if they finish decoding current storage block. We pad extra bits after the code at the end of the stream to align on a storage block boundary.

Algorithm 1 is developed to support parallel decompression of two bitstreams. The goal is to guarantee that each decoder has enough bits to decode in the next cycle after they receive the current storage block. Figure 5 illustrates our bitstream merge procedure using previous code compression example in Figure 3. The size of storage blocks and slots are 8 bits and 4 bits respectively. In other words, each storage block has two slots. The SDL is 5. When the merge process begins (translates Figure 5a to Figure 5b), the merge logic gets A_1, A_2 and B_1', then assigns them to the first and second slots[2]. Similarly, A_3, A_4, B_1'' and B_2' are placed in the second iteration (step 2). When it comes to the third output block, the merge logic finds that after Decoder$_2$ receives and processes the first two slots, there are only 3 bits left in its buffer, while Decoder$_1$ still has enough bits to decode in the next cycle. So it assigns both slots in the third output block from Stream$_2$. This process repeats until both input (compressed) bitstreams are placed. The "Full()" checks are necessary to prevent the overflow of decoders' input buffers. Our merge logic automatically adjusts the number of slots assigned to each bitstream, depending on whether they are effectively compressed.

[2]We use $'$ and $''$ to indicate the first and second parts of the same compressed instruction in case it does not fit in the same storage block.

Algorithm 1: Placement of Two Bitstreams
Input: Every Storage Block
Output: Placed Bitstreams

if *this is the first block* **then**
  Assign Stream 1 to Slot 1 and Stream 2 to Slot 2
else
  if *!Ready(1) and !Ready(2)* **then**
    Assign Stream 1 to Slot 1 and Stream 2 to Slot 2
  else if *!Ready(1) and Ready(2)* **then**
    Assign Stream 1 to Slot 1 and 2
  else if *Ready(1) and !Ready(2)* **then**
    Assign Stream 2 to Slot 1 and 2
  else if *!Full(1) and !Full(2)* **then**
    Assign Stream 1 to Slot 1 and Stream 2 to Slot 2
end
$Ready(i)$ checks whether the i^{th} decoder's buffer contains at least SDL bits.
$Full(i)$ checks whether corresponding buffer has enough space to hold more slots.

3.4 Bitstream Split

The bitstream split logic uses the reverse procedure of the bitstream merge logic. The bitstream split logic divides the single compressed bitstream into multiple streams using the following guidelines:

- Use the given BPA to determine the bitstream placement within current compressed storage block, then dispatch dif-

ferent slots to the corresponding decoder's buffer.
- If all the decoders are ready to decode the next instruction, start the decoding.
- If the end of current branch block is encountered, force all decoders to start.

We use the example in Figure 5 to illustrate the bitstream split logic. When the placed data in Figure 5b is fed to the bitstream split logic (translates Figure 5b to Figure 5c), the length of the input buffers for both streams are less than SDL. So the split logic determines the first and the second slot must belong to Stream1 and Stream2 respectively in the first two cycles. At the end of the second cycle, the number of bits in the $Decoder_1$ buffer, Len_1 (i.e., 6), is greater than SDL (i.e., 5), but Len_2 (i.e., 3) is smaller than SDL. This indicates that both slots must be assigned to the second bitstream in the next cycle. Therefore, the split logic dispatches both slots to the input buffer of Decoder_2. This process repeats until all placed data are split.

3.5 Decompression Mechanism

The design of our decoder is based on the Huffman decoder hardware proposed by Wolfe et al. [1]. The only additional operation is to check the first bit of an incoming code, in order to determine whether it is compressed using Huffman coding or not. If it is, decode it using the Huffman decoder; otherwise send the rest of the code directly to the output buffer. Therefore, the decode bandwidth of each single decoder (Decoder_1 to Decoder_N in Figure 2 (b)) should be similar to the one given in [1]. Since each decoder can decode 8 bits per cycle, two parallel decoders can produce 16 bits per cycle. Decoders are allowed to begin decoding only when i) all decoders' decoder buffers contains more bits than SDL; or ii) bitstream split logic forces it to begin decoding. After combining the outputs of these parallel decoders together, we obtain the final decompression result.

3.6 Bitstream Placement for Four Streams

In order to further boost the output bandwidth, we have also developed a bitstream placement algorithm which enables four Huffman decoders to work in parallel. During compression, we take every two adjacent instructions as a single input storage block. Four compressed bitstreams are generated by high 16 bits and low 16 bits of all odd instructions, as well as high 16 bits and low 16 bits of all even instructions. We also change the slot size within each output storage block to 8 bits, so that there are 4 slots in each storage block. We omit the complete description of this algorithm here due to the lack of space. However, the basic idea remains the same and it is a direct extension of Algorithm 1. The goal is to provide each decoder with sufficient number of bits so that none of them are idle at any point. Since each decoder can decode 8 bits per cycle, four parallel decoders can produce 32 bits per cycle.

Although we can still employ more decoders, the overall increase of output bandwidth will slow down by more start up stalls. For example, we have to wait 2 cycles to decompress the first instruction using four decoders in the worst case. As a result, high sustainable output bandwidth using too many parallel decoders may not be feasible, if its start up stall time is comparable with the execution time of the code block itself.

4 Experiments

The code compression and parallel decompression experiments of our framework are carried out using different application benchmarks compiled using a wide variety of target architectures.

4.1 Experimental Setup

We used benchmarks from MediaBench and MiBench benchmark suites: adpcm_en, adpcm_de, cjpeg, djpeg, gsm_to, gsm_un, mpeg2enc, mpeg2dec and pegwit. These benchmarks are compiled for four target architectures: TI TMS320C6x, PowerPC, SPARC and MIPS. The TI Code Composer Studio is used to generate the binary for TI TMS320C6x. GCC is used to generate the binary for the rest of them. Our computation of compressed program size includes the size of the compressed code as well as the dictionary and all other data required by our decompression unit.

We have evaluated the relationship between the division position and the compression ratio on different target architectures. We have observed that for most architectures, the middle of each instruction is usually the best partition position. We have also analyzed the impact of dictionary size on compression efficiency using different benchmarks and architectures. Although larger dictionaries produce better compression, our approach produces reasonable compression using only 4096 bytes for all the architectures. Based on these observations, we divide each 32-bit instruction from the middle to create two bitstreams. The maximum dictionary size is set to 4096 bytes. The output bandwidth of the Huffman decoder is computed as 8 bits per cycle [1] in our experiments. To the best of our knowledge, there have been no work on bitstream placement for enabling parallel decompression of variable length coding. So we compare our work (BPA1 and BPA2) with CodePack [11], which uses a conventional bitstream placement method. Here, **BPA1** is our bitstream placement algorithm in Algorithm 1, which enables two decoders to work in parallel, and **BPA2** represents our bitstream placement algorithm in Section 3.6, which supports four parallel decoders.

4.2 Results

Figure 7 shows the efficiency of our different bitstream placement algorithms. Here, "decode bandwidth" means the sustainable output bits per cycle after initial stalls. The number shown in the figure is the average decode bandwidth over all benchmarks. It is important to note that the decode bandwidth for each benchmark also shows the same trend. As expected, the sustainable decode bandwidth increases as the number of decoder grows. Our bitstream placement approach improves the decode bandwidth up to four times. As discussed earlier, it is not profitable to use more than four decoders since it will introduce more start up stalls.

We have studied the impact of bitstream placement on compression efficiency. Figure 8 compares the compression ratios between the three techniques on various benchmarks on MIPS

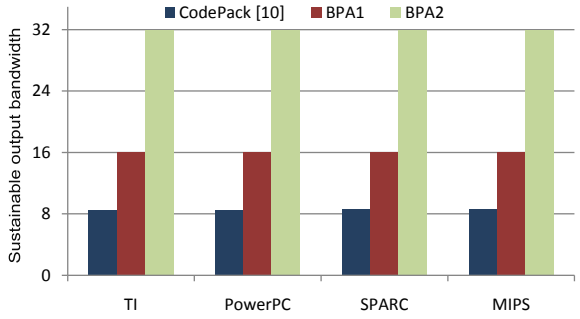


Figure 7. Decode bandwidth of different techniques

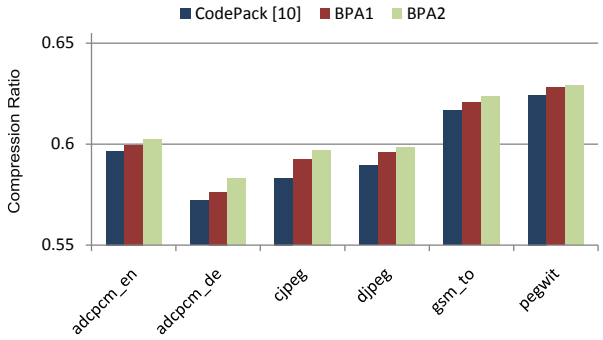


Figure 8. Compression ratio for different benchmarks

architecture. The results show that our implementation of bitstream placement has less than 1% penalty on compression efficiency. This result is consistent across different benchmarks and target architectures as demonstrated in Figure 9 which compares the average compression ratio of all benchmarks on different architectures.

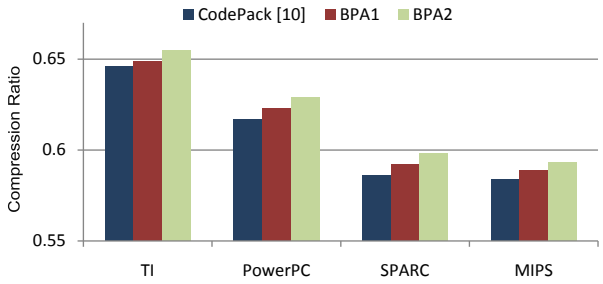


Figure 9. Compression ratio on different architectures

We have implemented the decompression unit using Verilog HDL. The decompression hardware is synthesized using Synopsis Design Compiler and TSMC 0.18 cell library. Table 1 shows the reported results for area, power, and critical path length. It can be seen that "BPA1" (uses 2 16-bit decoders) and CodePack have similar area/power consumption. On the other hand, "BPA2" (uses 4 16-bit decoders) requires almost double the area/power compared to "BPA1" to achieve higher decode bandwidth, because it has two more parallel decoders. The decompression overhead in area and power is negligible (100 to 1000 times smaller) compared to typical reduction in overall area and energy requirements due to code compression.

Table 1. Comparison using different placement algorithms

	CodePack [11]	BPA1	BPA2
Area/μm^2	122263	137529	253586
Power/mW	7.5	9.8	14.6
Critical path length/ns	6.91	5.76	5.94

5 Conclusions

Memory is one of the key driving factors in embedded system design since a larger memory indicates an increased chip area, more power dissipation, and higher cost. As a result, memory imposes constraints on the size of the application programs. Code compression techniques address the problem by reducing the program size. Existing researches have explored two directions: efficient compression with slow decompression, or fast decompression at the cost of the compression efficiency. This paper combines the advantages of both approaches by introducing a novel bitstream placement technique for parallel decompression. We addressed four challenges to enable parallel decompression using efficient bitstream placement: instruction compression, bitstream merge, bitstream split and decompression. Efficient placement of bitstreams allows the use of multiple decoders to decode different parts of the same/adjacent instruction(s) to enable the increase of decode bandwidth. Our experimental results using different benchmarks and architectures demonstrated that our approach improved the decompression bandwidth up to four times with less than 1% penalty in compression efficiency.

References

[1] A. Wolfe and A. Chanin, "Executing compressed programs on an embedded RISC architecture," *MICRO* 81-91, 1992.

[2] H. Lekatsas and Wayne Wolf, "SAMC : A code compression algorithm for embedded processors," *IEEE TCAD*, 18(12), 1999.

[3] H. Lekatsas, J. Henkel and W. Wolf, "Code compression for low-power embedded system design," *DAC*, 294–299, 2000.

[4] C. Lin, Y. Xie, and W. Wolf, "LZW-based code compression for VLIW embedded systems," *DATE*, 76–81, 2004.

[5] S. Liao, S. Devadas, and K. Keutzer, "Code density optimization for embedded DSP processors using data compression techniques," *IEEE TCAD*, 17(7), 601–608, 1998.

[6] J. Prakash et al., "A simple and fast scheme for code compression for VLIW processors," *DCC*, pp 444, 2003.

[7] M. Ros and P. Sutton, "A hamming distance based VLIW/EPIC code compression technique," *CASES*, 132–139, 2004.

[8] S. Seong and P. Mishra, "Bitmask-based code compression for embedded systems," *IEEE TCAD*, 27(4), 673–685, April 2008.

[9] S. Nam et al., "Improving dictionary-based code compression in VLIW architectures," *FECCS*, E82-A(11), 2318–2324, 1999.

[10] H. Lekatsas and W. Wolf, "Code compression for embedded systems," *DAC*, 516–521, 1998.

[11] C. Lefurgy, *Efficient Execution of Compressed Programs*, Ph.D. Thesis, University of Michigan, 2000.

[12] Y. Xie et al., "Code compression for embedded VLIW processors using variable-to-fixed coding," *IEEE TVLSI*, 14(5), 2006.

2009 22nd International Conference on VLSI Design

FPGA based High Performance Double-precision Matrix Multiplication

Vinay BY. Kumar Siddharth Joshi Sachin B. Patkar H. Narayanan
vinayby@iitb.ac.in {s_joshi, patkar, hn}@ee.iitb.ac.in
Department of Electrical Engineering
Indian Institute of Technology, Bombay
India, Mumbai - 400076

***Abstract*—We present two designs (I and II) for IEEE 754 double precision floating point matrix multiplication, an important kernel in many tile-based BLAS algorithms, optimized for implementation on high-end FPGAs. The designs, both based on the rank-1 update scheme, can handle arbitrary matrix sizes, and are able to sustain their peak performance except during an initial latency period. Through these designs, the trade-offs involved in terms of local-memory and bandwidth for an FPGA implementation are demonstrated and an analysis is presented for the optimal choice of design parameters. The designs, implemented on a Virtex-5 SX240T FPGA, scale gracefully from 1 to 40 processing elements(PEs) with a less than 1% degradation in the design frequency of 373 MHz. With 40 PEs and a design speed of 373 MHz, a sustained performance of 29.8 GFLOPS is possible with a bandwidth requirement of 750 MB/s for design-II and 5.9 GB/s for design-I.**

I. Introduction

Field Programmable Gate Arrays (FPGAs) are increasingly being seen as a promising avenue for High Performance Computing (HPC), especially with the introduction of high-end FPGAs like Virtex-4/5. These FPGAs are a very attractive choice due to their abundant local memory, embedded high-speed resources like DSP blocks, PCIe endpoints etc., and their reconfigurability and lower power consumption compared to general purpose hardware.

An inspection of Level-3 BLAS routines shows that matrix multiplication (*dgemm*) and triangular equation solution (*dtrsm*) form the building blocks for many important linear algebra algorithms, in fact, the *dtrsm* itself can be expressed in terms of *dgemm*. MEMOCODE-07 hosted a challenge problem requiring a hw/sw codesign based acceleration of complex integer matrix-multiply. Underwood [1] chose matrix multiplication as one of the three main routines for FPGA acceleration in order for HPC. Matrix-Multiplication, therefore, presents as an important candidate for hardware acceleration.

In this paper, we present designs for double precision floating point matrix multiplication, based on the rank-1 update algorithm, targeted at the Virtex-5 SX240T, a high-end Xilinx FPGA. As compared to others this algorithm enables better reuse of data from the input matrices. The processing elements (PEs) use off-the-shelf floating point operators from Xilinx Coregen, unlike other related work, resulting in advantages such as the choice of custom-precision, short-design time, portability across devices, better IEEE 754 compliance, etc. The PEs are designed so as to scale linearly in terms of resources with negligible (<1%) degradation in speed. Some of the recent work [2] reports 35% speed reduction associated with scaling, which is typically due to increased routing complexity. The proposed design(II) is tolerant to burst-like input which suits well with a high performance I/O bus like PCIe – allowing it to scale seamlessly across multiple FPGAs.

The under utilisation of device primitives and the over-dependence on the distributed memory available in FPGAs results in lower performance with respect to scaling. The designs presented in this work have evolved by careful use of the high-performing resources on modern FPGAs. Special care has been taken to address issues related to scaling for large FPGAs, setting this work apart from related art.

The following sections are organised as follows — Section II discusses related work with a quick background on the subject, Section III discusses the underlying algorithm, Section IV elaborately discusses both the designs, Section V presents an evaluation of the design, Section VI presents an analysis on design parameters, Section VII critically compares our design with the best among the related work, and finally Section VIII concludes the paper.

II. Background

The rank-1 update algorithm used in this paper is an elementary idea, variations of which have also been applied to cache-aware computing on general purpose processors [3], though not as aptly. This was chosen to be implemented on an FPGA since it is particularly suitable for the task as verified by both Dou and Prasanna.

Much of the related work are designs targeted and optimised for Virtex II Pro, which is an entry level device for HPC that made floating point computation feasible for the first time on FPGAs.

A. Related Work

The two most recent significant designs are those by Dou [4] and Prasanna [2]. They propose linear array based processing elements which are able to sustain their performance using a technique called memory switching.

Dou has proposed the design of a matrix multiplier highly optimised for the Virtex II Pro series of FPGAs. The design included an optimized custom 12-stage pipelined floating point

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.13

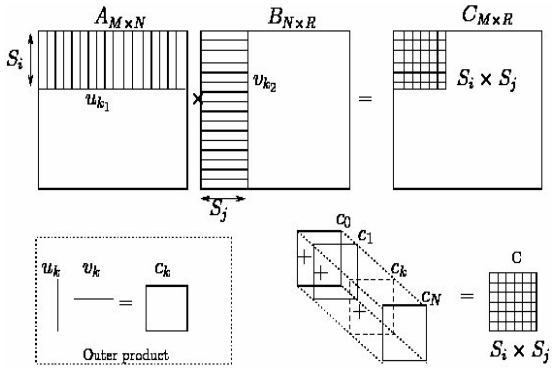


Fig. 1. The rank-1 update scheme

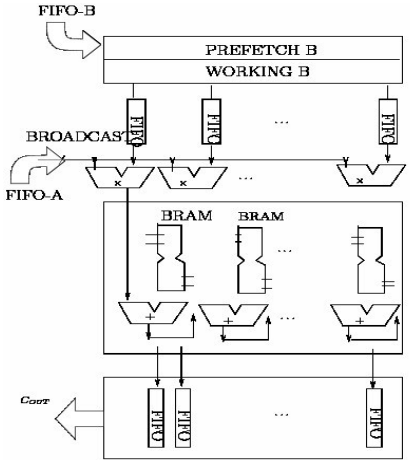


Fig. 2. Overview of Design-I

MAC, but with a few limitations like no support for zero and other denormal numbers. This design required the subblock dimensions to be powers of two. The bandwidth requirement was low at 400 MBps with 12.5 Mb of local memory utilization and they report a PE design with a synthesis frequency of 200 MHz and therefore estimated a 15.6 GFLOPS performance accommodating 39 PEs on a Virtex II pro XC2V125, a large theoretical device. As these are only synthesis results, the real frequency after placing and routing 39 PEs could be much less. Also, correcting the limitations of MAC would affect resource usage and speed.

Zhou and Prasanna have reported an improved version [2] of their design reported in [5]. The recent one reports 2.1 GFLOPS for 8 PEs running at 130 MHz on a cray XD1 with XC2VP50 FPGAs. About 35% speed degradation was observed when scaled from 2 to 20 PEs. In the earlier paper they presented a design with a peak performance of 8.3 GFLOPS for the Virtex II Pro XC2V125, where the clock degradation was 15% when the number of PEs increases from 2 to 24.

III. Algorithm

The rank-1 update scheme for matrix multiplication, illustrated in the Figure 1, has been described here for the convenience of the reader. The paper partly follows the notation introduced by Dou [4] as both are variations of the same algorithm. Consider A, B and C of dimensions M×N, N×R and M×R respectively. The objective is to compute $C = AB$. When a S_i×N panel of A (say, PA) and a N×S_j panel of B (say PB) are multiplied, the result is a subblock of the matrix C with dimensions $S_i \times S_j$. The outer product of the k^{th} column of the vector (u_k) from PA and the k^{th} row vector(v_k) from PB is an intermediate result and accumulation of such results with k ranging over the panel length (from 1 to N) is the required subblock of C.

For an outer product, each element of vector u_k multiplies each element of vector v_k. Thus, $n(v_k)$ or S_j elements are re-used $n(u_k)$ or S_i times with S_j multiplications each time. What also follows is that if one element of u_k and all the elements of v_k are available to the system then each of the products can be carried out independently. This design exploits these key ideas. With this, the focus for the design now shifts to effective utilization of resources like BRAMs, DSP48 blocks etc.

IV. Implementation

The designs have been optimised to effectively use the high performance primitives available on a modern high-end FPGA. This section describes two designs, I and II. The goals for the first design were full utilisation of PEs and overlapping I/O and computation and thereby sustaining peak performance. These were achieved at the cost of a high I/O bandwidth and sub-optimal use of available resources. The second design addresses all the limitations of the first design, and the section on design II describes its evolution in terms of more effective use of the previous data-path and resources.

Broadly, broadcasting elements of PA to all PEs and the streaming in of elements of PB to the prefetch registers is central to both the schemes and the relative rate at which they are streamed in, and the manner of their re-use is what essentially differentiates the two.

A. Design-1

This design assumes $P = S_j$ and $S_i = S_j$, where P is the number of PEs. The $S_i > S_j$ case is also acceptable. Figure 2 gives an overview of this design. The following enumerated list will describe all the major labeled components, shown in Figure 3, of the PE. It will be clear shortly that this design requires 2 words per design clock.

Component Description:

1) **B Prefetch Unit**: This unit is used to prefetch S_j elements of the next row of PB while the current row is used. The input to the first of such units is a stream of elements from the matrix B, in a row major fashion. Each unit has one data input and two data outputs: a serial-shift-forward, which happens every clock cycle and a parallel-load-down which happens every P shifts

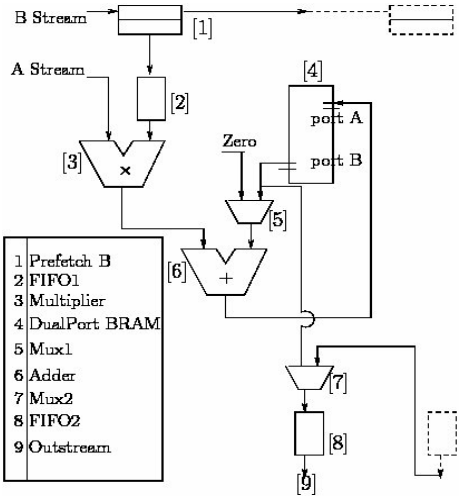


Fig. 3. 1 PE: Design-I

(or S_i clocks if $S_i > S_j$). These P words are available at the output for at most S_i clocks.

2) **FIFO1** : When the B Prefetch units were connected directly to the multipliers, a severe and unexpected drop in the design frequency was observed. This drop was inferred to be due to the routing overhead in bringing the data lines from a 64 bit register to the 13 DSP48 blocks which make up a double precision multiplier. A FIFO built out of BRAM was placed in the path in order to reduce the length of the routing path thus ensuring the expected design frequency. Due to the physical proximity of the BRAMs to the DSP blocks on Xilinx FPGAs, complicated routing is avoided and also, now, the 64 bits have to route to one BRAM instead of 13 DSP48s.
3) **Multiplier** : A standard double precision floating point multiplier IP(version v3.0) from Xilinx is used for this block. A latency of 19 cycles gives it a maximum clock speed of about 431 MHz using 13 DSP48 units. One input to the multiplier comes from the prefetch unit via the FIFO1 and the other input is the element from matrix A which is broadcast to all multipliers. The output of this multiplier is one of the inputs to the adder. The recent floating-point v4.0 is superior in terms of area resource(DSP48) usage and latency, especially for Virtex-5 series, but reduces the speed, hence is not used in the design.
4) **Dual-Port BRAM** : This dual port blockram is used as the storage space for the accumulation step of the algorithm. The *adder* writes back to the RAM using port A and reads from the RAM using Port B. The output of port B is duplicated as the input to Mux2 as well.
5) **Mux1** : This mux ensures '0' is added to the incoming product stream, whenever a new panel is read in, while the previous result in the BRAM is copied into FIFO2's.
6) **Adder** : A standard double precision floating point adder IP(version v3.0) from Xilinx is used for this block. It receives two inputs, one from the multiplier and one from the Mux1. It writes back to blockram at the appropriate location considering its own latency.
7) **Mux2** : The mux is used to switch connections between the BlockRAM (Result-backup mode), and the other instances of FIFO2 (Serial-dataout mode).
8) **FIFO2**: In order to ensure that there is no stalling in the pipeline the result needs to be backed up. Since both the ports of the result BRAM are busy, a separate memory unit is used for the back-up, in the form of FIFO2. The final updated data of the result sub-matrix 'C' (one column of 'C' when we talk about 1 PE) will be loaded into the corresponding FIFOs (Result-backup mode) of the PEs. When the result has been copied into the FIFO, input of the FIFO gets connected to the output of FIFO2 of the previous PE, thus allowing us to take the output in a streaming fashion (Serial-dataout mode).

Data Flow - design 1: The inputs, elements from PA and PB are streamed in column major and row major order respectively. First, one of the rows from PB (v_k) shifts into the B prefetch unit. Once a complete row is shifted-in ($S_j = P$), the 'prefetch-line' registers are full and this data is loaded down to the 'working-line' registers. In the meanwhile the prefetch-line continues to shift-in the next row from PB (v_{k+1}). At this point, when working-line is available and connected to one of the inputs of the multiplier, elements from the corresponding column (u_k) from A are broadcast to the second input of all the multipliers. After a latency period of the multiplier (say, L_m) the first result of multiplication is available at the output along with the appropriate handshaking signals which are used to trigger accumulation of the outerproducts at the storage area.

Once the pipeline of the multiplier has been filled it shall not be stalled since no data dependencies exist between subsequent multiplications. This allows continuous feeding of the data at the maximum design frequency. The result of the addition, available after a latency period (say, L_a), is stored in the BRAM. We will see later that the pipeline may stall in one case. Zero is accumulated with the product stream during the first outer product of each new PA$\times$PB, after which the accumulation happens with the appropriate location in the BRAMs. The FIFOs responsible for the output are loaded with the values parallely from the BRAMs once the final value of the result subblock $C_{S_i \times S_j}$ is ready. After the loading/backup is completed, these FIFOs switch modes allowing us to stream the data out in a serial fashion.

Merits and Summary: The design described above requires that $S_i \geq S_j = P$, implicitly assuming a bandwidth of 2-words per design clock cycle. PCIe is capable of such high bandwidths, and is the norm for today's large FPGAs. The merits and demerits of this design have been summarised in the following list, details about the performance and analysis are presented in a later section.

1) Overlaps I/O and computation completely. Therefore, except for the initial latency period, all the processing

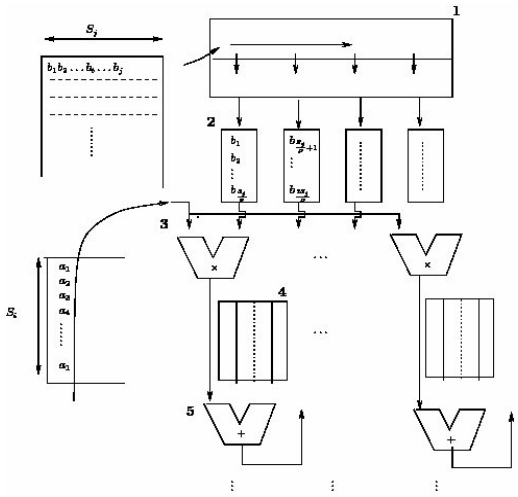


Fig. 4. 1 PE: Design-II

elements (both the floating point operators) are in use all the time, thus sustaining peak performance.

2) The design scales seamlessly ($< 1\%$ reduction in speed) as seen from Table I.
3) Uses off-the-shelf Coregen floating point adders and multipliers, allowing for portability across technologies and generations, custom-precision option, better IEEE compliance etc.

Drawbacks:

1) The design requires a sustained bandwidth of 2-words per cycle corresponding to the design frequency. For a design speed of, say, 350 MHz this translates to a 5.6 GBps bandwidth requirement, which can well be provided by today's standards, but is high nonetheless.
2) Little to almost no flexibility in the choice of S_i and S_j which might affect the overall runtime even with the sustained peak performance.
3) S_i cannot be less than the latency of the floating point adder as otherwise there would arise a data dependency. Practically this is not usually a problem due to the large sizes of matrices under question.

B. Design-II

Design-I assumed $S_i \geq S_j = P$ and so $S_j = P$ elements of B were being reused S_i times. The design-II ensures more re-use by allowing S_i and S_j to be greater than P, (i.e., $S_i, S_j \geq P$), however S_j needs to be a multiple of P.

As shown in the Figure 4 the data-path is similar to that of design-I. The design actually evolved from design-I in an attempt to find a better use of the existing components, especially the dummy FIFO1 used earlier. The following enumerated list describes the major modifications as

1) **B Prefetch Unit**: This is the same as described earlier in design-I, however two sets of registers are not necessary. 'BRAM Cache' is made part of the working-line by using it for storage.
2) **B Cache**: The design-I employed a dummy FIFO1 (BRAM based) to prevent the design frequency from falling drastically. This component now assumes a central role in design-II. Each BRAM now stores S_j/P consecutive elements of a row from the chosen panel of the matrix B, hence renamed B Cache. Configured in dual port mode, this BRAM can easily implement the working-line required for the design.
3) **Simple Dual-Port RAM**: To accommodate the larger sizes of S_i and S_j, the storage area has been increased in size and logically segmented into $S_j/n(PEs)$ zones each storing results corresponding to one S_j. Over all, the storage area accounts for the storage of $S_i \times S_j$ elements of the result block. Writing to the appropriate segments is handled by address generation and control.

Design Merits:

1) Inherits all the merits from design I – as enumerated earlier.
2) Addresses all the identified drawbacks of design I, *viz* drastically reduces the bandwidth requirement, more flexibility in the choice of S_i and S_j and relaxes the constraint on S_i w.r.t the latency – the details of which are described in the following sections.

Data Flow: Data stream from a row of PB is fed to the prefetch unit as before, but the sequence of data is such that the i^{th} consecutive $\frac{S_j}{P}$ elements from a row with S_j elements, are loaded into the B-Cache storage corresponding to the i^{th} PE. Thus the following sequence is observed, assuming $b_1,\ b_2,\ b_3 \ldots$ are the contents of PB in row major fashion:

$$b_1, b_{\frac{S_j}{P}+1}, b_{\frac{2S_j}{P}+1}, \ldots b_2, b_{\frac{S_j}{P}+2}, \ldots$$

One element from A is used for $\frac{S_j}{P}$ cycles, where it multiplies all S_j elements of a row. During the first outerproduct computation, the multiplier result is accumulated with 0 and stored in the BRAM. Thus, one outerproduct computation takes $\frac{S_i \times S_j}{P}$ cycles to complete, after which the elements from the next column of A are required. As a result of this, the restriction of $S_i \geq L_a$ is relaxed to $\frac{S_i \times S_j}{P} \geq L_a$. But the most important consequence of the new design is that the bandwidth requirement is considerably reduced as a result of a trade-off with local memory usage/data re-use.

Illustrative Example: Consider the product of two square matrices A and B each with dimensions 800×800. With a design speed of 350 MHz, we consider the following two cases.

Case I : $[S_i = S_j = P;\ P = 50]$

In this case, the bandwidth required is 2 words per cycle which with 350 MHz means 5.6 GB/s ($= 2 \times 8 \times 350$). One outerproduct computation in this case takes S_i cycles and therefore one $S_i \times S_j$ subblock computation of the result takes $S_i \times 800$ cycles. For the entire matrix multiplication of $A \times B$, there are 16×16 such subblocks. Therefore, the total number of cycles for $A \times B$ computation is S_j +$S_i \times 800 \times 16 \times 16 = 10240050$.

Case II : [$S_i = S_j = 400$; $P = 50$]

As $\frac{S_j}{P} = 8$, we see that one word of A is required every 8 clock cycles. So, a bandwidth which gives us 2 words for every 8 cycles, or 0.25 words per cycle or 700 MB/s(= $.25 \times 8 \times 350$) will be sufficient. As for the total computation time, one can see that an $S_i \times S_j$ result subblock computation requires $\frac{S_i \times S_j}{P}$ and there are 4 such blocks here. Therefore, the total number of cycles for $A \times B$ computation is $S_j + \frac{S_i \times S_j}{P} \times 800 \times 4 = 10240400$.

Thus, design II solves the problem using significantly lower bandwidth than the first design. **The increase in the cycles required for computations is because of the increased setup time**.

V. Design Evaluation

Xilinx ISE 10.1sp1 and ModelSim 6.2e was used for implementation and simulation of the design respectively.

The most significant aspect of the design, from the Table I, appears to be the negligible variation of the speed despite scaling up to 40 PEs, an explanation for which is offered in the comparison section. **The drastic reduction in speed, from 373 to 201 MHz, on SX95T is attributed to the expected poor routing when the resource utilization reached $> 95\%$, therefore considered a corner case.**

As shown in Table II, due to abundance of resources their liberal use is justified. Appropriate pipeling, not shown in the figures, has been done in order to break the critical paths. It can also be seen from the resource usage at 40 PEs that a few more PEs can be accommodated in the SX240T.

The design was ported to the Virtex 2 Pro XC2VP100 for the sake of comparison and as shown in Table III, about 20 PEs can be fit with a frequency of about 134 MHz as opposed to 31 PEs and 200 MHz(synthesis) respectively by [4] (In a later usage of the same PE by one of the co-authors of [4], the actual implementation frequency was about 100 MHz [6])

TABLE I
Timing Information (post PAR)

No. PEs	SX240T(-2)[MHz]	SX95T (-3)[MHz]
1 PE	374	377
4 PEs	373	374
8 PEs	344	373
16 PE	-	373
19 PEs	-	373
20 PEs	372.8	201
40 PEs	371.7	-

VI. Performance Analysis

We present an analysis of the design parameters listed in Table IV studying their effect on performance and the constraints they impose. All the analysis is with respect to design-II.

Each element of A is used $\frac{S_j}{P}$ times, in an outerproduct and therefore the entire computation of the outerproduct takes $\frac{S_i \times S_j}{P}$ cycles. In order to overlap I/O and computation, the algorithm requires that we prefetch S_j elements of B for the next outerproduct. We have therefore

$$S_i + S_j \leq \frac{Si \times S_j}{P} \times \beta \tag{1}$$

We also see that the $S_i \times S_j$ needs to be maximized here. The constraint on memory gives us Eq 2 which on approximation gives Eq 3

$$2S_i \times S_j + 2S_j = 2(S_i + 1) \times S_j \leq m \tag{2}$$

$$2(S_i) \times S_j \leq m \tag{3}$$

To maximize $f(S_i, S_j) = S_i \times S_j$, under the constraints Eq 1 and Eq 3 we use the Lagrangian constrained optimization method

$L(S_i, S_j, \lambda, \mu) =$
$S_i \times S_j + \lambda \left(\beta \frac{S_i \times S_j}{P} - (S_i + S_j)\right) + \mu\,(m/2 - S_i \times S_j)$

$$\frac{\partial L}{\partial S_i} = S_j - \lambda + \lambda\beta\frac{S_j}{p} - \mu S_j = 0 \tag{4}$$

$$\frac{\partial L}{\partial S_j} = S_i - \lambda + \lambda\beta\frac{S_i}{p} - \mu S_i = 0 \tag{5}$$

Equations 4 and 5 suggest $S_i = S_j$. If we substitute $S_i = S_j = S$, we get

$$\text{Maximize} \quad S \tag{6}$$

$$S \geq \frac{2P}{\beta} \tag{7}$$

$$S \leq \sqrt{\frac{m}{2}} \tag{8}$$

TABLE II
Resource utilization for SX95T and SX240T devices (post PAR)

No PEs	DSP48E	FIFO	BRAM	Slice Reg	Slice LUT
1 PE	16	1	2	2511	1374
4 PE	64	4	8	10377	5451
8 PE	128	8	16	20865	10886
16 PE	256	16	32	41841	21750
20 PE	320	20	40	52329	27176
40 PE(sx240)	640	40	80	69%	36%
					RESOURCES PER DEVICE
Device					
SX240T	1056	516	516	149760	149760
SX95T	640	244	244	58880	58880

TABLE III
Resource utilization for Virtex II Pro XC2VP100 (post PAR)

	$\text{Tot}_{\textbf{xc2vp100}}$	$U_{15\ PE}$	$U_{20\ PE}$
MULT18x18s	444	240	304
RAMB16s	444	90	114
Slices	44096	30218 (68%)	37023 (83%)
Speed		133.94 MHz	133.79 MHz

TABLE IV
List of parameters

parameters	meaning
β	bandwidth in terms of the no. of words per design clock
x_a, x_b	such that $x_a + x_b \leq \beta$
m	total amount of local memory
n	num. of columns of a (or rows of b)

The following analysis for the minimum required bandwidth demonstrates the burst-friendly nature of the design. We know that S_i words of A are required for S_j words of B within $\frac{S_j \times Si}{p}$ cycles. Thus we get the values for min(x_a) and max(x_b) for a constant bandwidth of β. Thus we get the values for min(x_a) and max(x_b) for a constant bandwidth of β.

$$\min(x_a) = \frac{P}{S_j} \quad (9)$$

$$\max(x_b) = \beta - \frac{P}{S_j} \quad (10)$$

For the case where $S_i = S_j$ equal distribution of bandwidth is the best approach, for other cases a similar analysis results in the appropriate distribution. The availability of more than the minimum amount of bandwidth means that the excess bandwidth can be used to transfer as much A as required in one go – further trading bandwidth with local storage. This caching also creates time during which the bandwidth can be used for other I/O, allowing for sharing the same bandwidth across multiple FPGA boards.

VII. Comparison

The following compares a few aspects of ours designs with the recent related work. In particular we compare with Dou [4] and Prasanna [2], [5], the former of which was identified superior to other related work by Craven-2007 [7].

- **Scaling**: As reported previously [2], [5] frequency falls by about 35% and 15% respectively by scaling to 20 PEs. Our designs show negligible($<$1%) degradation in frequency up to 40 PEs. Further, the low-bandwidth requirements and burst-friendly behaviour allows design-II to scale well across multiple FPGAs due to low bandwidth requirement per FPGA.
- **Flexibility**: Dou's design requires matrix subblock dimensions to be powers of 2. Prasanna supports square matrices of limited size in [5] and arbitrary size in [2]. Our designs support arbitrary matrix sizes as in [4] without placing extra constraints on $S_i,\ S_j$.
- **PE/MAC**: Dou's custom MAC [4] is highly optimized for Virtex-2 Pro and may not scale across families of FPGAs. The MAC doesn't support zero and denormal numbers. Our design uses floating-point units from core-generator making the design more flexible(portable, scalable, customizable) along with better IEEE compliance. It is to be noted that these are optimized for Virtex-5.
  We were able to place and route only 20 PEs on Virtex-2 Pro XC2VP100 as opposed to 31 PEs (synthesis-only) by [4] which was possible because of the custom designed MAC which use only 9 18x18 multipliers as opposed to 16 by core generator. But such custom MAC may not be appropriate in the context of, say, Virtex-5 SX240T where there are about 1200 DSP48s and the coregen floating point units are highly optimized to use them effectively.
- **I/O–Computation Overlap**: The designs use a variant of 'pipelining' or buffering for the purpose of overlapping I/O and computations as opposed to memory switching used in related works. This may be a factor in the better scaling of our designs as explained below. Memory switching requires two memory-banks to alternately feed the processing elements. This places constraints on the placement of the memory banks with respect to the processing units. In this implementation, one memory unit feeds another, except for those connected directly to the processing units. This takes advantage of their physical proximity on the device and the better routing between BRAMs and DSP48 blocks.

VIII. Conclusion

In this paper two designs for matrix multiplication are presented which vividly demonstrate the trade-off between memory and bandwidth. The simplicity of the designs and the use off-the-shelf floating point units from Xilinx Coregen offer easy reproduction of the design, portability across FPGA families and maintainability along with better IEEE compliance and options such as custom precision. The designs are able to sustain the peak performance, like a few other related work, achieved by use of a technique alternative to memory switching, which also has a favourable impact on routing. Our designs scale well with $<$1% degradation in speed and design-II further enables scaling across multiple FPGAs. For about 40 PEs, with a design frequency of 373 MHz on Virtex-5 SX240T FPGA, a sustained performance of 29.8 GFLOPS is possible with a bandwidth requirement of 750 MB/s for design-II and 5.9 GB/s for design-I. The design can be made available upon request. Future work includes porting it for use with the CRL-India's supercomputer.

Acknowledgment

The authors acknowledge Sunil Puranik and others from CRL-India for their insights on HPC; Rahul Badghare and Pragya Sharma for their help in the timing analysis.

References

[1] K. D. Underwood and K. S. Hemmert, "Closing the gap: Cpu and fpga trends in sustainable floating-point blas performance," in *FCCM*. IEEE Computer Society, 2004, pp. 219–228.

[2] L. Zhuo and V. K. Prasanna, "Scalable and modular algorithms for floating-point matrix multiplication on reconfigurable computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 433–448, 2007.

[3] K. Goto and R. van de Geijn, "High performance implementation of the level-3 BLAS," accepted: 28 October 2007.

[4] Y. Dou, S. Vassiliadis, G. K. Kuzmanov, and G. N. Gaydadjiev, "64-bit floating-point fpga matrix multiplication," in *FPGA '05: Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. New York, NY, USA: ACM, 2005, pp. 86–95.

[5] L. Zhuo and V. K. Prasanna, "Scalable and modular algorithms for floating-point matrix multiplication on fpgas," *IPDPS*, vol. 01, p. 92a, 2004.

[6] G. Kuzmanov and W. van Oijen, "Floating-point matrix multiplication in a polymorphic processor," in *International Conference on Field Programmable Technology (ICFPT)*, December 2007, pp. 249–252.

[7] S. Craven and P. Athanas, "Examining the viability of fpga supercomputing," *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, pp. 13–13, 2007.

FPGA Implementation of Support Vector Machine based Isolated Digit Recognition System

J.Manikandan, B.Venkataramani and V.Avanthi
Department of ECE, National Institute of Technology, Trichy (NITT), Trichy-620015, TN, INDIA
{jmanikandan.nitt@gmail.com, bvenki@nitt.edu}

Abstract

In this paper, two schemes for FPGA implementation of multi-class SVM based isolated digit recognition system are proposed, one using only logic elements and another using both soft-core processor and logic elements(LEs). One of the major contributions of this paper is the proposal for implementation of the decision function using only fixed point arithmetic without compromising the recognition accuracy. Compared to the scheme which uses floating point arithmetic, the proposed scheme reduces the number of LEs required by a factor of 3.29. The second scheme proposed results in about 25 times lower area compared to the first scheme. For the soft-core processor approach, a custom instruction is proposed for floating point arithmetic. Speaker dependent TI46 database of isolated digits is used for training and testing. Features are extracted using both Linear Predictive Coefficients (LPC) and Mel Frequency Cepstral Coefficients(MFCC) and features are compressed using Self Organized Feature Mapping (SOFM). This in turn is used by the SVM classifier to evaluate the recognition accuracy and the hardware resources utilized. Both the schemes proposed result in 100% recognition accuracy when implemented on Altera Cyclone II FPGA. The proposed schemes can also be used for speaker verification and speaker authentication applications. Since the scheme which uses soft-core processor requires lower area, it can be used for systems which require a large vocabulary size.

1. INTRODUCTION

Support Vector Machine (SVM) is one of the popular techniques for pattern recognition and is considered to be the state-of-the-art tool for linear and nonlinear classification [1]. SVM is basically a binary classifier and it has been employed for several applications such as beam forming [2], ultra wide band (UWB) channel equalization [3], channel estimation in Orthogonal Frequency Division Multiplexing (OFDM) systems [4], voice activity detection [5], target recognition [6] and many more. The SVM classifier was originally developed for two-class or binary classification and the demanding applications of pattern recognition led to the design of multi-class SVM classifiers using the binary SVM classifiers.

FPGAs have come a long way from mere glue-logic applications of interconnecting discrete components to high-performance reconfigurable signal processors. FPGA-based processing outperforms conventional processors, resulting in improvements in processing speed, size, weight, power consumption, supply voltage, and cost. Compared to basic microcontrollers that have built-in analog-to-digital converters, CAN, and other features, the FPGA seems to be resource-efficient. Moreover, embedding soft processors such as Nios II (for Altera FPGAs), Picoblaze/Microblaze (for Xilinx FPGAs) offers a radical yet stable way to effectively eradicate the problem of processor obsolescence. The reconfigurability feature of the FPGA is a great advantage. Coding, compilation, simulation and testing are more straightforward with an FPGA than with a microcontroller and hence it has been employed for design of several applications [7]. FPGAs have also been proposed for the implementation of support vector machines. A two-class SVM classifier for multispectral brain images is proposed in [8]. SVM classifier using logarithmic number system (LNS) is proposed in [9]. Both these implementations use Xilinx FPGA.

In this paper, a multi-class SVM classifier which dispenses with signum function is proposed for isolated digit recognition system. It is implemented on Altera Cyclone II FPGA using two schemes, one which uses only LEs and another which uses LEs and the soft-core processor.

The organization of the paper is as follows: Section II gives a brief introduction to Support Vector Machine (SVM) and the architecture of SVM based isolated digit recognition system. Section III presents the results on the performance of the proposed system on MATLAB. Section IV presents the implementation results on FPGA, followed by conclusion and references.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.23

2. OVERVIEW OF SVM CLASSIFICATION AND ISOLATED DIGIT RECOGNITION SYSTEM

A. SVM Classification

The aim of SVM Classifier is to devise a computationally efficient way of learning 'good' separating hyperplanes between different classes in a high dimensional feature space. SVM is used to identify a set of linearly separable hyperplanes which are linear functions of the high dimensional feature space as shown in Figure 1. The hyperplanes are placed such that the distance between the classes is maximum. An introduction to SVM algorithm for linearly non-separable classes is given here. More details about SVM classifier can be obtained from [10],[11]. Let {xi,yi} for i=1, 2, … N denote the training data set where y_i is the target output for training data x_i, The aim of SVM is to maximize the objective function L(α)

$$L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_i \Phi(x_i)^T \Phi(x_j) \qquad (1)$$

subject to constraints

(i) $\sum_{i=1}^{N} \alpha_i y_i = 0$

(ii) $0 \le \alpha_i \le C$ for all i, where.

where C is the cost parameter, α_i is the hyperparameter and Φ(.) is the feature mapping.

To solve the optimization problem mentioned in (1) with above-mentioned constraints, a well known procedure called Quadratic Programming (QP) is used to minimize [-L(α)] [12] and the following classification function is obtained

$$Y = \text{sgn}(w.\Phi(z) + b)$$

$$Y = \text{sgn}(\sum_{i=1}^{q} \alpha_i y_i K(x_i, z) + b) \qquad (2)$$

where w = $\sum_{i=1}^{q} \alpha_i y_i \Phi(x_i)$ gives the weight for the SVM classifier, α_i is the Lagrange multiplier assigned to each training data whose value depends on the role of the corresponding training data in the design of the classifier system. Nonzero values of α_i's correspond to support vectors that are used to construct the classifier in (2), 'q' denotes the number of support vectors, $K(x_i,z) = <\Phi(x_i).\Phi(z)>$ is the kernel function, 'b' is the bias and 'Y' is the classifier output for the test data 'z'.

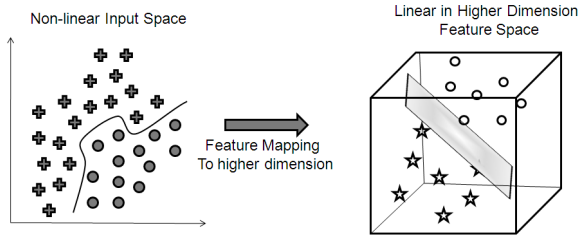


Figure 1. Non-linear to linear transformation using SVM Technique

The kernel functions map the input into higher dimensional feature space and the kernel functions must satisfy the Karush-Kuhn-Tucker (KKT) Conditions and Mercers Condition [10] to retain the geometrical interpretation of the feature space, so that the solution of the objective function gives a hyperplane in the feature space. The linear kernel for SVM classifier is given as $K(x_i,z) = <\Phi(x_i).\Phi(z)> = < x_i.z >$.

The weights 'w' and bias 'b' are stored in memory for each digit/class. For any test input data 'z', $K(x_i,z)$ is computed and all the values are substituted in (2).

A binary or 2-class SVM classifier will have the recognition result as z Є class 1, if Y = 1 and z Є class 2, if Y = -1. Figure 2 shows the block diagram for two-class SVM classification. Assume D_c to be the decision function given for class 'c' as

$$Dc = \sum_{i=1}^{q} \alpha_i y_i K(x_i, z) + b \qquad (3)$$

For a multi-class problem, the recognition result for SVM classifier is given as Y = $\arg\max_c(D_c)$ for c Є {1,2,3,…N} and in general z Є class 'n' for Y=n.

SVM models have a cost or penalty parameter C, which controls the trade-off between the complexity of the decision function and the number of wrongly classified testing points. For applications requiring higher feature dimensions such as speech, it is difficult to visualize the decision boundary, but the recognition accuracy can be evaluated for various C values. While comparing the classification methods using SVM, the correct value of C for each of the compared methods is crucial, as only then a reliable estimate of the performance is obtained for each of the methods. The

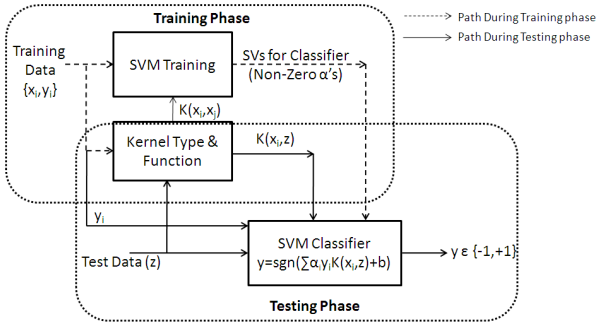


Figure 2. Block diagram of a two-class SVM Pattern Classifier

correct C value cannot be known in advance and a wrong choice of the SVM penalty parameter C can lead to a severe loss in performance. Therefore, the parameter values are usually estimated from the training data by cross-validation and exponentially growing sequences of C.

B. Isolated Digit Recognition

Isolated digit recognition is a problem of pattern classification which has a very vast feature size. Figure 3 shows the block diagram of proposed isolated digit recognition system using SVM classifier, comprising of feature extraction, self organized mapping of the features extracted and SVM classifier architecture stage. The input speech is divided into frames and LPC/MFCC coefficients are computed for each frame. (The Linear Predictive Coefficient (LPC) features are extracted from the input speech using the pre-emphasis, frame blocking, windowing, and autocorrelation blocks. Details about the steps involved in extracting the LPC features can be had from [13],[14].) Similarly, Mel Frequency Cepstral Coefficients (MFCC) is computed using the following blocks: pre-emphasis, frame blocking, windowing, FFT, Magnitude Square, Mel filter bank processing, DCT, computation of energy, computation of delta and double delta features. Detailed account of the procedure adopted for computation of MFCC is given in [15].) Figure 4(a) and 4(b) shows the block diagram for LPC and MFCC feature extraction of the input speech respectively. Each frame results in either 11 LPC coefficients or 25 MFCC coefficients (13 MFCC + 12 delta MFCC). The total number of LPC/MFCC coefficients for each digit is not uniform because each input digit comprises of different number of frames. SVM based recognition requires equal number of feature vectors for each input. In order to make the size of input features uniform for all the digits, Kohonen's self-organizing feature map (SOFM)[16] is used. SOFM reduces the input feature size for classification too. The SOFM transforms the acoustic vector sequences of features into trajectories in a square matrix of fixed dimension, where each node of the matrix take on binary values[17]. The feature map is trained by Kohonen's self-organization learning. For the purpose of SOFM training, ten utterances of all the ten digits are used. The self-organization learning has the property that after training, physically similar input vectors in Euclidean space, correspond to topologically close nodes in the feature map. The size of SOFM can be 12x12, 18x18, 24x24 or any other combination. Mapped features of SOFM for each digit are obtained using the SOFM weights. In the SOFM array shown in Figure 3, black dots, white dots denote the values '1' and '0' respectively.

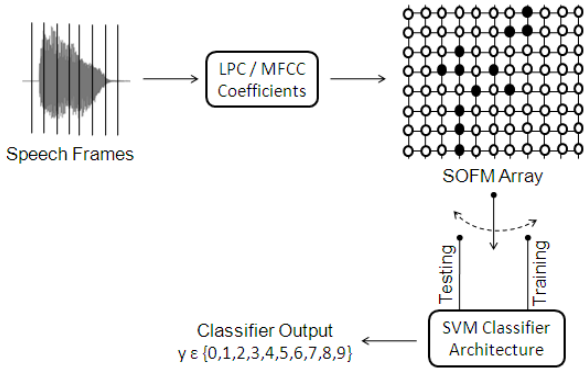


Figure 3. Block Diagram of proposed SVM-based Isolated Digit Recognition

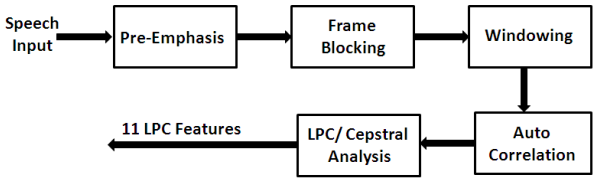


Figure 4(a). Block diagram for LPC Feature Extraction

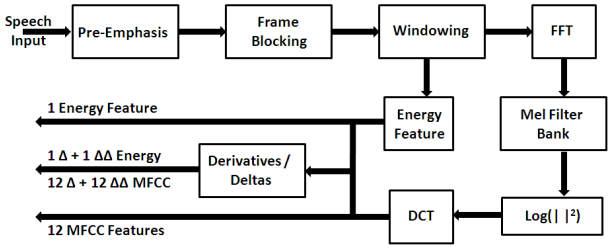


Figure 4(b). Block diagram for MFCC Feature Extraction

3. PERFORMANCE RESULTS OF SVM BASED ISOLATED DIGIT RECOGNITION SYSTEM

In this section, the results of software implementation of the SVM based isolated digit recognition system is given. The speaker dependent TI46 database corpus [18] is used for the evaluation of proposed SVM based isolation digit recognition system. Ten utterances of each digit from 1 female speaker are used for training the system and ten utterances of test data for each digit from the same speaker at different sessions are used for testing. Feature extraction using LPC/MFCC and feature mapping using SOFM has been implemented using C-code. The SVM classification algorithm described above is implemented using MATLAB [19] and the

performance of proposed SVM based isolated digit recognition system for LPC and MFCC feature inputs with various SOFM sizes and kernels is evaluated. The recognition accuracy (RA) achieved with number of support vectors (SVs) required for different SOFM sizes is given in Table I. It may be observed that MFCC feature inputs with SOFM of size 12x12 gives 100% recognition accuracy with least number of support vectors. The performance results of proposed SVM based isolated digit recognition with MFCC feature input and SOFM size 12x12 for different kernels is given in Table II. It may be observed from Table II that the best recognition accuracy is achieved by using sigmoid and linear kernel for LPC and MFCC feature inputs respectively. From Table I and Table II, it is concluded that the best combination for FPGA implementation of the SVM classifier is to use MFCC feature inputs mapped to SOFM of size 12x12 with linear kernel and hence this combination is employed for hardware implementation of SVM using FPGA.

Table 1. Recognition Performance for various SOFM sizes

Feature Extraction→	LPC		MFCC	
SOFM Size ↓	RA	SVs	RA	SVs
12x12	85%	627	**100%**	**602**
18x18	**97%**	**692**	100%	691
24x24	94%	739	99%	747

RA : Recognition Accuracy ; SVs : Number of Support Vectors

Table 2. Recognition Performance for Various Kernels with SOFM Size 12x12

Feature Extraction→	LPC		MFCC	
SVM Kernel ↓	RA	SVs	RA	SVs
Linear	85%	627	**100%**	**602**
Poly(deg=2)	87%	759	100%	747
RBF	90%	677	100%	611
Sigmoid	**90%**	**653**	100%	669
Wavelet(a=10)	90%	703	100%	682

4. HARDWARE IMPLEMENTATION OF SVM BASED ISOLATED DIGIT RECOGNITION SYSTEM

In this section, the results on the implementation of SVM based isolated digit recognition on Cyclone II EP2C35F672C6 FPGA are given. MFCC feature input is mapped to SOFM of size 12x12 and linear kernel is used. Two schemes are proposed for implementation, one which uses only the LEs and another which uses both LEs and Nios II soft-core processor. Altera Quartus II software is used for the design and implementation. Nios II soft-core processor is a general-purpose RISC processor core and the term "Nios II processor system" refers to a Nios II processor core, a set of on-chip peripherals, onchip memory, and interfaces to off-chip memory, all implemented on a single Altera device.

Figure 5 shows the SVM classifier architecture employed for the proposed isolated digit recognition system and the same has been implemented in FPGA. The first stage is a binary or two-class SVM classifier for each digit. For MFCC feature inputs with SOFM size 12x12, the test input '**z**' to SVM classifier is of size [1x144]. The size of weights 'w' for each SVM digit classifier is [1x144] totaling to [10x1x144] weights for all the ten digits. SVM learning is a one-time operation to obtain the weights and bias for each class. The weights and bias for each class are obtained using MATLAB and they are stored in FPGA memory. The output from SVM classifier block for each digit is defined as decision function D_c and for linear kernel, the decision function given in (3) is written as :

$$D_c = \sum_{i=1}^{q} \alpha_i y_i x_i z + b \qquad = \mathbf{WZ}^T + b \qquad (4)$$

where **W** and **Z** are the weight and test input in matrix form. The results from classifier blocks of each digit are passed to a decision logic block, where the class with maximum decision function is adjudged as the recognized digit.

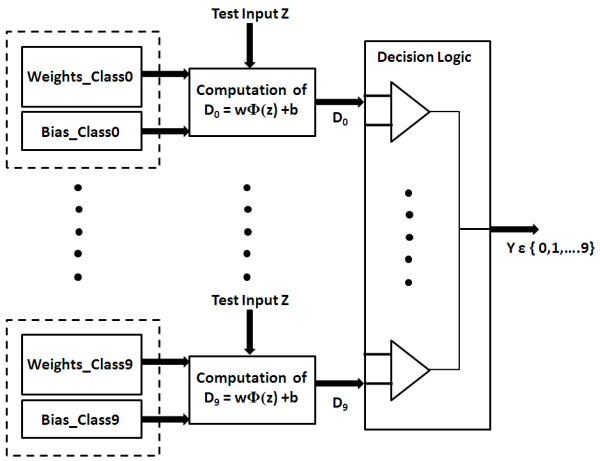


Figure 5. SVM Classifier Architecture for Isolated Digit Recognition System

A. Floating point format

In this section, implementation of SVM classifier using only logic elements (LEs) is considered. The weights '**w**' and bias '**b**' for each class are in floating point format and hence the proposed SVM classifier requires floating point arithmetic operations for each class. In order to optimize the FPGA resources, the (1,6,5) format (1 sign bit, 6 exponent bits and 5

mantissa bits)[7] for floating point number is evaluated. Value of the floating point number is given by $X = (-1)^s * 1.m * 2^{e-bias}$ (5)

$bias=2^{E-1}-1$

where 'E' denotes the number of exponent bits, 's' denotes the sign bit, 'm' denotes value of mantissa and 'e' denotes the value of exponent. For example, $9.25_{10} = 1001.01_2 = 1.00101 \times 2^3$, where e - bias = 3 => e = 3 + bias => e = 34 and 9.25 is written in (1,6,5) format as

Sign	Exponent	Unsigned mantissa
s	e	m
0	100010	00101

Floating point addition is more complex than multiplication because two numbers in floating point format f1(s1,e1,m1) and f2 (s2,e2,m2) can only be added if the exponents are same. If exponents are not same, then a normalization factor, d=e1-e2 has to be used and then the two aligned mantissas are added if the two floating point operands have the same sign, otherwise subtracted. The floating point arithmetic units are implemented on FPGA using both (1,6,5) and (1,6,11) format and the accuracy of the results obtained are given in Table III. It is observed that the (1,6,11) format gives better accuracy than (1,6,5) format. Hence (1,6,11) format is used for implementation of the SVM based isolated digit recognition.

The number of LEs required for the implementation of proposed isolated digit recognition system using (1,6,11) format is given in Table IV. Each floating point addition consumes 368 LEs, whereas Altera Megawizard Function for 32-bit floating point addition requires 1143 LEs and 838 LEs for full functionality and reduced functionality respectively [20].

Recognition results of the system using (1,6,11) format match with the results obtained from MATLAB, but it can be employed only with FPGAs having larger LE count.

Table 3. Results of Floating Point Addition using (1,6,5) And (1,6,11) Format

Example	(1,6,5)		(1,6,11)	
	Result	Error	Result	Error
9.25+0.6=9.85	9.75	1.01%	9.84765	0.023%
0.2+0.005=0.205	0.1875	8.54%	0.20489	0.053%

B. Fixed point format

It is observed that the FPGA implementation of SVM based isolated digit recognition system using (1,6,11) floating point format consumes nearly 1,16,750 LEs. In order to optimize the number of LEs, fixed point format is employed, where the weight and bias values are scaled by 10^6 and the resulting number is stored after truncating the fractional part. In this case, floating point arithmetic is replaced with fixed point arithmetic operations. The number of LEs used for the hardware implementation of the system is given in Table IV. The proposed isolated digit recognition system using fixed point format reduces the number of LEs by a factor of 3.29 over the (1,6,11) floating point format based implementation.

C. Nios II Soft Core Processor

The FPGA implementation using fixed point format reduces the number of LEs by a factor of 3.29 for the 10-class isolated digit recognition system, but the number of LEs is bound to increase if the vocabulary size for recognition is increased. In this section, hardware implementation of the proposed SVM based isolated digit recognition system using Nios II soft-core processor is explained. The system-on-programmable-chip (SOPC) approach has several advantages over a pure hardware design. SOPC approach supports programming using high level languages such as C++ and a complex system can be easily designed and tested. These soft-core processors have a differentiating factor over their hard-core counterparts that they are flexible and are superior compared to hard core processor, as the silicon area is not dedicated permanently and the size of RAM, I/O and peripherals can be configured by the user depending on the requirement, while the hard-core processor based approach is inflexible and may result in under utilization of silicon area. Custom instructions take the flexibility of soft-core processors a step ahead with algorithm-specific additions of hardware to the soft-core microprocessor's arithmetic logic unit (ALU). Figure 6. shows the use of custom instruction to extend the ALU of Nios II soft core processor, and these new hardware instructions cast the software algorithm into a hardware block. Nios II soft-core processor is implemented on Cylcone II EP2C35 FPGA and floating point arithmetic operation is performed using only software and using floating point custom instructions [21] on 1000 pairs of random operands and the acceleration factor obtained is given in Table V. From this it may be concluded that the custom instruction accelerates the floating point arithmetic by a factor of 13 to 41 and hence it is adapted for SVM implementation. The SVM classifier is implemented on Cyclone II FPGA and the number of LEs required for the implementation of proposed isolated digit recognition system is given in Table IV. It is noticed that SOPC implementation reduced the number of LEs by a factor of 25 over the floating point format based implementation and the fixed point format based implementation reduced the number of LEs by a factor of 3.29 over the (1,6,11) floating point format based implementation.

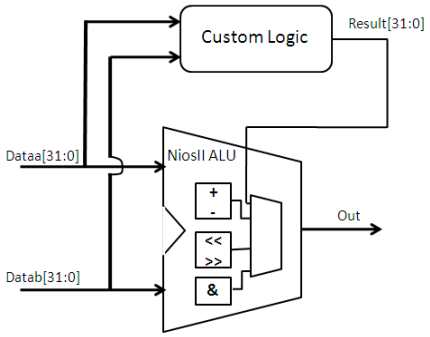


Figure 6. NiosII Processor with Custom Instruction Logic Block

Table 4. Hardware Resource Utilization for SVM Classifier

Hardware Implementation	Logic Elements (LEs)
(1,6,11) Floating point format	1,16,750
Fixed point format	35,525
Nios II Soft-Core Processor (with FP CB)	4,736
Nios II Soft-Core Processor (without FP CB)	2,734

FP CB: Floating Point Custom Block

Table 5. Acceleration Factors on using Floating Point Custom Instruction

Operation	Clock Cycles		Time (Sec)		Accel. Factor
	SW	CI	SW	CI	
FP Add	856287	61453	0.01713	0.00123	14x
FP Sub	916255	61735	0.01833	0.00123	15x
FP Mul	2470669	59158	0.04941	0.00118	41x
FP Div	1035598	79401	0.02071	0.00159	13x

SW : Software Only; CI : Custom Instruction

5. CONCLUSION

Two schemes proposed for FPGA implementation of SVM based isolated digit recognition system are implemented on Altera Cyclone II FPGA. (1,6,11) floating point format proposed in this paper achieves 100% recognition accuracy for the above system. Compared to direct implementation of SVM using the above floating point representation, two schemes, one using normalization followed by truncation and another using soft core processor reduce the area required by a factor of 3.29 and 25 respectively. The scheme using the soft-core processor has the potential for implementation of speech recognition system with large vocabulary size. The proposed SVM based isolated digit recognition system can be extended for applications such as speaker verification and speaker authentication based applications. The techniques proposed in this paper are also applicable for implementation on FPGAs from other vendors such as Xilinx.

REFERENCES

[1] Christopher J.C. Burges, "A tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery 2*, 1998, pp. 121-167.

[2] M. Martínez Ramón, Nan Xu, and C. G. Christodoulou, "Beamforming using Support Vector Machines," *IEEE Antennas And Wireless Propagation Letters*, Vol. 4, 2005, pp. 439-442.

[3] Mohamed S. Musbah and Xu Zhu, "Support Vector Machines for DS-UWB Channel Equalisation," Department of Electrical Engineering & Electronics, University of Liverpool, *UK*, *IEEE 2007*, pp. 524-527.

[4] M. Julia Fernández-Getino García, José Luis Rojo-Álvarez, "Support Vector Machines for Robust Channel Estimation in OFDM," *IEEE Signal Processing Letters*, Vol. 13, No. 7, July 2006, pp. 397-400.

[5] Fengyan Q, Changchun Bao and Yan Liu, "A Novel Two-Step SVM Classifier for voiced/unvoiced/silence classification of speech," *ISCSLP 2004*, pp. 77-80.

[6] Qun Zhao and Jose C. Principe, "Support Vector Machines for SAR Automatic Target Recognition," *IEEE Transactions On Aerospace And Electronic Systems*, Vol. 37, No. 2 April 2001,pp.643-654.

[7] U.Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, Second Edition,Springer-Verlag Berlin Heidelberg,2004.

[8] Omar Pina-Ramirez, Raquel Valdes-Cristerna and Oscar Yanez-Suarez , "An FPGA Implementation of Linear Kernel Support Vector Machines," *IEEE International Conference on Reconfigurable Computing and FPGA's, ReConFig 2006*, Sept 2006,pp.1-6.

[9] F. Khan, M. Arnold, and W. Pottenger. "Hardware-based support vector machine classification in logarithmic number systems." *IEEE Euromicro Symp. Digital System Design (DSD)*, September 2004, pp. 254–261.

[10] Nello Cristianini and John Shawe-Taylor, *An introduction to Support Vector Machines and other kernel-based learning methods*, Cambridge University Press, 2000.

[11] Corrina Cortes and V Vapnik, "Support Vector Networks," *J. of Machine Learning*, 1995.

[12] Edwin K.P.Chong and Stanislaw H. Zak, *An introduction to Optimization*, Wiley Interscience Publications, 2004.

[13] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition,* Prentice Hall Signal Processing Series, 1993.

[14] Wai C. Chu, *Speech Coding Algorithms*, Wiley Interscience, 2003.

[15] Daniel Jurafsky and James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice-Hall New Jersey, 2008.

[16] Martin T Hagan, Howard B. Demuth and Mark Beale, *Neural Network Design*, PWS Publishing Company, 1996.

[17] Zezhen Huang and Anthony Kuh, "A Combined Self-Organizing Feature Map and Multilayer Perceptron for Isolated Word Recognition," *IEEE Transactions on Signal Processing*, Vol.40, No. 11, Nov 1992.

[18] http://www.ldc.upenn.edu/Catalog/readme_files/ti46.readme.html

[19] MATLAB *User's Guide*. The Mathworks, Inc. http://www.mathworks.com/

[20] *altfp_add_sub Megafunction User Guide*, UG-041305-3.1, Altera Corporation, July 2007.

[21] *Using NiosII Floating-Point Custom Instructions*, TU-N2FLTNGPNT-1.0, Altera Corporation, May 2006.

A *"Stitch"* in Time: Accurate Timekeeping with On-Chip Compensation

Prashant Bhargava, Mohit Arora
Freescale Semiconductor
prashantb@freescale.com, mohit.arora@freescale.com

Abstract

Applications like Energy Meters that rely on real time data require accurate time under all environmental conditions. Typically, these applications rely on Real Time Clock (RTC) for all real time operations but there are many factors like crystal aging, incorrect loading and temperature variations that tend to change the frequency of the clock used for RTC resulting in inaccurate time. Hence there is an unavoidable need to have compensation technique inside the RTC to counter balance this error in clock frequency of crystal. This paper describes a digital hardware compensation technique which compensates by adding or removing pulses in a particular timing window thus maintaining accurate clock. Technique described in this paper uses simple hardware to ensure low power consumption thus maintaining longer battery life. This enables applications to use cheaper crystal that may be inaccurate and compensate for the inaccuracies within the hardware thus reducing board cost.

1. Factors affecting clock accuracy

Accuracy is a measure of frequency error, which is the magnitude of the difference between the actual frequency and the expected frequency. Accuracy is usually expressed in terms of fractional frequency offset and is computed as the frequency error divided by the expected frequency. Some of the factors that affect the accuracy of a clock are as follows.

a) Crystal characteristics
The accuracy of a crystal used on board may vary among different manufactures. Vendors often prefer to choose crystals that are cheaper and these typically have a bad tolerance over temperature variation. High precision crystals with an accuracy of better than 10 PPM at room temperature can be expensive.

b) Inaccurate loading of the crystal
Crystals are usually specified by their series or parallel resonant frequencies, or both. An oscillator will nearly always operate near its series resonance, the parallel resonance being more of a convenience of measurement. The terms "series" and "parallel" resonant operation are often used but simply refer to low or high loading impedance across the crystal's terminals respectively. The "loading" impedance can be measured by removing the crystal and "look into" the circuit at the operating frequency.

So crystal frequency may depend on loading applied to them. Even a presence of high EMI in a poorly loaded crystal might make it unstable.

c) Handling of the crystal
Thermal shock during soldering can alter a crystal's frequency.

d) Variation in temperature
Several environmental factors like pressure, humidity and vibration can influence RTC accuracy but it is mainly the inferior characteristics of a quartz crystal over temperature which results in deviations if temperature is changing. This is the most critical factor affecting clock accuracy. Change in environment temperature affects the crystal and thus the clock accuracy.

e) Aging characteristics
Crystal aging applies to the cumulative change in frequency over time, which results in a permanent change in the operating frequency of the crystal unit. Many interrelated factors are involved in aging; some of the most common being: internal contamination, excessive drive level, surface change of the crystal, various thermal effects, wire fatigue and frictional wear. Proper circuit design incorporating low operating ambients, minimum drive level and static pre-aging will greatly reduce most of the aging problems.

Typically, accuracy may deteriorate by 5 PPM per year (or approx. 2.6 minutes in a year) for a typical 32 kHz crystal.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.70

2. Need for compensation

Many appliances, such as water and electricity meters, operate over a wide range of temperature. The real-time clock (RTC) is a key component for these applications relying on data in real time. An RTC, for example, is the key to determining how much energy is consumed per house through an electricity meter application. Since clock accuracy may vary based on temperature variation, an inaccurate time may result in an inaccurate billing.

Many software applications that gather data in real time rely heavily on the RTC's accuracy for complex data analysis. Clearly, the RTC and its accuracy are vital components for applications relying on real-time values. So these kinds of applications require precise real time clocks that can adjust their time based on variation in temperature or environmental changes. Generally these data critical application that affect the billing require accuracy of better than 5 PPM (Parts per Million).

Vendors often use cheaper crystals on boards that are cheap but may not have accuracy better than 100 PPM. It is desirable to provide a means of compensation built into the RTC to compensate for the inaccuracies due to the factors mentioned in previous section.

The next few sections describe some of the techniques that can be applied in an RTC to compensate for the clock frequency change with temperature.

3. Compensation basics

Figure 1 shows a frequency variation vs. temperature curve for a typical 32 kHz crystal.

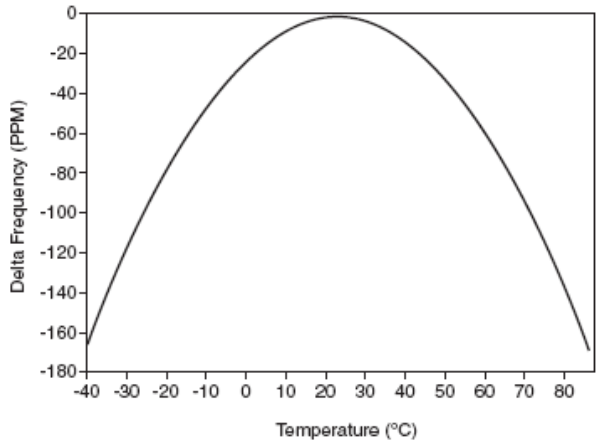


Figure 1: Variation in frequency with temperature

Characteristic of the Curve:-

- Peak drift of more than 110 PPM occurs at the temperature extremes of –40 °C and +85 °C.
- The crystal frequency peaks close to 25 °C
- Frequency falls in a parabolic manner above and below 25 °C
- Results may slightly vary from vendor to vendor

Looking into various datasheets of crystals from same vendor, it is apparent that aging characteristics are same for low and high precision versions of the crystal.

With an on-chip temperature sensor that reads the temperature in real time from the chip, the temperature drift can be calculated and relative number of pulses can be added or removed in a timing window.

Now since the variation in frequency with temperature is not a linear equation (second order equation) it is complex to calculate the number of pulses to be added or removed in hardware.

Solving the parabolic equation would mean to implement DSP functionality for accurately calculating number of corresponding pulses for a temperature drift. This costs both in area of chip as well as in power thereby reducing the battery life which is very critical. Often Energy Meters require a battery life of 15-20 years and to achieve that ultra low power consumption it is more efficient to do calibration in Software and corresponding compensation in Hardware. This would mean CPU would monitor the temperature sensor (over a period of time say a second or few seconds) and calculate the temperature drift. Software will calculate the numbers of pulses to be removed or added over a temperature drift with the help of a fixed lookup table (LUT) maintained in its memory. CPU would write back the number of pulses to be adjusted in the RTC's register. RTC hardware would compensate the clock based on the register value. The next section provides details on digital compensation technique.

4. Digital compensation technique

The oscillator clock is divided by RTC to generate a 1 pps (pulse per second) which clocks the time and date counters. As mentioned various factors tend to change the frequency of the oscillator clock and hence vary the 1 pps (or 1 Hz) clock.

To generate a 1 Hz clock, a counter is used to count the number of edges of the oscillator clock, which is equal to the frequency of the oscillator clock. For example, if we would count 32000 edges for a 32 kHz clock; 32768 edges for a 32.768 kHz clock to

generate the 1 Hz clock. At room temperature (i.e. 25 °C), there is no variation in the oscillator clock and time is accurate.

With the change in temperature the frequency of the oscillator clock also changes. Since we are counting a fixed number of oscillator clock edges, the total duration now does not remain 1 second. Thus, time cannot remain accurate. For example, if a change in temperature causes the frequency of 32.768 kHz crystal to change to 32.766 kHz then counting 32768 clock edges would give us 0.999984 Hz clock (i.e. 32768/32766 = 1.000061 seconds), which means that after 100000 seconds (or almost 1 day) the RTC would be ahead in time by 6 seconds and this may not be acceptable for any time critical application. Hence in order to bring the seconds clock (or 1 Hz clock) to correct value, the oscillator clock divider counter needs to count less number of oscillator clocks (i.e. till 32766). This in effect causes clock pulses to be subtracted from the oscillator clock. In order to do this, we shift the starting point of the divider counter to 2 (instead of 0) and count up till 32767. This will have the effect of dropping 2 oscillator clock cycles (i.e. count 32766 clocks) every second and give the correct 1 Hz clock output. This final 1 Hz output is called the Balanced or Compensated 1 Hz clock. There is no change to the oscillator clock frequency. This is shown in figure below.

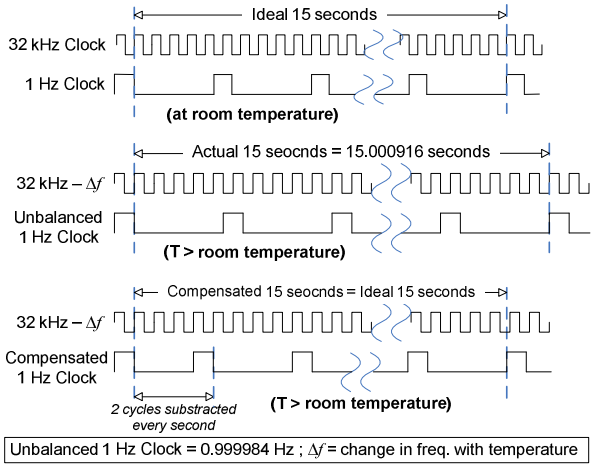


Figure 2: Error in clock and compensation

For all practical cases the change in frequency (Δf) might never be an integer value or is very small in value and hence it is important that we accumulate the delta in frequency over several seconds till it becomes close to an integer value with enough significant digits. Thus, this accumulated value is called the Correction or Compensation Value and the time over which it is accumulated is called the Compensation Interval. These two values are determined by the CPU and programmed into the RTC for it to digitally compensate the 1 Hz clock. The Correction Value is expressed as the 2's complement of the number of oscillator clock pulses to be added or removed from the divider counter to get an accurate 1 Hz clock.

Based on the above, the digital compensation hardware comprises the following main components:

a) *32 kHz Divider Counter* – running on the 32 kHz oscillator clock. Counter timeout generates the 1 Hz clock tick. The load value of this counter can be shifted to add or remove oscillator clock cycles for performing compensation. When compensation is disabled or the load value is zero, the counter simply rolls over to zero after timeout.

b) *Compensation Interval Counter* – running on 1 Hz clock and counts the period for compensation. This counter keeps track of the compensation cycle and its timeout indicates the end of a compensation cycle to the state machine. When compensation is disabled, this counter does not run.

c) *Compensation Control State Machine* – that supervises the entire compensation operation. The state machine ensures that:

   i) Compensation always starts near the second's boundary and not in between a second so as to prevent erroneous toggling in RTC counters and correction is transparent. This is the condition (load enable for RTC Divider counter and reset for Compensation Interval counter) when the counters are loaded with new values.

   ii) New compensation cycles start with new compensation values if changed.

The working of the compensation state machine is explained in the following flow chart.

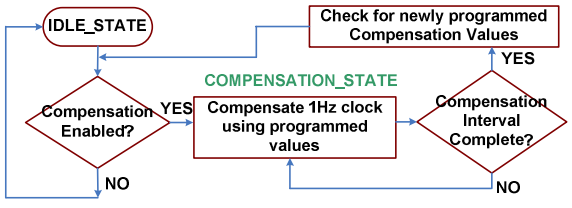


Figure 3: Compensation state machine

Steps:-

1) Remain in Idle state till compensation is not enabled. At this point, no compensation is

done and inaccuracies will remain in the 1 Hz clock. CPU can program the compensation registers at any point in time.

2) Enabling of compensation is checked at the second's boundary.
3) If enabled, compensation is done for the entire compensation interval.
4) At the end of compensation interval, the state machine checks for new programmed values and if compensation is still enabled, next compensation cycle starts with these newly programmed compensation values.
5) If compensation is disabled, the state machine keeps checking for the status of compensation registers at every second's boundary.

The block diagram of the above described digital compensation hardware is shown in Figure 4.

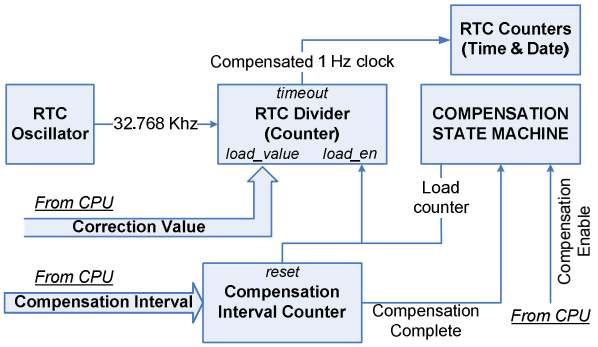


Figure 4: Digital compensation hardware

Using a single counter to generate compensated 1 Hz clock eliminates the need for any big dividers to add or delete pulses from the oscillator clock. This makes the above technique to consume low power.

5. Compensation accuracy

As mentioned in previous section, two values are important for Compensation (i.e. Correction Value and Compensation Interval). These also determine the accuracy of the compensation done. Accuracy of a compensation circuit is defined as the amount of error in clock frequency that can be corrected by it. Compensation value is the error to be removed that is accumulated over the Compensation interval. For example, if a circuit is able to correct 2.6 minutes over a period of one year then its accuracy is said to be 5 PPM. Thus, the accuracy of the above explained digital compensation circuit will be defined as the number of seconds lost or gained in a day or year to increase number of significant digits.

Mathematically speaking,

$$PPM = \frac{|Correction\_Value|}{(32768 \times Compensation\_Interval)} \times 1e6$$

Note:- In the above equation, the correction value (given in terms of oscillator clocks) has been divided by 32768 (Oscillator clock frequency at room temperature) to get the equivalent value in seconds.

Now, the accuracy of the circuit depends on how many cycles can be added or removed in the least compensation interval or the least number of cycles (which is of course 1) that can be added or removed in the largest compensation interval. These will determine the range of correction for the circuit. This in-turn depends on the width of the Correction (or Compensation) Value & Compensation Interval.

For example, a 5-bit Correction Value and 5-bit Compensation Interval can correct errors as low as 1 cycle per 32 seconds of compensation interval to as high as –16 cycles per 1 second of compensation interval. Thus, as per the above formula, the accuracy for this configuration is 0.954 PPM to 488.28 PPM. Thus, with just a 5-bit value we are able to achieve a big range of compensation. The user has the option of using a highly accurate crystal or a very low cost crystal which has more vulnerability towards frequency variation.

Table 1 shows the accuracy range of the above compensation circuit as we increase the width of the Correction Value and Compensation Interval values. It is assumed that widths of both values are equal and changed by the same amount.

Table 1: Accuracy of digital compensation circuit

Width	Correction Value	Compensation Interval	PPM
5	-16 to 15	1 to 31	0.954 to 488.28
6	-32 to 30	1 to 63	0.477 to 976.56
7	-64 to 63	1 to 127	0.238 to 1953.13

In the above table, the least correction value of 1 and max value have been used for PPM calculation and a correction value of 0 indicates the compensation is disabled.

Thus, increasing the width of Correction Value and Compensation Interval increases the accuracy (and range of compensation) but the current consumption will also go up. In applications, where current consumption is critical for longer battery life, a trade-off between accuracy and power consumption would have to be done.

6. Temperature compensated RTC application

As mentioned earlier, metering applications require Temperature Compensated RTC's for accurate time keeping and hence accurate billing. All calculations in metering applications are based on time for which the RTC must in all conditions be very accurate.

The heart of the energy meter comprises of a microcontroller having an embedded temperature compensated RTC with on-chip temperature sensor connected via an ADC to the CPU Core. The internal RTC runs on the 32 kHz clock supplied by the oscillator connected to a 32 kHz crystal. All energy measurement hardware is on chip and the CPU software computes the energy consumed per hour using the time from RTC. Figure 5 shows the block diagram of an energy meter microcontroller with a temperature compensated RTC inside it.

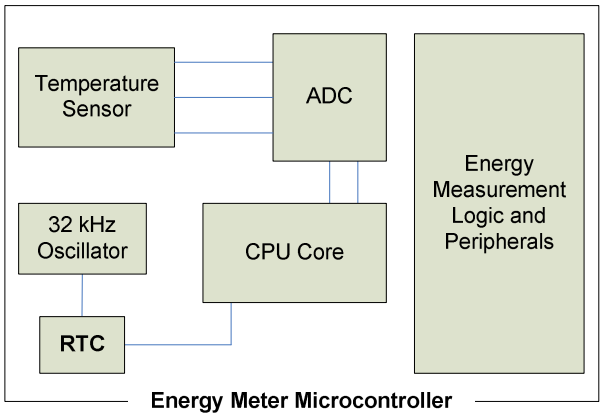


Figure 5: Temperature compensated RTC in an energy meter application

The energy meter as a whole system must perform both clock frequency error detection and correction in order to keep the real time clock accurate. The mechanism to achieve this is explained in the steps below.

DETECTION PHASE:

1) The detection phase of the clock compensation technique is done by using an on-chip or on-board temperature sensor connected to an ADC inside the microcontroller chip.
2) This microcontroller wakes up periodically and measures the temperate indicated by the temperature sensor via the ADC.
3) Based on this reading of temperature, the software running in the CPU Core of the microcontroller can calculate the necessary compensation required to keep RTC accurate. This calculation can be implemented in software using Lookup Tables (LUTs) stored in the CPU's memory. These LUTs are pre-programmed during the calibration of the energy meter and are based on the type of crystal being used and operating temperature.
4) The correction value and compensation intervals are then programmed into the RTC registers.

CORRECTION PHASE:

1) RTC latches these compensation values and does not pick new until the current compensation cycle is complete.
2) RTC starts to compensate with these values using the technique described above.
3) RTC continues the compensation operation until programmed with new compensation values or compensation is disabled.
4) If no value is programmed into the compensation register, the RTC shall continue to compensate with the existing values.

7. Conclusion

The compensation technique presented in this paper provides a solution that not only consumes less power but also provides a wide range of accuracy, even with small widths of Compensation Value and Compensation Interval values. This provides the user with the option of using cheaper crystals in their applications thereby lowering their product cost. Separating out the error detection phase into software allows the use of a common correction value irrespective of the type of factor affecting the frequency of the crystal.

For time critical applications that rely on real time data and require an accuracy of 5 PPM or better, Compensation for these applications is a must. These applications would need to compensate for factors that alter the 32 kHz clock frequency. There are several ways in which compensation can be done and the method presented in this paper targets the compensation technique that can be reusable across

technologies and simultaneously consume as less power as possible.

8. References

[1] Application Note AN3566 – Maxim Integrated Products (*http://www.maxim-ic.com/an3566*)

[2] Application Note AN10652 – NXP Semiconductors (*www.nxp.com*)

[3] Application Note AN1342.0 – Intersil Corp. (*www.intersil.com*)

[4] Application Note AN3938 – Maxim Integrated Products (*http://www.maxim-ic.com/an3938*)

Session 6A

Analog and Mixed Signal II

Systematic Methodology for High-Level Performance Modeling of Analog Systems

Soumya Pandit
Meghnad Saha Institute of Technology
Kolkata, India

Chittaranjan Mandal
Indian Institute of Technology
Kharagpur, India

Amit Patra
Indian Institute of Technology
Kharagpur, India

Abstract— **This paper presents a systematic methodology for construction of high-level performance models using least squares support vector machine. The transistor sizes of the circuit-level implementation of a component block along with a set of geometry constraints applied over them define the sample space. Optimal values of the model hyper parameters are computed using genetic algorithm. The novelty of the methodology is that the models constructed with this methodology are accurate, fast to evaluate with good generalization ability and low construction time. The present methodology has been compared with two other standard methodologies and the novelties are clearly demonstrated with experimental results.**

I. Introduction

An important step of an analog design automation process is analog high-level design. This is defined as the translation of analog system-level specifications into a proper topology of component blocks, in which the specifications of all the component blocks are completely determined so that the overall system meets its desired specifications [1]. There are two broad types of design methodologies available in literature [2] to address the problem of analog high-level design: optimization-based methodology and library-based methodology. An optimization-based analog high-level design methodology has two primary components: a search algorithm and a high-level performance estimator. A high-level performance estimation model is a function that estimates the performance of an analog component block when some high-level design parameters of the block are given as inputs [3].

Analog performance models constructed with regression technique are generally fast to evaluate and the accuracy with respect to real circuit-level simulation results is also good. This technique is therefore, often used for construction of performance models [4], [5], [6]. There are two types of regression-based technique – parametric regression technique and non-parametric regression technique. In parametric regression technique, a parameterised model is first proposed by the model developer and the values of the parameters are then fitted by some least-square error optimisation so that the model response matches closely the response of the real circuit. In non-parametric regression technique, a training network (e.g., support vector machine, artificial neural network) is used that is being trained with SPICE simulation results of the real circuit until the response of the network matches closely enough the response of the real circuit. An important advantage of a non-parametric regression technique over a parametric technique is that it does not require any model template. However, a major limitation of the non-parametric technique is that, the generalization ability of the constructed models is often not good. In addition, the model construction time is generally high which increases the design overhead.

In this work, we have developed a methodology for generation of high-level performance models for analog system using least squares support vector machine (LS-SVM) technique. The novelty of the methodology is that the constructed models are accurate, fast to evaluate with good generalization ability and low construction time. This methodology can be used in conjunction with an optimization procedure to develop a procedure for high-level topology sizing/optimization.

The rest of the paper is arranged as follows. Section II reviews some related works. Section III presents the necessary preliminary concepts. The methodology is described in detail in Section IV. Experimental results are provided in Section V and finally conclusion is drawn in Section VI.

II. Related Work

A fairly complete survey of related works is given in [7]. An analog performance estimation (APE) tool for high-level synthesis of analog integrated circuits is described in [8]. It takes the design parameters of an analog circuit as inputs and determines its performance parameters (e.g., power consumption, thermal noise) along with anticipated sizes of all the circuit elements. A power estimation model for ADC using empirical formulae is described in [3]. The estimators are fast to evaluate. However, the accuracy with respect to real simulation results under all conditions is off by orders of magnitude. The technique for generation of posynomial equation-based performance estimation models for analog circuits like opamps, multistage amplifiers, switch capacitor filters, etc., is described in [9]. An automatic procedure for generation of posynomial models using fitting technique is described in [4]. A neural network based tool for automated power and area estimation is described in [6]. Circuit simulation results are used to train a neural network model, which is subsequently used as an estimator. Fairly recently, support vector machine (SVM) has been used for modeling of performance parameters for RF and analog circuits [10], [5].

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.26

III. Preliminaries

A. High-Level Performance Model

Let us consider an analog system defined by a set of specification parameters $\bar{X}$ (e.g., gain, bandwidth of the system) and performance parameters $\bar{\rho}$ (e.g., input referred noise, power consumption). These two parameters are related as

$$\bar{\rho} = \bar{\mathcal{P}}(\bar{X}) \tag{1}$$

where $\bar{\mathcal{P}}$ is referred to as the set of high-level performance models of the system. Note that, a high-level performance model is different from circuit-level performance model in the sense that for high-level model, the input parameters are specification parameters of the component-blocks, whereas for circuit-level performance models, the input parameters are transistor sizes and/or biasing.

The important requirements for a good high-level performance model are: (i) The model needs to be low dimensional. Only those specification parameters are to be considered as inputs which have dominant contributions on a performance parameter to be estimated. (ii) The predicted results need to be accurate. The model accuracy is measured as the deviation of the model predicted value from the true function value. The function value in this case is the performance parameter obtained from transistor level simulation. (iii) The evaluation time must be short. This is measured by the CPU time required to evaluate a model. (iv) The time required to construct an accurate model must be small, so that the design overhead does not become high. This is relatively harder to quantify. This process involves both applying design knowledge to setup testbench circuit and design variable selection and computational time needed to use an algorithm to train a model. As a rough estimate, the construction cost is measured as

$$T_{\text{construction}} = T_{\text{data generation}} + T_{\text{training}} \tag{2}$$

where the terms are self explanatory. There exists a trade-off between these requirements since a model with lower prediction error generally takes more time for construction and evaluation.

B. Least Squares Support Vector Regression

In recent years, the support vector machine (SVM), as a powerful new tool for data classification and function estimation, has been developed [11]. Suykens and Vandewalle [12] proposed a modified version of SVM called least squares SVM. In this subsection, we briefly outline the theory behind the LS-SVM as function regressor.

Consider a given set of training samples $\{x_k, y_k\}_{k=1,2,...,N_{tr}}$ where x_k is the input value and y_k is the corresponding target value for the k^{th} sample. With a SVR, the relationship between the input vector and the target vector is given as

$$\hat{\bar{y}}(x) = w^T \phi(x) + b \tag{3}$$

where ϕ is the mapping of the vector $\bar{x}$ to some (probably high-dimensional) feature space, b is the bias and w is the weight vector of the same dimension as the feature space. The mapping $\phi(\bar{x})$ is generally non-linear which makes it possible to approximate non-linear functions. The approximation error for the k^{th} sample is defined as

$$e_k = y_k - \hat{y}_k(x_k) \tag{4}$$

The minimization of the error together with the regression is given as

$$\min \;\; \mathcal{J}(w,e) = \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{k=1}^{N_{tr}} e_k^2 \tag{5}$$

with equality constraint

$$y_k = w^T \phi(x_k) + b + e_k, \;\; k = 1, 2, ..., n \tag{6}$$

where N_{tr} denotes the total number of training data sets and the suffix k denotes the index of the training set, i.e., k^{th} training data, γ is the regularization parameter. LS-SVM considers the optimization problem to be a constrained optimization problem and uses dual Lagrangian-based formulation

$$\mathcal{L} = \mathcal{J}(w,e) - \sum_{k=1}^{N_{tr}} \alpha_k \left(w^T \phi(\bar{X}_k) + b + e_k - \rho_k \right) \tag{7}$$

and applying 'kernel trick', we arrive at the final model [12]

$$\hat{\bar{y}}(\bar{x}) = \sum_{k=1}^{N_{tr}} \alpha_k K(x_k, x) + b \tag{8}$$

where $K(x_k, x)$ is the kernel function. The elegance of using the kernel function lies in the fact that one can deal with feature spaces of arbitrary dimensionality without having to compute the map $\phi(\bar{x})$ explicitly. The Gaussian kernel function defined as

$$K(x_k, x) = exp\left(- ||x_k - x|\,|^2 / \sigma^2 \right) \tag{9}$$

is used in the present work, where σ^2 denotes the kernel bandwidth.

IV. High-Level Performance Model Generation

In this section, we describe the various steps of the construction methodology in detail.

A. Sample Space and Design of Experiments

While choosing the set of inputs, only those specification parameters forming a set $\bar{X}' \subseteq \bar{X}$ which have dominant contributions to specific performance parameters $\bar{\rho} = \{\rho_1, \rho_2, ..., \rho_n\}$ are considered as inputs. This choice of inputs relies on the designer's knowledge depending upon the application system and the topology considered. The dominant specification parameters are referred to as the high-level design parameters. For ease of notation, the prime indicating the reduction is omitted in the rest of this paper. Both the inputs and the output of the performance model $\mathcal{P}$ are taken to be functions of a set of geometry parameters $\bar{\alpha}$ (transistor sizes) of a component block, expressed as

$$\bar{X} = \mathcal{R}(\bar{\alpha}) \tag{10}$$

$$\bar{\rho} = \mathcal{Q}(\bar{\alpha}) \tag{11}$$

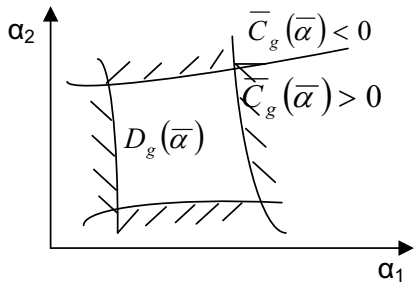


Fig. 1. 2D projection of a four dimensional sample space.

$\mathcal{R}$ and $\mathcal{Q}$ represents the mapping of the geometry parameters to electrical parameters. The multidimensional space spanned by the elements of the set $\bar{\alpha}$ is defined as circuit-level design space $\mathcal{D}_\alpha$.

A set of geometry constraints is applied on the transistor sizes to enclose a region within $\mathcal{D}_\alpha$, from which samples are extracted for training data generation. These geometry constraints include equality constraints as well as inequality constraints. The equality constraints, expressed as algebraic equations directly correlate the transistor sizes. For example, for matching purpose, the sizes of a differential pair transistors are equal. The equality constraints eliminate elements of the set $\bar{\alpha}$ and therefore reduce the dimension of the circuit-level design space $\mathcal{D}_\alpha$. The inequality constraints exclude additional portion of the reduced design space $\mathcal{D}_\alpha$, (correct notation is $\mathcal{D}_{\alpha'}$, which we avoid for ease of notation) without further reducing its dimension. The inequality constraints are usually given as box constraints, i.e., in the form of lower bounds and upper bounds. The lower bounds are determined by the feature size of a technology. The upper bounds are selected such that the transistors are not excessively large. With elementary algebraic transformations, all the geometry constraints are combined into a single non-linear vector inequality, which is interpreted element wise as:

$$\bar{C}_g(\bar{\alpha}) \geq 0 \;\Leftrightarrow \forall_{i\in\{1\ldots q\}} C_{gi}(\bar{\alpha}) \geq 0 \tag{12}$$

These constraints as functions of $\bar{\alpha}$ define a space, which we call as a sample space $\mathcal{D}_g$, defined as

$$\mathcal{D}_g = \{\bar{\alpha} \,|\, \bar{C}_g(\bar{\alpha}) \geq 0\} \tag{13}$$

Clearly $\mathcal{D}_g \subset \mathcal{D}_\alpha$. A two dimensional projection of a four dimensional sample space is illustrated in Fig. 1. Within the sample space, the circuit performance behavior becomes weakly non-linear [13]. Therefore, simple sampling strategies are used to construct models with good generalization ability.

The transistor sizes for generating training data corresponding to $\bar{X}$ and $\bar{\rho}$ are restricted to $\mathcal{D}_g(\bar{\alpha})$. The data generation process is generally an expensive process. Strategies from design of experiments (DOE) provide a mathematical basis to select a limited but optimal set of sample points from the sample space for training data generation. In the present work, these points are generated using a Halton sequence

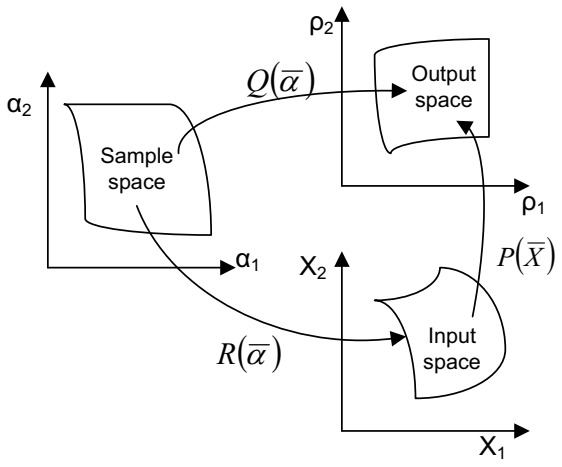


Fig. 2. Non-linear relation between the sample space and the input, output space.

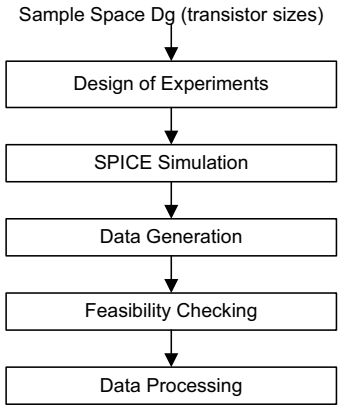


Fig. 3. An outline of the procedure for generation of training data.

generator. A Halton sequence generator is a quasi-random number generator which generates a set of uniformly distributed random points in the sample space. This ensures a uniform and unbiased representation of the sample space.

B. Training Data Generation

From (10) and (11), we see that the inputs $(\bar{X})$ and output $(\bar{\rho})$ of a high-level performance model $\mathcal{P}$ are functions of transistor-level parameters $\bar{\alpha}$. The inputs and the outputs are electrical parameters, whereas $\bar{\alpha}$ is a set of geometry parameters. The functions $(\mathcal{R}, \mathcal{Q})$ for mapping the geometry parameters to the electrical parameters are complex non-linear functions, considering the deep submicron effects of MOS transistors. In this work, these are achieved element-wise through a circuit simulation process, which is accepted to be the most accurate technique. The relationships are illustrated in Fig. 2. $\mathcal{R}$ and $\mathcal{Q}$ are used for generating the training data and $\mathcal{P}$ is the performance model to be constructed.

The training data generation process is outlined in Fig. 3.

For each input sample (transistor sizes) extracted from the sample space $\mathcal{D}_g$, the chosen circuit topology of a component block is simulated using SPICE through Cadence Spectre tool. The BSIM3v3 model is used for simulation, ensuring that the important deep submicron effects are considered while generating the training set. Depending upon the selected input-output parameters of an estimation function, it is necessary to construct a set of test benches that would provide sufficient data to facilitate automatic extraction of these parameters via postprocessing of SPICE output files. The commonly used SPICE analysis are ac analysis, transient analysis, dc sweep etc. The voltages and currents at the various nodes of the circuit are also measured. A set of constraints, referred to as feasibility constraints is then considered to ensure that only feasible data are taken for training.

The generated input-output data are considered to be feasible, if either they themselves satisfy a set of constraints or the mapping procedures $(\mathcal{R}, \mathcal{Q})$ through which they are generated satisfy a set of constraints. The constraints are as follows [13]:

1) Functionality constraints C_f : These constraints are applied on the measured node voltages and currents. They ensure correct functionality of the circuit and are expressed as

$$C_f = \{f_k(v, i) \geq 0 \;\; k = 1, 2, ..., n_f\} \qquad (14)$$

For example, the transistors of a differential pair must work in saturation.

2) Performance constraints C_p : These are applied directly on the input-output parameters, depending upon an application system. These are expressed as

$$C_p = \{f_k(\bar{\rho}) \geq 0 \;\; f_k(\bar{X}) \geq 0 \;\; k = 1, 2, ..., n_p\} \quad (15)$$

For example, the phase margin of an opamp must be greater than 45^0.

The total set of constraints for feasibility checking is thus $C = \{C_f \cup C_p\}$.

Data scaling is an essential step to improve the learning/training process of SVMs. The present methodology employs both linear scaling as well as logarithmic scaling depending upon the parameters chosen.

C. LS-SVM Hyperparameter determination

To obtain good performances, some parameters in the SVM models have to be chosen carefully. These parameters include: (i) the regularization parameter γ, which determines the trade-off between minimizing the training error and minimizing the model complexity and (ii) parameter (σ^2) of the kernel function that implicitly defines the non-linear mapping from the input space to some high-dimensional feature space. These higher level parameters are usually referred to as hyper parameters. In general, in any SVM problem, if the hyper parameters of the model are not well selected, the predicted results will not be good enough and the generalization ability will also be poor. Tuning of these hyper parameters is usually done by minimizing the estimated generalization error. The generalization error is a function that measures the generalization ability of the constructed models, i.e., the ability to predict correctly the performance of an unknown sample. In the present methodology, hold-out technique is used for estimating the generalization error. This is a simple technique for estimating the generalization error. The data set is separated into two sets, called the training set and the testing set. The SVM is constructed using the training set only. Then it is tested using the test data set. The test data are completely unknown to the estimator. The errors it makes are accumulated to give the mean test set error, which is used to evaluate the model.

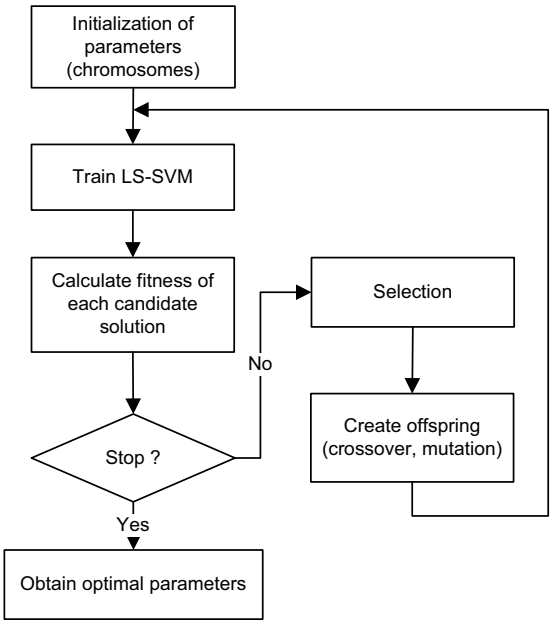


Fig. 4. Outline of GA-based hyperparameter selection procedure

The present methodology uses genetic algorithm (GA)-based technique for determining optimal values of the model hyperparameters. The task of selection of the hyper parameters is same as an optima searching task, and each point in the search space represents one feasible solution (specific hyper parameters). An outline of the GA-based process is shown in Fig. 4. The chromosomes consist of two parts, $\log_2 \gamma$ and $\log_2 \sigma^2$. During the evolutionary process of GA, a model is trained with the current hyper parameter values. The fitness of the chromosomes depends on the average relative error (ARE) calculated over the test samples. The fitness function is defined as

$$\text{fitness} = \frac{1}{ARE(\gamma, \sigma^2)} \qquad (16)$$

Thus, maximizing the fitness value corresponds to minimizing the predicted error. The ARE function is defined as

$$ARE = \frac{1}{N_{te}\rho'} \sum_{1}^{N_{te}} (\rho' - \rho) \qquad (17)$$

Here N_{te}, ρ and ρ' are the number of test data, the SVM estimator output and the corresponding SPICE simulated value,

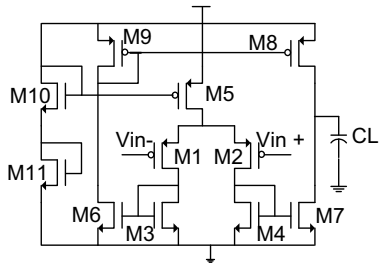


Fig. 5. PMOS OTA circuit

TABLE I
TRANSISTOR SIZES AND FEASIBILITY CONSTRAINTS FOR OTA

	Parameters	Ranges
Transistor Sizes Geometry Constraints	$W_1 = W_2$	$[280nm, 400\mu m]$
	$W_3 = W_4 = W_6 = W_7$	$[1\mu m, 20\mu m]$
	$W_8 = W_9$	$[280nm, 10\mu m]$
	W_5	$[1\mu m, 50\mu m]$
	C_L	$[1pF, 10pF]$
	Parameters	**Range**
Functional Constraints	$V_{gs} - V_{th}$	$\geq 0.1V$
	V_{op}	$\approx 0.9V$
	V_{off}	$\leq 2mV$
	Slew rate	$[0.1V/\mu s, 20V/\mu s]$
Performance Constraints	Bandwidth	$\geq 2MHz$
	DC Gain	≥ 70 dB
	Phase margin	$[45^0, 60^0]$

TABLE II
HYPER PARAMETER VALUES AND QUALITY MEASURES

Model	σ^2	γ	$ARE(\%)$		R		T_{tr}
			Training	Test	Training	Test	(min)
ρ_1	2.38	250.13	1.82	2.48	0.999	0.998	12.06
ρ_2	5.62	480.19	2.12	3.82	0.994	0.961	10.83
ρ_3	5.19	140.15	1.98	2.90	0.999	0.998	11.56

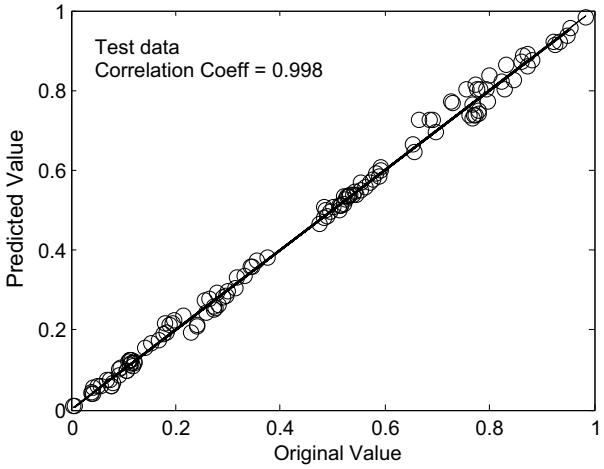


Fig. 6. Scatter plot of estimated and original values for the noise model with normalized test data.

respectively. The fitness of each chromosome is taken to be the average of five repetitions. This reduces the stochastic variability of the model training process in GA-based LS-SVM.

D. Quality Measures

Statistical functions are generally used to assess the quality of the generated estimator. The ARE function defined in (17) is one such measure. Another commonly used measure is the correlation coefficient. This is defined as follows:

$$R = \frac{N_{te}\sum \rho\rho' - \sum \rho \sum \rho'}{\sqrt{\left[N_{te}\sum \rho^2 - \left(\sum \rho\right)^2\right]\left[N_{te}\sum \rho'^2 - \left(\sum \rho'\right)^2\right]}} \quad (18)$$

The correlation coefficient is a measure of how closely the LS-SVM outputs fit with the target values. It is a number between 0 and 1. The higher the correlation coefficient, the better it is.

V. EXPERIMENTAL RESULTS

In this section, we provide experimental results demonstrating the methodology described above. The entire methodology has been implemented in Matlab environment and the training of the LS-SVM has been done using Matlab toolbox [14].

A. Experiment 1

A two stage CMOS operational transconductance amplifier (OTA) is shown in Fig. 5. The technology is $0.18\mu m$ CMOS process, with a supply voltage of $1.8V$. The transistor level parameters along with the various feasibility constraints are listed in Table I. We consider the problem of modeling input referred thermal noise (ρ_1), power consumption (ρ_2) and output impedance (ρ_3) as functions of DC gain (X_1), bandwidth (X_2) and slew rate (X_3). From the sample space defined by the transistor sizes, a set of 5000 samples is generated using a Halton sequence generator. These are simulated through ac analysis, operating point analysis, noise analysis and transient analysis using SPICE program. Out of all samples, only 1027 samples are found to satisfy the functional and performance constraints listed in Table I.

The estimation functions are generated using LS-SVM technique. The generalization errors are estimated through the hold-out method. The hyper parameters are computed through the GA-based technique. The population size is taken to be ten times the number of optimization variables. The crossover probability and the mutation probability are taken as 0.8 and 0.05 respectively. These are determined through a trial and error process. The hyper parameter values and the quality measures of the constructed models are reported in Table II. We find that the constructed models are quite accurate with average relative generalization ability error less than 4%.

The scatter plot of SPICE-simulated and LS-SVM estimated values for normalized test data of the noise model is shown in Fig. 6. The scatter plot illustrates the correlation between the SPICE simulated and the LS-SVM estimated test data. The correlation coefficient is very close to unity. Perfect accuracy would result in the data points forming a straight line along the diagonal axis.

TABLE III

COMPARISON BETWEEN OUR METHOD AND [5]

Model	Method	$ARE(\%)$		R	
		Training	Test	Training	Test
ρ_1	Our	1.82	2.48	0.999	0.998
	[5]	2.86	6.48	0.999	0.875
ρ_2	Our	2.12	3.82	0.994	0.961
	[5]	3.32	7.18	0.980	0.800
ρ_3	Our	1.98	2.90	0.999	0.998
	[5]	2.02	5.14	0.999	0.937

B. Experiment 2

In this experiment, we provide a comparison between our methodology of developing a performance model and that presented in [5]. The model hyper parameters are determined in [5] through heuristic technique. The same performance models are used for comparison purpose. The comparison results with respect to average relative generalization error (ARE), correlation coefficient (R) are reported in Table III. We observe from the comparison results that the generalization ability of the model constructed with our methodology is better than that constructed through [5] technique. This is because of the optimal choice of LS-SVM hyper parameters in our methodology through an optimization process.

C. Experiment 3

Here we present a comparison between our methodology and the EsteMate technique [6]. The power consumption model is reconstructed using the EsteMate technique. A set of 5000 samples is considered. For each selected sample, an optimal sizing is performed with a simulated annealing-based optimization procedure and standard analytical equations. The performances of each configuration is measured within the optimization process and are checked for feasibility. Out of all samples, only 3205 samples are accepted and the rest are rejected. The determination of the training set took 10 hours of CPU time. The training is done through an artificial neural network structure with two hidden layers. The comparative results are shown in Table IV. The generalization ability of our methodology is better than that of [6]. This is becuase of the use of SVM in our methodology. The generalization ability of SVM is found to be better than that of neural network [12]. In EsteMate, for each sample, a complete circuit sizing task using a global optimization algorithm is required for generation of the training data. This is usually prohibitively time consuming. On the other hand in our method, simple circuit simulations using the sampled transistor sizes are required for data generation. Feasibility checking are done afterwards Therefore, the cost of training data generation in our method is much less compared to that in the EsteMate methodology. This is evident from the experimental results also.

VI. CONCLUSION

This paper presents a systematic methodology for generation of analog high-level performance model using LS-SVM.

TABLE IV

COMPARISON BETWEEN OUR METHODOLOGY AND ESTEMATE [6]

Method	# Samples		$ARE(\%)$		Generation time	Training time
	Training	Test	Training	Test		
Our	821	206	2.12	3.82	14 min	10.83 min
[6]	2564	641	2.88	6.53	10 hour	21 min

The transistor sizes along with a set of feasibility constraints applied over them define the sample space. The SVM hyper parameters are determined through GA-based optimization technique. The quality of the constructed models is estimated by comparing the predicted performances with actual circuit-level simulation results. The novelty of present methodology is that the models constructed with this methodology are accurate, fast to evaluate with good generalization ability and low construction time. The methodology has been compared with other standard methodologies and the advantages of our methodology have been demonstrated with experimental results. The current methodology can be used in conjunction with an optimization procedure to develop a procedure for high-level topology sizing/optimization.

REFERENCES

[1] Georges.G.E. Gielen. Modeling and Analysis Techniques for System-Level Architectural Design of Telecom Front-Ends. *IEEE Trans. MTT*, Vol.50:pp.360–368, January 2002.

[2] S.Donnay et.al. High-level synthesis of analog sensor interface front-end. In *Proc. of ED&TC*, pages 56–60, 1997.

[3] E. Lauwers and Georges Gielen. Power Estimation Methods for Analog Circuits for Architectural Exploration of Integrated Systems. *IEEE Trans. VLSI Systems*, Vol.10:pp.155–162, April 2002.

[4] W. Daems, G. Gielen, and W. Sansen. Simulation-Based Generation of Posynomial Performance Models for the Sizing of Analog Integrated Circuits. *IEEE Trans. CADICS*, Vol.22:pp.517–534, May 2003.

[5] T. Kiely and G. Gielen. Performance Modeling of Analog Integrated Circuits using Least-Squares Support Vector Machines. In *Proc. of DATE*, pages 448–453, Feb 2004.

[6] G.V. Plas, J. Vandenbussche, G. Gielen, and W. Sansen. EsteMate: A Tool for Automated Power and Area Estimation in Analog Top-down Design and Synthesis. In *Proc. of CICC*, pages 139–142, May 1997.

[7] Rob.A. Rutenbar, Georges.G.E. Gielen, and J.Roychowdhury. Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs. *Proceedings of the IEEE*, Vol.95:pp.640–669, March 2007.

[8] A. Doboli, N. Dhanwada, A. Nunez-Aldana, and R. Vemuri. A Two-Layer Library-Based Approach to Synthesis of Analog Systems from VHDL-AMS Specifications. *ACM Trans. DAOES*, Vol.9:pp.238–271, April 2004.

[9] P. Mandal and V. Visvanathan. CMOS Op-amp Sizing Using a Geometric Programming Formulation. *IEEE Trans. CADICS*, Vol.20:pp.22–38, January 2001.

[10] X. Ren and T. Kazmierski. Performance Modeling and Optimization of RF Circuits using Support Vector Machines. In *Proc. of MIXDES*, pages 317–321, 2007.

[11] V Vapnik. *Statistical Learning Theory*. Springer, New York, 1998.

[12] J.A.K Suykens, T.V Gestel, J.D Brabanter, B.D Moor, and V. Joos Vandewalle. *Least Squares Support Vector Machines*. World Scientific, 2002.

[13] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich. The Sizing Rules Method for Analog Integrated Circuit Design. In *IEEE/ACM ICCAD*, pages 343–349, 2001.

[14] LS-SVM Toolbox. http://www.esat.kuleuven.ac.be/sista/lssvmlab, February 2003.

2009 22nd International Conference on VLSI Design

A Comparison of Approaches to Carrier Generation for Zigbee Transceivers

Leburu Manojkumar, Arun Mohan, & Nagendra Krishnapura
Department of Electrical Engineering, Indian Institute of Technology, Madras, Chennai 600 036, India

***Abstract*— Two methods for generating in phase and quadrature local oscillator signals at 2.4 GHz for Zigbee transceivers are investigated. In one method, the output of a 4.8 GHz LC VCO is divided by two to obtain I and Q phases at 2.4 GHz. In another method, outputs of a four stage differential ring VCO at 1.2 GHz are appropriately multiplied to obtain I and Q phases at 2.4 GHz. These circuits are designed and laid out in a 0.18 μm CMOS process and they operate from a 1.8 V power supply. The former architecture occupies 0.052 mm^2, consumes 7.56 mW, and has a phase noise of -117 dBc/Hz at 3.5 MHz. The latter occupies 0.021 mm^2, consumes 9 mW, and has a phase noise of -97 dBc/Hz at 3.5 MHz. Temperature variations of the ring oscillator are minimized using a combination of constant current and constant g_m biasing.**

I. Introduction

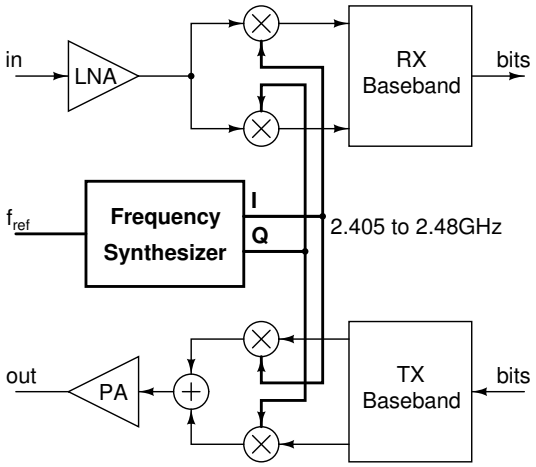


Fig. 1. Zigbee transceiver block diagram

Zigbee (IEEE 802.15.4) standard[1] is intended for low data rate sensor network applications. It specifies 16 channels in the 2.4 GHz band. Fig. 1 shows the block diagram of a possible Zigbee transceiver architecture. The receiver consists of a low noise amplifier followed by downconversion mixers and the baseband circuitry (channel selection filtering and digitization). The transmitter consists of baseband circuitry that does pulse shaping and digital to analog conversion followed by upconversion and power amplification. Both the receiver and the transmitter require a local carrier for frequency conversion. The local carrier will be at the signal frequency in a direct conversion architecture or offset by the intermediate frequency (IF) in a heterodyne architecture. In either case, a frequency synthesizer with in-phase (I) and quadrature (Q) outputs is required. Conventionally, LC oscillators are used in on chip radios due to constraints on phase noise and power dissipation. In this paper, we will investigate the possibility of using a ring oscillator based carrier generator for Zigbee transceivers.

The paper is organized as follows. In section II, we outline the requirements of the local oscillator in a Zigbee transceiver. In section III, we discuss two possible architectures for quadrature carrier generation. Sections IV and V show the design details of these architectures. The design of a frequency synthesizer around these I/Q generators is shown in section VI. In section VII, we compare the two architectures in terms of their performance and chip area.

II. Local oscillator requirements

IEEE 802.15.4 standard specifies 16 channels in the 2.405-2.48 GHz band with a spacing of 5 MHz. The frequency synthesizer (Fig. 1) is required to generate these frequencies offset by the IF. In our case, we have assumed a direct conversion architecture[2] and the frequency synthesizer is designed to generate the channel center frequencies.

To meet the standards, the frequency synthesizer has to have the following requirements: ≤ -92 dBc/Hz phase noise at 3.5 MHz offset and ≤ 200 μs settling time to 40 ppm accuracy. Additionally, it should be able to drive the mixers in the transmitter or the receiver (only one would be operating at any given time). At 3.5 MHz offset, the phase noise of the synthesizer is solely due to the phase noise of the voltage controlled oscillator (VCO). Therefore the VCO must meet this specification. The settling time requirement imposes a lower limit on the bandwidth of the phase locked loop (PLL) used for the frequency synthesizer. The oscillator must drive the mixers and the programmable divider used in the feedback path of the phase locked loop frequency synthesizer.

One of the concerns with a direct conversion architecture is that the transmitted signal can be coupled to the receiver input, and be fed back to the frequency synthesizer through the LNA and mixers, due to finite isolation between different ports of each block (even if the receiver is off). If the oscillator in the frequency synthesizer is at the carrier frequency (2.4 GHz in this case), it will be very sensitive to the modulated transmit signal leaking back into it at the same frequency. This can deteriorate the phase noise of the oscillator significantly. Therefore we have considered only those architectures in which the oscillator is not operating in the transmitted signal band.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.50

III. Architectures for I/Q carrier generation

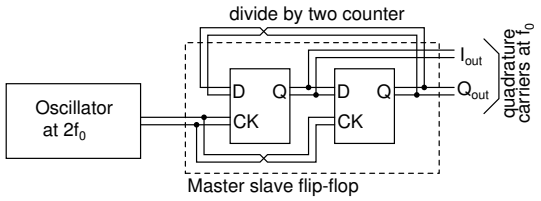


Fig. 2. IQ generation by dividing a double frequency waveform

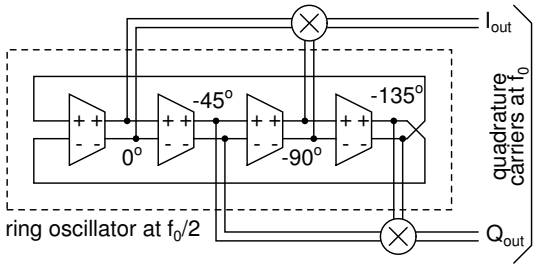


Fig. 3. IQ generation by doubling half frequency waveforms

One of the methods for generating quadrature waveforms is shown in Fig. 2. In this case, an oscillator at twice the desired frequency is used to drive a divide by two counter using a master-slave flip flop. The outputs of the master and the slave latches are separated by half the input period (assuming an input duty cycle of 50%), or equivalently, a quarter of the output period. These can be used as quadrature carriers in the transceiver.

An alternative method for generating quadrature waveforms is shown in Fig. 3. This is based on a four stage differential ring oscillator at half the desired frequency. The waveforms in such an oscillator are 45° apart. Multiplying alternate stage outputs which are separated by 90° results in frequency doubling. From the four waveforms, two such double frequency waveforms can be obtained and they will be 90° apart at the doubled frequency.

When the two schemes are compared, we can observe the following:

- The oscillator in Fig. 2 is at four times the frequency of the oscillator in Fig. 3. This leads to a four times higher power dissipation in the former if ring oscillators are used for both.
- Only one output is required from the oscillator in Fig. 2 whereas Fig. 3 requires a four stage oscillator. Using inductors in each stage of the latter results in a very large chip area.

From the above it appears that an LC oscillator is the only realistic option in Fig. 2 if its power dissipation has to be comparable to that of Fig. 3. Also, a ring oscillator is the only realistic option for the latter if its area of has to be comparable to that of the former. Consequently we can expect to achieve lower phase noise or lower power dissipation with Fig. 2, and a lower area with Fig. 3. However, a definitive comparison can be made only after designing both for a given set of specifications. Since the phase noise specifications for the Zigbee standards are not very stringent[1], the poorer noise performance of Fig. 3 may not be important.

In the rest of the paper, we present the design of quadrature carrier generators using the above schemes and compare the two. The power consumption is minimized while ensuring that the IEEE 802.15.4 requirements are met and that the circuits are capable of driving the load formed by the programmable divider in the feedback path of the frequency synthesizer and the transmit or receive mixers. From the layout of these blocks, the total load on the quadrature outputs was estimated to be 180 fF each.

IV. IQ generation based on dividing a double frequency waveform

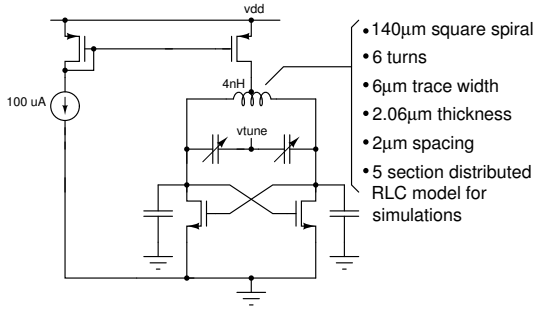


Fig. 4. LC VCO schematic

Fig. 4 shows the schematic of the LC oscillator used in Fig. 2[3]. The negative resistance of the cross coupled nMOS pair cancels the tank circuit's loss to result in sustained oscillations across the LC circuit. The tank circuit is made of a 4 nH differential square spiral inductor, MOS accumulation varactors and some fixed capacitance across the tank. The bias current is adjusted to have a differential swing of 600 mVpp across process and temperature variations. To minimize power consumption, the tank impedance, and hence the inductance must be maximized. But, beyond a certain value of inductance, the required tank capacitance becomes so small that it is dominated by parasitics, and adequate tuning range cannot be obtained. To minimize phase noise with a given tank circuit, the magnitude of the negative conductance must be minimized while ensuring reliable start up. In our design, we have sized the transistors such that, across process and temperature variations, the smallest magnitude of the negative conductance is 1.5 times the equivalent parallel conductance of the tank circuit. The MOS accumulation varactors are laid out in a differential configuration to maximize their quality factor[4].

Fig. 5 shows the schematic of the latch used in the divide by two counter (Fig. 2). The latch uses fully differential current mode logic (CML) with active inductor loads. The active inductors are formed by nMOS load transistors with resistances in series with their gates[5]. The high voltage bias

[1]Compared to cellular phone or wireless LAN standards

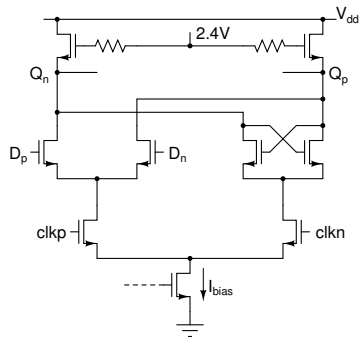


Fig. 5. CML latch used in the divide by two counter

is generated using a charge pump. The resistors are adjusted to obtain sufficiently fast rise times across all process corners. The bias currents are chosen such that the latch can drive the following buffer. Fig. 6 shows the buffer used to drive

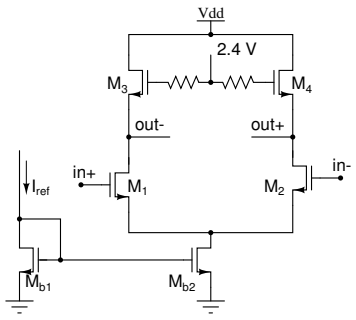


Fig. 6. Buffer to drive the programmable divider and the mixers

the programmable divider and the mixers. It comprises a differential pair with an active inductor load. The bias current is chosen to be sufficient to drive a 180 fF load.

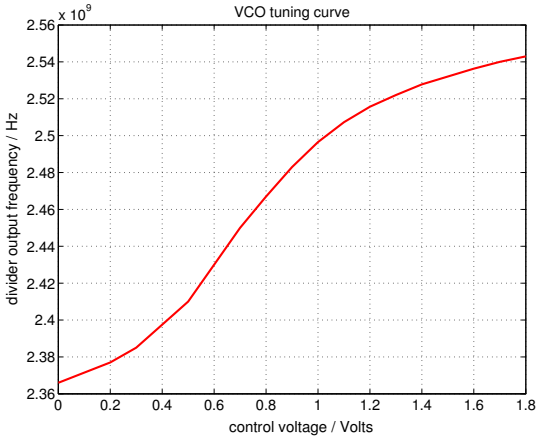


Fig. 7. Frequency (at the divider output) vs. control voltage

Fig. 7 shows the simulated frequency versus control voltage characteristics of the combination of the LC oscillator and the divider. The maximum VCO gain (K_{vco}) is 200 MHz/V. The waveforms at the output of the VCO and

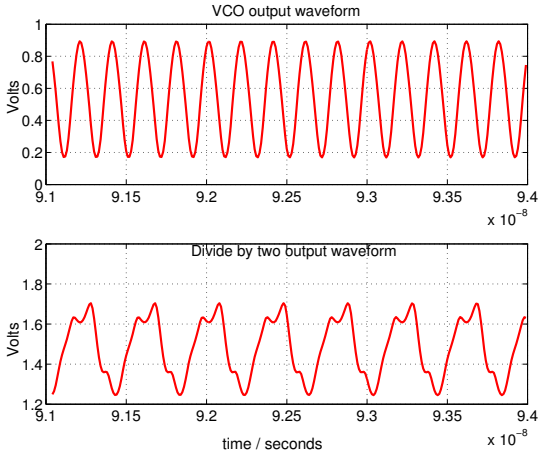


Fig. 8. VCO and divider output waveforms

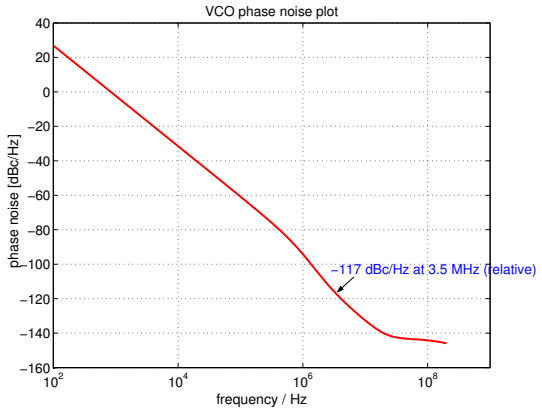


Fig. 9. Phase noise at the divider output

the frequency divider are shown in Fig. 8. Fig. 9 shows the phase noise of the VCO as measured at the divider output[2]. The phase noise is -117 dBc/Hz at 3.5 MHz.

V. IQ GENERATION BASED ON DOUBLING HALF FREQUENCY WAVEFORMS

Fig. 10 shows the delay cell used in the architecture in Fig. 3. It consists of a differential pair (M_{3-4}) loaded by pMOS transistors in triode region (M_{1-2}). The oscillation frequency is varied by varying the transconductance of the differential pair through the tail current. The gate bias of the pMOS transistors is adjusted using a replica bias circuit (M_{7-10}) which maintains the drain voltage at 1.3 V. This ensures a constant swing as the VCO is tuned, and also across process and temperature variations. The replica circuit is common to the four stages of the oscillator. The tail current is derived such that the variations of the VCO characteristics across temperature are minimized. This is explained in more detail later. For sustained oscillations in a four stage oscillator, the delay cell needs to have a dc gain of at least $\sqrt{2}$. The

[2]Relative phase noise above 0 dB at very low frequencies is a an artifact of simulation in SpectreRF.

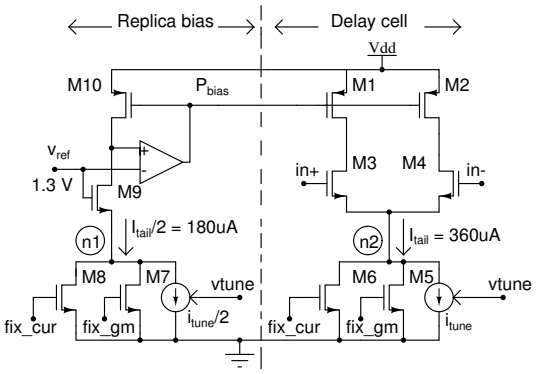


Fig. 10. Schematic of the delay cell with replica bias. "fix_cur" and "fix_gm" are derived from a constant current bias and a constant g_m bias respectively.

transistor sizes are chosen such that the lowest dc gain across process and temperature corners is greater than this limit.

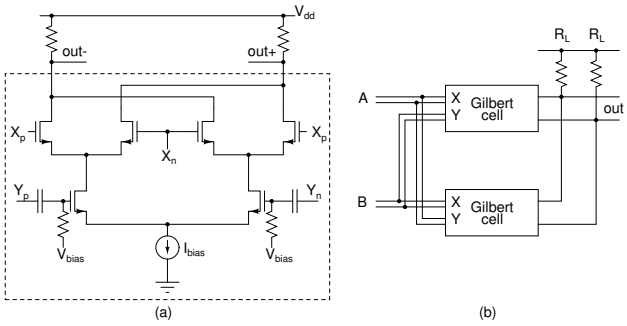


Fig. 11. (a) Gilbert multiplier cell (b) Multiplier with symmetric inputs

The multipliers in Fig. 3 are realized using the well known Gilbert cell (Fig. 11(a)). The disadvantage of this circuit is that the two inputs present different input impedances. If used as is in Fig. 3, the symmetry of the ring oscillator will be destroyed by unequal loading at its nodes. To overcome this problem, two Gilbert cells are used in parallel with each input driving the upper input of one cell and lower input of the other as shown in Fig. 11(b). This results in identical impedances at the two input ports A and B. The lower inputs (Y) are ac coupled and biased at the appropriate common mode level. The multipliers are followed by buffers (Fig. 6) to drive the mixers and the programmable divider.

Fig. 12 shows the frequency variation with temperature of the carrier generator (Fig. 3) using the delay cell in Fig. 10 and the doubler in Fig. 11(b) when the tail current is constant with temperature. A negative temperature coefficient of frequency is observed. Fig. 12 shows the frequency variation with temperature when the tail current is derived from a fixed g_m bias circuit[6]. In this case, as the temperature increases, the bias current is increased to maintain a constant g_m. A positive temperature coefficient is observed. Therefore, a constant current and a current from a fixed g_m bias circuit are added (using M_5 and M_6 in Fig. 10) in the right proportion to cancel the temperature coefficient. The resulting frequency variation with temperature is shown in Fig. 12 and is seen

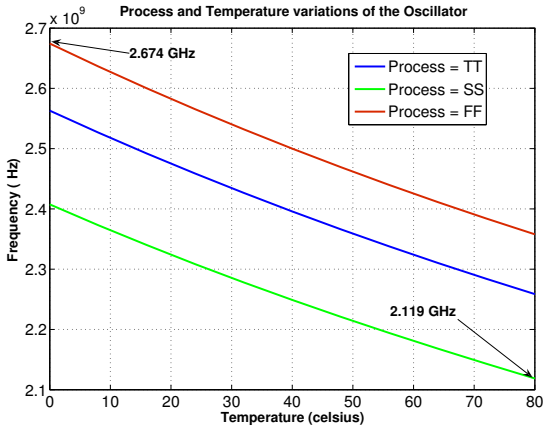


Fig. 12. Process and Temperature variations with fixed current biasing

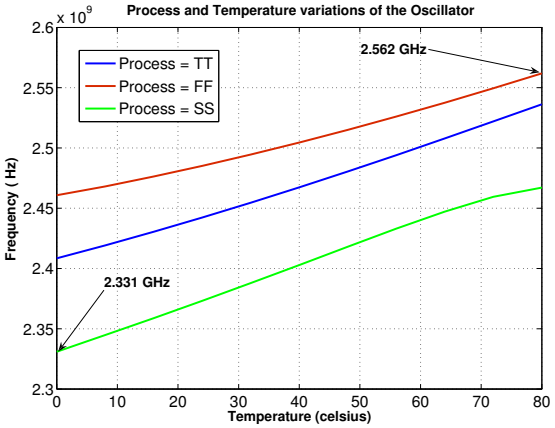


Fig. 13. Process and Temperature variations with fixed g_m biasing

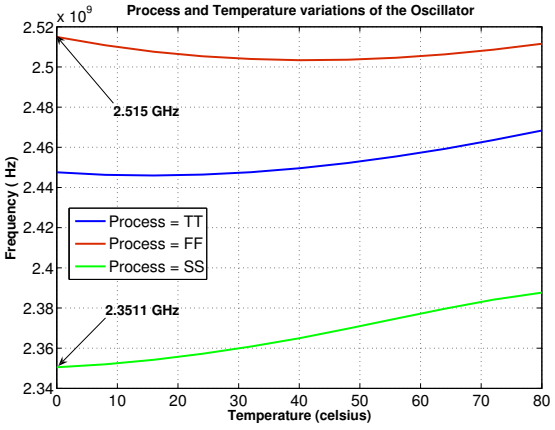


Fig. 14. Process and Temperature variations with mixed biasing

to be significantly smaller than with either constant current or constant g_m biasing. The current consumption of the oscillator increases with "slow" process and high temperatures due to this biasing scheme.

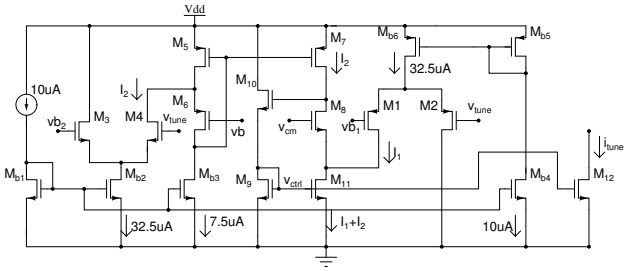


Fig. 15. Voltage to current converter

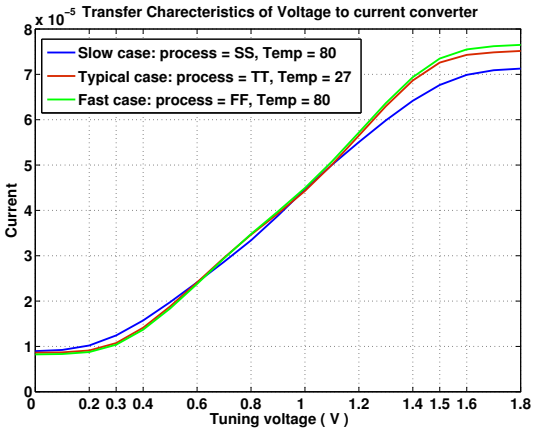


Fig. 16. Transfer characteristics of voltage to current converter

In the ring oscillator above, the frequency is varied using the tail current. In a frequency synthesizer using a charge pump and a passive loop filter, the output is a voltage. To control the ring oscillator using this voltage, a voltage to current converter is used (Fig. 15). To obtain a large range for the control voltage V_{tune}, it is applied to an pMOS differential pair $M_{1,2}$ and a nMOS differential pair $M_{3,4}$ with different reference levels ("vb1" and "vb2"). The currents from the two differential pairs are added to obtain the control current for the ring oscillator. For small values of V_{tune} only M_1 and M_2 are active and for large values of V_{tune}, only M_3 and M_4 are active. The characteristics of the V-I converter are shown in Fig. 16. The usable range of V_{tune} is from 0.3 V to 1.5 V.

Fig. 17 shows the frequency vs. voltage characteristics of the VCO with the doubler. The maximum gain is about 220 MHz/V. Fig. 18 shows the phase noise at the doubler output. Fig. 19 shows the output waveforms.

VI. Frequency Synthesizer Design

Fig. 20 shows the frequency synthesizer designed around the quadrature carrier generators described above. It is a standard type II phase locked loop[7] with $I_{cp} = 10\,\mu$A, $R_1 = 80\,\text{k}\Omega$, $C_1 = 220\,$pF, and $C_2 = 7\,$pF.

The frequency divider in feedback can be programmed to obtain division ratios from 481 to 495. The architecture of the programmable divider is shown in Fig. 21[8]. CML latches are used throughout. The signal frequency reduces as

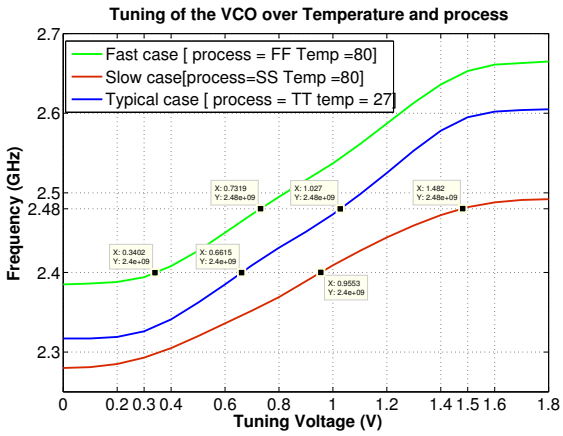


Fig. 17. Frequency vs. voltage at the doubler output

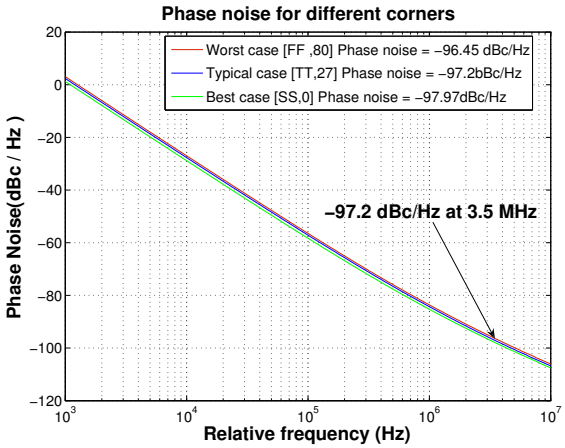


Fig. 18. Phase noise at the doubler output

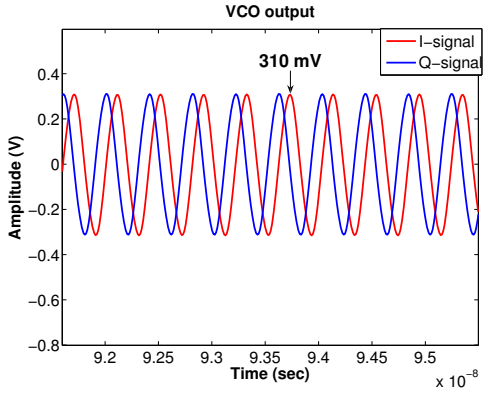


Fig. 19. Quadrature outputs of ring oscillator and doubler combination

one goes down the divider chain and the bias currents are reduced accordingly. The differential output is converted to single ended output before being fed to the phase frequency detector.

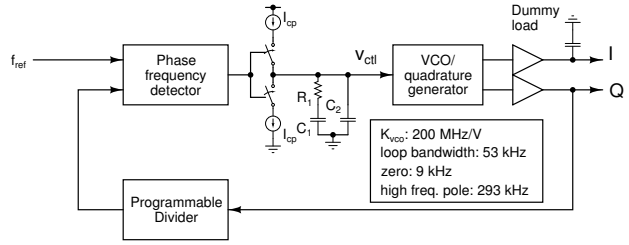


Fig. 20. Frequency synthesizer block diagram

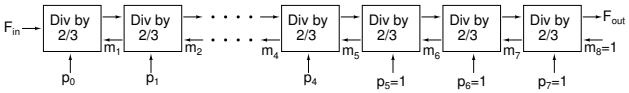


Fig. 21. Programmable frequency divider

VII. Simulation results

Both circuits are designed and laid out in a 0.18 μm CMOS process. Table I shows their salient features. The current consumption of the ring oscillator increases at high temperatures and "slow" processes due to the biasing scheme described earlier. The current consumption of the LC oscillator is independent of process and temperature variations.

The ring oscillator consumes up to 43% higher power than the LC oscillator. The LC oscillator occupies a 2.25$\times$ larger area. The phase noise requirements for a Zigbee transceiver can be met with a ring oscillator based architecture. As expected the LC oscillator achieves a much better phase noise.

Table II shows the simulated results of the frequency synthesizer built around the carrier generators described earlier. Since K_{vco} is the same in the two cases, the performance is the same with either architecture.

VIII. Comparison and conclusions

We have presented two approaches to IQ carrier generation for IEEE 802.15.4 Zigbee transceivers. We have designed circuits following both approaches in a 0.18 μm CMOS process and compared them. The comparison is made based on optimizing each design for low power while meeting the requirements of the standard and the load conditions expected in a transceiver. We conclude that it is possible to build a ring oscillator based frequency synthesizer for this standard with a small power penalty and a large area advantage over the LC based approach. Although the phase noise of the LC oscillator is much lower than necessary, the quality factor of the spiral inductor cannot be lowered (while retaining the inductance value) in an attempt to reduce its area, because, doing so reduces the equivalent parallel resistance of the tank circuit and reduces the amplitude of oscillation making it difficult to drive the following stages.

The power consumption of the LC oscillator is constrained by the highest inductance that can be realized while not being overwhelmed by parasitic capacitances. The power consumption of the ring oscillator is limited by the phase noise requirements. Technology scaling does not offer either oscillator significant advantages in terms of power consumption (for the same phase noise specifications). A small reduction in power consumption can be seen due to reduction of parasitic capacitances. The power consumption of the remaining blocks—frequency divider or the multipliers, and the buffers—will be reduced owing to reduction of parasitic capacitances and on chip load capacitances with technology scaling. This reduction will be proportional to the reduction in capacitance and will be the same for either architecture. Therefore we believe that the comparison presented in this paper will remain valid for other CMOS processes.

TABLE I

Performance summary of quadrature generators

	LC osc. + divider (Fig. 2)	Ring osc. + doubler (Fig. 3)
VCO	1 mA	1.44 mA
Bias circuit	—	0.355 mA
V-I converter	—	0.1 mA
Divider	1.8 mA	—
Multipliers	—	0.45 mA each
Buffers	0.7 mA each	0.83 mA each
Total current	**4.2 mA**	**4.455 mA (nom.) 6 mA (max.)**
Phase noise	**-117 dBc/Hz**	**-97 dBc/Hz**
Area	**360 μm x 140 μm**	**160 μm x 140 μm**
K_{vco}	**200 MHz/V**	**220 MHz/V**

TABLE II

Performance summary of the frequency synthesizer

Programmable divider	1.09 mA
Differential to single ended converter	22 μA
Phase frequency detector	23 μA
Charge pump	20 μA
Bias generation circuits	350 μA
Total current	**1.5 mA**
Settling time	**110 μs**
Area	**400 μm x 310 μm**
Reference feedthrough	**-39 dBc (5 MHz) -50 dBc (10 MHz)**

References

[1] "Wireless medium access control (MAC) and physical layer (PHY) specifications for low rate wireless personal area networks (LR-WPANs)", IEEE 802.15.4, available for download at http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf

[2] Behzad Razavi, *RF Microelectronics*, Prentice Hall, 1997.

[3] M. Tiebout, *Low power VCO design in CMOS*, Springer, 2006.

[4] A. S. Porret et al., "Design of High-Q Varactors for Low-Power Wireless Applications Using a Standard CMOS Process", *IEEE Journal of Solid-state Circuits*, pp. 337-345, vol. 35, no. 3, March 2000.

[5] E. Sackinger and W. C. Fischer, "A 3-GHz 32-dB CMOS limiting amplifier for SONET OC-48 receivers", *IEEE Journal of Solid-State Circuits,* pp. 1884-1888, vol. 35, no. 12, Dec. 2000.

[6] Behzad Razavi, *Design of Analog CMOS Integrated Circuits*, McGraw-Hill, August 2000.

[7] J. Craninckx and M. Steyaert, "A fully integrated CMOS DCS-1800 frequency synthesizer", *IEEE Journal of Solid State Circuits*, vol. 33, no. 12, pp. 2054-2065, 1998.

[8] C. Vaucher et al., "A family of low-power truly modular programmable dividers in standard 0.35 um CMOS technology," *IEEE Journal of Solid-state Circuits*, vol. 35, no. 7, pp. 1039-1045, 2000.

A 2.4Gbps-4.8Gbps XDR-DRAM I/O (XIO) Link

Vijay Khawshe, Kapil Vyas, Renu Rangnekar, Prateek Goyal, Vijay Krishna, Kashinath Prabhu, Pravin Kumar Venkatesan, Leneesh Raghavan, Rajkumar Palwai, Thrivikraman M, Kunal Desai, Abhijit Abhyankar
Rambus Inc. (India Design Center)
vkhawshe@rambus.com, kvyas@rambus.com

Abstract

This paper focuses on the design of a 2.4Gbps to 4.8Gbps link developed in TSMC65nmG+ technology, for the high speed and high throughput interface between XDR™ (Extreme data rate) DRAM and ASIC. Applications such as HDTV and high end graphics require high bandwidth interface between controllers and memory. This XDR I/O (XIO) link which is integrated in the controller, interfaces with the XDR™ DRAM and provides the very high per pin bandwidth. To maintain a constant transmit swing the link supports automatic calibration for the on-die termination (ODT) and driver circuit bias. The channel timing between, ASIC pin to XDR-DRAM pin, is calibrated for all the individual pins to de-skew any channel electrical timing differences to align the data transfer during Memory Read and Writes. This calibration is done periodically to maintain constant timing margin throughout the operation. The self biased [1] regulated PLL dual loop architecture based on [2] is used which minimizes the clock jitter and enables high speed operation. A novel programmable Voltage Control Oscillator is used here to work at wide range of frequencies. The cell with 8bit wide data bus and 16bit wide command bus, consumes 520mW at 4.0Gbps.

***Index Terms*—XIO, DQ, RQ, Regulated PLL, TSMC65nm g+**

1. Introduction

With increasing demand for bandwidth and latency in Processor to DRAM communication in the graphic and compute space, there is a strong need to enhance the capability of these interface I/Os, for higher bandwidth/throughput. Rambus XDR™ IO Cell (XIO) is a high-performance, low latency controller interface to XDR-based DRAM memory systems. The general-purpose cell is independent of the logical memory controller design, enabling support for a wide variety of memory applications needing high bandwidth and low latency. The implemented cell, in current reference, has 8-bit wide DQ (data) channel and 16-bit wide RQ (request) channel.

The DQ block takes 8 bytes of parallel data from the ASIC interface and serializes it in the single system clock cycle for 8-bit wide interface pins. This is possible by generating an internal clock which is 4x of the system clock and sampling the data at both edges of the clock to eventually transmitting the data at 8x of the system clock. Similarly the RQ commands are sent at 2x of the system clock after sampling at both the edges of clock.

Timing calibration is done with the help of programmable phase mixers and calibration logic called FlexPhase™. Using these FlexPhase circuits inside the ASIC's XIO cell, individual bit timing is de-skewed, from the ASIC to the XDR DRAM, to present a synchronized parallel data word at the DRAM. For the data transfer from DRAM to ASIC, FlexPhase circuits in XIO are used and no added circuits are needed on DRAM devices.

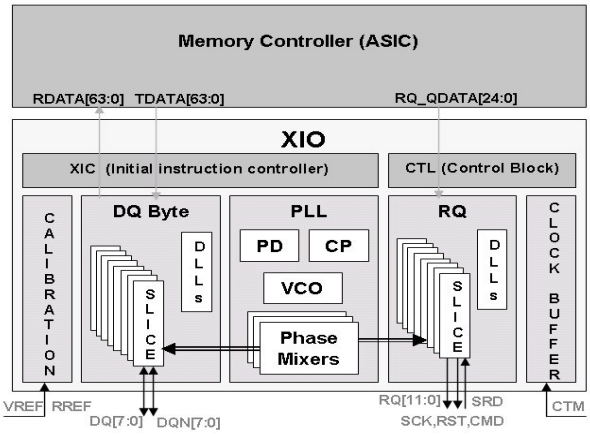


Fig-1 Top level XIO with all components

™ *XDR™ and FlexPhase™ are trademark of Rambus Inc.*

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.65

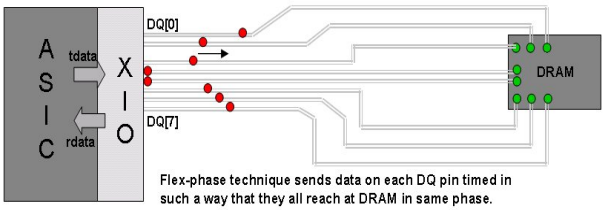


Fig-2 XIO-XDR interface with Flex phase circuit

Operation sequences called "initial calibration" are performed after the XIO is first powered up. These sequences set channel de-skewing timing, bias current, and termination values to their optimized values. Operations sequences called "periodic calibration" are performed at intervals, between sequences of normal memory read and write operations. These sequences adjust channel de-skewing timing, bias current, and termination values regularly to compensate for changing system conditions due to temperature etc. This way the system margins are maintained without causing bit errors.

2. Phase Locked Loop

2.1. Architecture

A self bias PLL based on dual loop architecture as mentioned in [2] is used here. The basic building blocks are shown in the Fig-3. Replica compensated regulator for the VCO [6] rejects high frequency noise without compromising on bandwidth of VCO regulator and unity-gain opamp loop for resistor of the RC filter in PLL loop ensures loop bandwidth will closely follow operating frequency [3]. The duty cycle corrector circuit based on common mode feedback tracks duty cycle error upto ±5%. As its bandwidth is almost 3 times larger than PLL loop bandwidth it can correct high frequency clock duty cycle to ±1%.

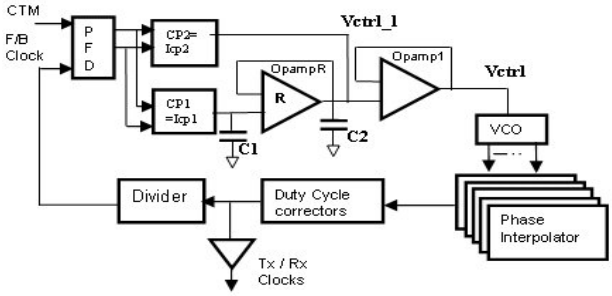


Fig-3 Phase Locked Loop diagram

However to ensure that loop parameter do not change with change in operating frequency it is required to keep control voltage variation range constant over VCO oscillation range. The output Tx/Rx clocks are function of reference clock called CTM.

Tclk/Rclk Frequency = 4 * (Frequency of CTM)

For 2.4Gbps/3.2Gbps/4.0Gbps/4.8Gbps outputs, the CTM frequencies are changed to 300/400/500/600MHz respectively. The PLL parameters are adjusted for the different frequency ranges and the VCO is switched for the different ranges as explained in next section.

2.2. VCO Circuit

To achieve multiple tuning ranges with single VCO ring oscillator, previously implemented methods were using switch capacitors. However it does not provide precise control of the device capacitors used, across different process corners with large variation in control voltages. Here a new technique (Fig-4) is proposed in which, instead of adding capacitors, tristate buffers are added in the feed forward path. Multiple different sized tristate buffers are added based on the frequency ranges required. These tristate inverters are connected as data anticipator, to boost oscillations.

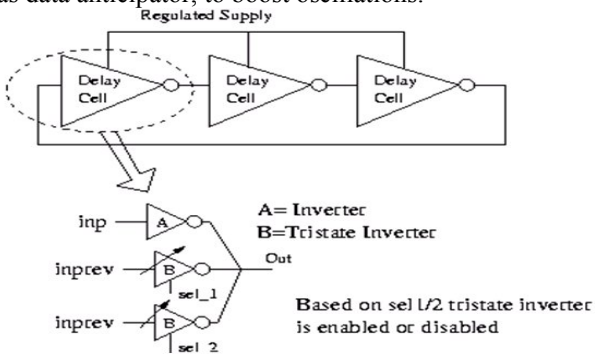


Fig-4 Programmable Voltage Controlled Oscillator

One of these buffers is kept ON, for a particular frequency. The buffer in ON condition reduces delay by feed forward mechanism and thus increase frequency of operation while in OFF state, act as capacitive load. Based on frequency of operation (i.e. 2.4Gbps/3.2Gbps/4.0Gbps/4.8Gbps) the buffer can be selected. This methodology will make VCO very less dependent on process variation and non-uniform parasitic.

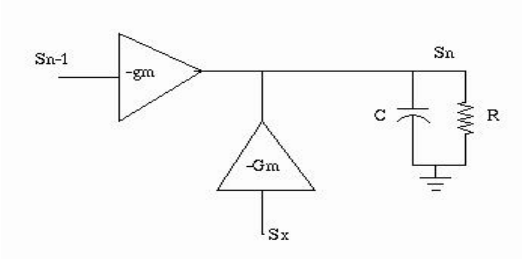


Fig-5 VCO Delay Cell

Above, Fig-5 shows the simplified circuit for the delay cell based on data anticipator circuit [4]. For the system shown in Fig-5, the transfer functions:

$$\frac{S_n}{S_{n-1}} = \frac{-g_m X_L}{1 + G_m X_L e^{-j\varphi}}$$

$$= \frac{-gmR}{(1 + GmR\cos\varphi) + j(\omega RC - GmR\sin\varphi)}$$

where φ is phase difference between S_n and S_x

X_L is load impedance at node S_n

For sustaining oscillation, Barkhausen criterion states that loop must have unity gain and phase as multiple of 2π.

$$\text{Phase, } \tan\theta = \frac{\omega RC - G_m R\sin\varphi}{1 + G_m R\cos\varphi}$$

$$\omega = \frac{\tan\theta}{RC} + \frac{Gm}{C}(\cos\varphi\tan\theta + \sin\varphi)$$

= Conventional freq. + Increment in freq. [4]

where θ is phase difference between S_n and S_{n-1}

Gm and C decide the gain in frequency, with optimal design of feed-forward inverters, higher G_m can be achieved. With multiple tapping we can increase Gm at the expense of loading capacitance.

3. DQ Data Block

3.1. Introduction

The XIO data block (DQ) provides a wide, on chip, CMOS-level signaling interface to memory controller logic (ASIC-side) and a narrow, high speed Differential Rambus Signaling Level (DRSL) interface to the external XDR memory system. The DQ block performs the serialization and de-serialization operations associated with memory write and read operations respectively. The serialization ratio of 8:1 is used. Using this octal data rate signaling, 8 serial bits are transmitted for every parallel interface clock cycle. Each DQ has 8-bit wide interface, and each bit is called DQ-slice.

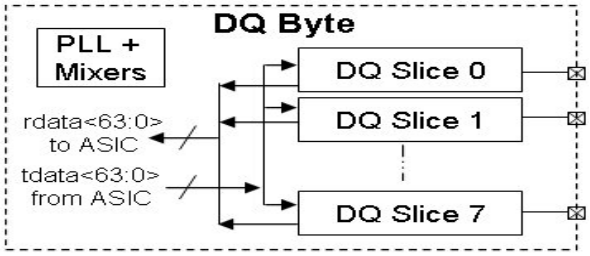


Fig-6 DQ Byte Block Diagram

Each signaling differential bit pair is calibrated in time to compensate for bit-to-bit timing differences between signals. The DQ byte and DQ slices architecture are shown in Fig-6 and Fig-7 respectively. Each DQ slice can transmit and receive 1byte of parallel data from DRAM during write and read operation respectively. Fig-7 shows serialization and de-serialization operations in DQ slice data path along with associated clock domains.

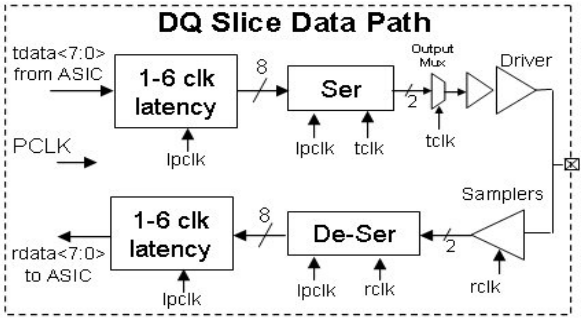


Fig 7 – DQ Slice Data Path

3.2. Clocking Architecture

The system reference clock, called CTM (Clock To Master) is derived directly from the on-board oscillator or buffered with the on-chip XDR clock generator (XCG) and the master PLL uses CTM to generate all internal fast clocks. This same CTM signal is routed back as CFM (Clock From Master) to XDR DRAM. So XIO–XDR DRAM interface works on CTM clock domain. Parallel ASIC side interface of XIO works on PCLK. For the clock domain crossing to work properly, PCLK should come from the same clock source as CTM and phase difference between CTM and PCLK should remain constant. The generation of various DQ clocks is illustrated in the fig-8 below.

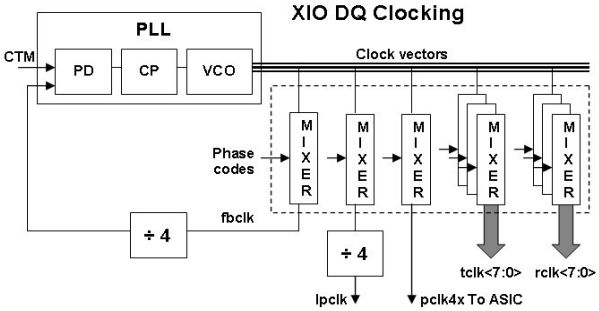


Fig-8 DQ clocks overview

XIO drives pclk4x signal to ASIC, which is generated by PLL using CTM and is four times the CTM frequency. This pclk4x is used by the ASIC to derive the PCLK after a divide-by-four circuit. To keep the phase difference between PCLK and CTM constant, a PCLK-CTM clock loop is used which locks the PCLK phase to CTM by dynamically adjusting pclk4x phase. Since the PCLK coming to different DQs and RQ from ASIC are not edge aligned to CTM and can have some skew among them due to ASIC clock tree mismatch, a

local pclk (lpclk) is derived from this ASIC pclk using another delay locked loop as shown in fig-9.

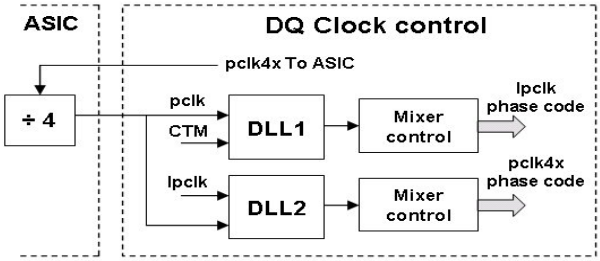


Fig-9 pclk4x and lpclk DLL circuit

Each clock within the XIO has an associated phase code indicating the phase of its edges in relation to the reference, CTM clock. The exact phase relationship between any two clocks is determined using their respective phase codes and is used to ensure reliable domain crossing between them. Due to numerous non-idealities and the high frequencies of the clocks involved, great attention has been paid for clock tree balancing and delay matching of different clock paths.

3.3. Transmitter and Receiver Front End

3.3.1. Transmit Driver and Pre-Driver. The parallel data from the ASIC core is serialized into even and odd bit streams. Both the bit streams are then multiplexed on both rising and falling edges of the transmit clock generated by the PLL. The signal levels are then translated from CMOS rail to rail swing to low swing CML levels using pre-driver circuit which is fed to the driver. Both the pre-driver and driver are implemented as current mode logic (CML) drivers as shown in Fig-10. The pre-driver works on core VddC supply (1.0V), whereas driver is on VddIO supply (1.2V). Differential current mode signaling is employed. The driver swing is maintained constant across PVT using calibrated on-die-termination(ODT).

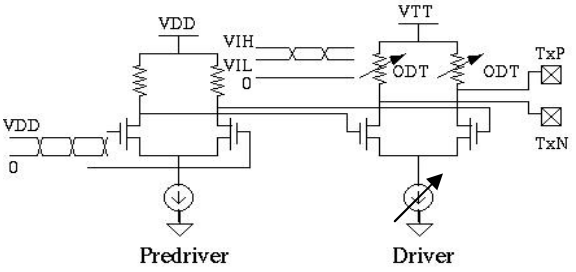


Fig-10 Transmitter front end

The driver current is continuously calibrated based on on-broad high precision resistance and voltage reference. The swing can be further adjusted by adjusting a 7-bit current control DAC.

3.3.2. Receive Sampler. The data from DRAM is received using sampler circuit. Due to noise and attenuation, the input data differential swing can be as low as 100mV with high timing jitter and it requires a comparator with high input impedance and very low offset. A voltage mode sense amplifier is used as comparator for input data. Due to its positive feedback, it achieves fast decision and full swing output is produced in a short time. Main challenges in designing sampler is meeting low offset and small dead zone of regenerative latch over process/voltage/temperature variations. As explained in [5] gm of input pair is increased by pumping more current in ON state as well as reducing transistor threshold voltage by increasing channel lengths.

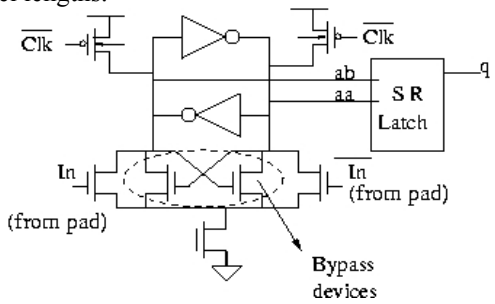


Fig-11 Receiver front end sampler

Same sampler is also used in very low frequency mode during low speed testing. During this mode, it was observed that when input is logic '0', and clock is high (sampling mode), node 'ab' [Fig-11] will be floating. This leads to charge build up on the node from neighboring leaky devices which eventually inverts logic level. To avoid this problem a cross coupled bypass devices are added in the circuit. For this case, as shown in Fig-12, output voltage doesn't change when clk is logic high and input voltage goes low.

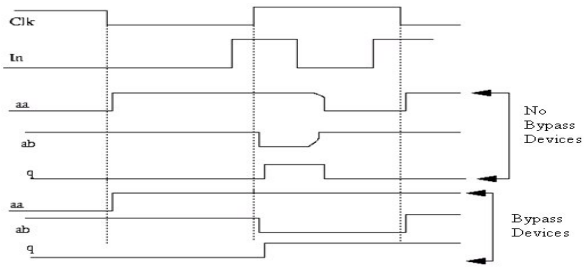


Fig-12 Sampler Timing Diagram

4. RQ Request Block

The address and command transfer to the XDR DRAM memory system is performed using RQ block. The CTL (Control) block performs register access, initialization, maintenance, and testability functions.

RQ serializes 24-bit parallel QDATA bits containing request command and 6-bit serial outputs

(handshake signals) from memory controller, to 15 bits outputs on channel. It has additional handshake signal from DRAM to total 16 bits on channel for controlling single/multiple DRAM devices. The RQ circuitry operates on the CTM frequency and performs 2:1 serialization to produce channel data rate of 2x speed and being transmitted using Rambus signaling level (RSL, 900mV). Address and command information travels synchronously from RQ to DRAMs and hence CFM has to be routed in parallel and length matched to the RQ bus to minimize skew and hence data clocking errors.

RQ block generates the operating clocks by dividing PLL clocks by four. In order to get the balanced setup-hold timing margins while sampling data on DRAM side at both clock edges, the RQ data is transmitted at quadrature (90 degrees) with respect to the CTM. To facilitate this, RQ generates a quadrature clock called QCLK which is used as sampling clock for transmission. The RQ-PCLK coming from ASIC is buffered and a local version called RQ-LPCLK is generated. Two DLLs are incorporated in RQ, one ensures the safe domain crossing from LPCLK to QCLK and other locks the transmit data phase at output pin, at quadrature to the CTM clock. The phase detector generates up/down signal which is used by mixer control logic to generate mixer phase codes such that the clocks at the input of phase detector align. This phase code information is used for reliable clock domain crossing using appropriate skip decision. All the clock forward and feedback paths of LPCLK, QCLK and RCLK are appropriately matched to ensure good timing margins in pipeline and at DRAM interface.

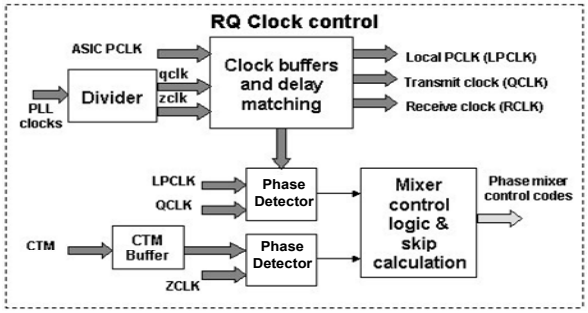


Fig-13 RQ clocking Diagram

The transmit data path comprises of the input register to capture QDATA on local PCLK, domain transfer circuit called skip logic to reliably capture data on transmit clock, the output multiplexer for 2:1 serialization of data and the output driver for sending single ended data to DRAM. The receive data path has receive sampler on both edges of receive clocks, re-timing circuit for even/odd data, domain transfer logic that skips data from RCLK to LPCLK. The RQ data-path components are shown in the Fig-14.

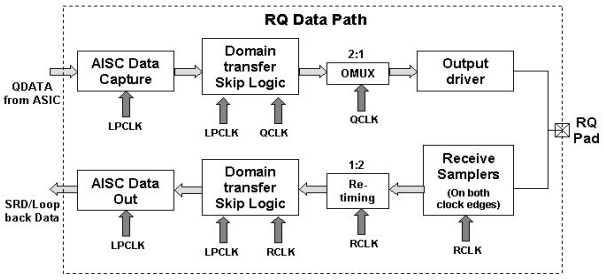


Fig-14 RQ Data path Diagram

5. Test and Scan Features

5.1. Scan

XIO is scan based design with stuck-at test coverage of 93% and at-speed-test coverage of 53%. At speed scan test feature is implemented for high-speed, timing critical nodes at actual data rates.

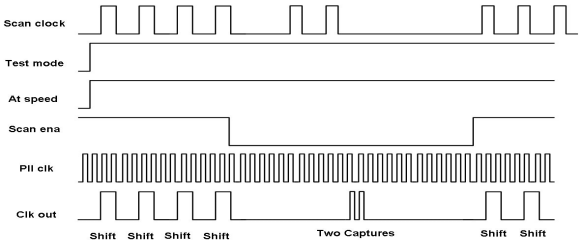


Fig-15 At-speed Scan: Shift and Capture

5.2. ASIC Side Parallel Loopback

In ASIC loop-back, the tester drives data pattern into one DQ slice and receives the pattern back on another DQ slice. The data pattern goes through the XIO internal pipelines and is driven on RDATA at the parallel interface either within XIO or inside ASIC. The tester does the comparison and determines the pass or fail of the test. This cell allows the loop back within the XIO to reduce dependency on the ASIC for test function.

5.3. Serial Loopback

Similar to parallel loop back the cell allows the serial side loop back, where ASIC sends the parallel data, loop back from the Tx of first bit to the Rx of fourth bit, and so on, and compare it at ASIC.

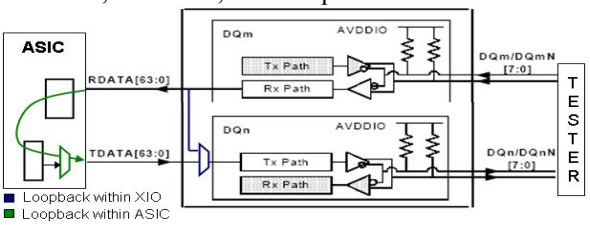


Fig-16 Parallel Loop back mode

5.4. Probing Internal Clocks

The XIO cell multiplexes the functional clocks with the data at the DQ and RQ output pins. These multiplexers are controlled by a test mode signal. When activated in test mode, various internal clocks can be probed simultaneously through different DQ and RQ pins and their mutual phase relationship can be verified.

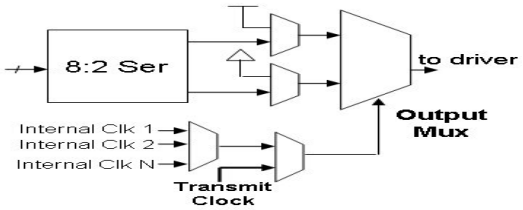


Fig-17 Internal Clock Probing Scheme

6. Silicon Characterization Results and Conclusion

The XIO cell was designed in TSMC65nmG+ process. The Memory controller was developed inside a test chip called Praveg, with basic functionality of driving the required data traffic and data comparison logic. It is connected externally on the system board with serial-interface bus. The XIO cell chip micrograph is shown Fig-18.

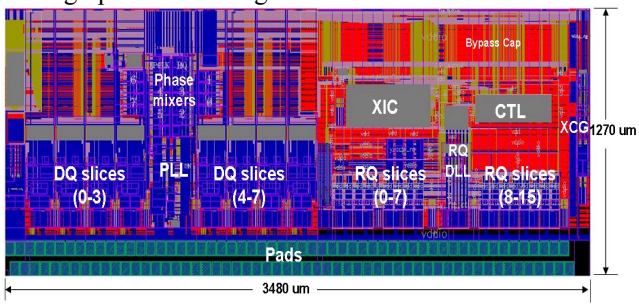


Fig-18 Chip Micrograph

The Silicon is characterized up to 4.8Gbps speeds and it meets all the required specifications. The 4.0Gbps speed operation results are summarized in Table-1. Fig-19 shows the DQ output transmit eye diagram sampled for the 4.0Gbps PRBS data pattern.

Design Information	
Technology	TSMC 65nm G+
Nominal Supply Voltage	Digital Core= 1.0V
	TX-RX I/O = 1.2V
	Analog Core(PLL etc.) = 1.0V
Temperature Range	0-130^{0}C
Area	~4.4mm^2
Output data-rate	4Gbps(nom)/2.4Gbps/3.2Gbps/4.8Gbps
Silicon Results	
DQ Tx Output Jitter	63.6ps for UI time of 250ps
DQ TX integral non linearity in phase mixer	11.3ps
DQ TX o/p Swing	200mV
UI Timing Margin	Write → 74%
	Read→69%
RQ Tx Output Jitter	156ps for UI time of 500ps
RQ TX o/p Swing	860mV
Mean Active Power	520mW
IDDQ Power	70mW

Table-1 Performance Summary

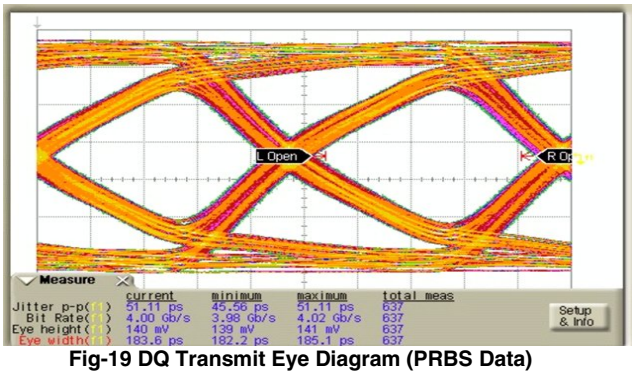


Fig-19 DQ Transmit Eye Diagram (PRBS Data)

7. References

[1] J. Maneatis, "Low-Jitter Process-Independent DLL and PLL Based on Self-Biased Techniques", IEEE Journal of Solid State Circuits, vol. 31, no. 11, Nov. 1996

[2] Kun-Yung Ken Chang et al. "A 0.4-4-Gb/s CMOS quad transceiver cell using on-chip regulated dual-loop PLLs" IEEE Journal of Solid-State Circuits, vol. 38, no. 5, May2003

[3] S. Sidiropoulos et al., "Adaptive bandwidth DLLs and PLLs using regulated CMOS buffers," in Symp. VLSI Circuits Dig. Tech. Papers, Jun. 2000, pp. 124–127.

[4] Lizhong Sun and Tadeusz A. Kwasniewski, "A 1.25-GHz 0.35-um Monolithic CMOS PLL Based on a Multiphase Ring Oscillator", IEEE Journal Of Solid-State Circuits, Vol. 36, no. 6, June 2001

[5] Wicht B., Nirschl T., Schmitt-Landsiedel D., "A yield-optimized latch-type SRAM sense amplifier" ISSCC, 2003. ESSCIRC apos, 03. Proc of 29th European Vol, Issue, 16-18 Sept. 2003 Page(s): 409-412

[6] Vijay Khawshe et.al. "A 2.5Gbps Quad CMOS Transceiver Cell Using Regulated Supply Low Jitter PLL" IEEE 20th conference on VLSI design, Bangalore, Jan 2007.

[7] Rambus XIO datasheet DL-0153 Version 0.92

Session 6B

Routing, Power Optimization

Design and Implementation of Fine-grain Power Gating with Ground Bounce Suppression

Kimiyoshi Usami[1], Toshiaki Shirai[1], Tasunori Hashida[1], Hiroki Masuda[1], Seidai Takeda[3], Mitsutaka Nakata[1], Naomi Seki[2], Hideharu Amano[2], Mitaro Namiki[4], Masashi Imai[3], Masaaki Kondo[5] and Hiroshi Nakamura[3]

[1]Shibaura Institute of Technology, [2]Keio University, [3]The University of Tokyo,
[4]Tokyo University of Agriculture and Technology, [5]The University of Electro-Communications

Abstract- **This paper describes a design and implementation methodology for fine-grain power gating. Since sleep-in and wakeup are controlled in a fine granularity in run time, shortening the transition time between the sleep and active states is strongly required. In particular, shortening the wakeup time is essential because it affects the execution time and hence does the performance. However, this requirement makes suppression of the ground-bounce more difficult. We propose a novel technique to skew the wakeup timings of fine-grain local power domains to suppress the ground bounce. Delay of buffers driving power switches is skewed in the buffer tree by selectively downsizing them. We designed a MIPS R3000 based CPU core in a 90nm CMOS technology and applied our technique to internal function units. Simulation results showed that our technique reduces the rush current to 47% over the case to turn-on the power switches simultaneously. This resulted in suppressing the ground bounce to 53mV with 3.3ns wakeup time. Simulation results from running benchmark programs showed that the total power dissipation for the function units was reduced by up to 15% at 25°C and by 62% at 100°C. Effectiveness in power savings is discussed from the viewpoint of the temperature-dependent break-even points and the consecutive idle time in the program.**

I. INTRODUCTION

As MOS transistors get scaled, leakage power dissipation has been increasing exponentially [1]. Reducing leakage becomes indispensable because leakage power becomes a major component in the total power dissipation even in the active mode. Power gating (PG) is one of the run-time techniques to switch a circuit into a low-leakage state when the circuit idleness is detected. So far, power gating control at run time has been implemented only at the IP-core level. IP cores such as CPU or DSP cores in an SoC are power gated and put into sleep depending on applications [2-4]. For example, in an SoC for cell phone applications, IP cores only used at video telephony are powered off when the operation is switched to the voice call. In contrast, more aggressive techniques to power gate internal circuits in much finer granularity have been studied and proposed in [5,6]. In [5], the authors presented a technique to power gate execution units such as a fix-point unit and a floating-point unit in a CPU. In [6], an approach has been proposed to power gate a group of combinational logic gates by employing an enable signal in a gated-clock design. These fine-grain PG techniques have more opportunities to reduce leakage at run time than coarse-grain PG techniques.

However, there are a couple of design issues in fine-grain PG that are more critical than in coarse-grain PG. One is the ground bounce induced by rush current at the wakeup. A power gating circuit using a footer power switch is depicted in Fig. 1. While the power switch (PS) is off in the sleep state, output nodes of logic gates and the virtual-ground (VGND) node are charged up to near the VDD voltage due to leakage from the supply. When PS is turned on at the wakeup, stored charge flows through PS to the ground as rush current [7]. The ground bounce induced by the rush current is proportional to $L\Delta i/\Delta t$ where L is the inductance of bonding wires and package pins. To reduce the ground bounce, reducing $\Delta i/\Delta t$ is required. This could be done by slowly turning on PS. However, in fine-grain PG the wakeup has to be done in a very short time because the wakeup penalty degrades the performance. This constraint makes the design for the ground bounce difficult in fine-grain PG. Several approaches to suppress the rush current in coarse-grain PG have been proposed so far. In [2], the authors proposed a technique to employ daisy-chained weak and strong PS transistors. At the wakeup, the weak (i.e. smaller size) PS transistors are turned on first to restore power, and then the strong (i.e. bigger size) PS transistors are turned on to deliver current for the normal operation. In [4], the authors address a technique to provide a gate-voltage sensor circuit for PS to achieve low slew-rate driving. Basically, these approaches are the technique to wakeup the entire huge IP-core where the wakeup delay in micro-sec order is permitted. In contrast, in fine-grain PG, rush current suppression by controlling independent power-switches for 10-100 local power domains is required while achieving the wakeup delay at nano-sec order. Ground-bounce analysis in fine-grain PG and a suppression technique have not been presented so far.

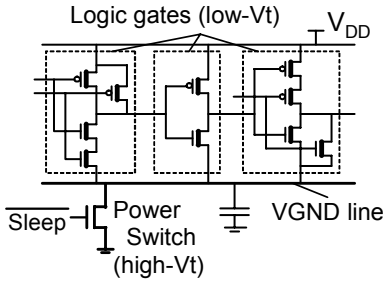


Fig. 1. Power-gating structure with footer power-switch.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.63

Another critical issue in fine-grain PG is energy overhead due to sleep-in and wakeup operations. In contrast to coarse-grain PG, the sleep period at a sleep event tends to be short in fine-grain PG. When the sleep period is too short, energy savings become less than the overhead, leading to increase of energy. In [5], the authors proposed an analytical model for the break-even point (i.e. the minimum sleep period) to get gain in energy savings and reported evaluations at the architecture level. However, evaluation for the break-even point through an actual implementation was not reported. In addition, the break-even point will change depending on environmental parameters such as temperature because sub-threshold leakage drastically varies with temperature. Evaluations based on these considerations have not been addressed.

In this paper, we present a design and implementation methodology for fine-grain PG. The primary contributions of this work are three-fold:

- Proposal for a rush-current suppression technique by delay skewing of power-switch driver trees
- Proposal for a design flow to implement fine-grain PG employing above-mentioned rush-current suppression
- Evaluation on power savings and ground-bounce suppression by applying the proposed design flow to a MIPS R3000 based CPU core.

Section II describes a proposed approach to suppress rush current and Section III presents a proposed design flow. Section IV presents the results based on the implementation.

II. Proposed Approach to Suppress Rush Current

In fine-grain PG, a power domain for a circuit is partitioned into many local power domains. First, domain partitioning is done "logically" based on enable signals. Power-gated circuits controlled by different enable signals are put into different logical domains. The logical domain is further partitioned "physically" into smaller power domains, leading to "local power domains". The physical partitioning is done with power-switch sizing such that the voltage of VGND at each local power domain may not exceed the user-specified constraint in the active state. This procedure is executed considering not only discharge current and discharge timing of each logic gate but also the conductance of PS. Power switches are driven by power-switch-driver (PSD) trees.

Power distribution network can be modeled using L, R, C of power/ground rails on the chip and those of the package interface. Considering our target clock frequency (200MHz), we use a simplified model where only R, C on the chip and L, R, C of the package interface are taken into account.

Turning on all power switches simultaneously may induce huge rush current. To suppress rush current, we propose a technique to skew the delay of PSD in the tree to avoid the simultaneous turn-on of power switches. There are two possible approaches to accomplish this: one is to unbalance the number of stages in the tree by inserting additional drivers. This approach, however, increases area and power. More critical drawback is an impact on the layout design because inserting a new driver changes the connectivity. This will affect design convergence if P&R is already finished.

The other approach is to skew the delay by sizing the PSD. In particular, delay skewing by downsizing the drivers minimizes the impact on the layout because this does not increase area or does not change connectivity. Meanwhile, the wakeup delay may be increased by downsizing the PSDs. Hence, the problem to be solved can be stated as follows: For a given fine-grain power-gating circuit, we minimize the ground bounce by downsizing PSDs to skew the delay such that the wakeup delay is within the specified value. To effectively avoid the simultaneous turning-on of power-switches, we focus on downsizing the leaf drivers in the tree.

III. Design Flow

In this section, we describe a design flow to implement the proposed approach to suppress the ground bounce in fine-grain PG. First, a layout style for the fine-grain PG is presented. We use a Locally-Shared Virtual-ground (LSV) scheme presented in [6]. The structure for LSV scheme is depicted in Fig. 2. This structure enables us to build different local power domains in a row. In addition, cells in a local power domain can be placed using two or more rows. Furthermore, power gated cells can share a row with non-power-gated cells such as PSDs, flip-flops, clock buffers, and isolation cells. This flexibility becomes a strong point when we implement fine-grain PG.

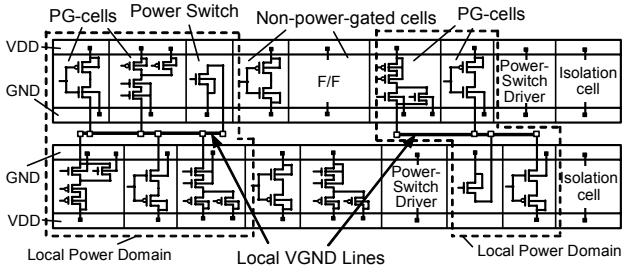


Fig. 2. Locally-Shared Virtual ground (LSV) structure.

In the LSV scheme, a modified cell called a power-gating cell (PG-cell) is used. The PG-cell is a low-Vt logic cell with the same functionality as the original standard cell but has a VGND pin. The source of an nMOS transistor in the PG-cell is connected to a VGND pin instead of the real ground rail. In addition, a power switch is defined as a cell with a VGND pin and an enable pin. A VGND line is wired in each local power domain at the routing such that VGND pins of PG-cells and those of power-switch cells are connected.

Next, we describe the proposed design flow. From RTL, we synthesize a netlist using low-Vt standard cell library. Then we partition the design into logical domains based on enable signals. By swapping the logic cell in the synthesized netlist with the same-sized PG-cell, we generate a netlist with logical domains. After the initial placement, we perform physical partitioning into "local power domains" and insertion of properly sized power-switches as described in the previous section. Inserting isolation cells to avoid signal floating is also

done at this step. Next we build PSD trees and optimize the drivers to suppress the ground bounce. PSD trees are constructed such that the output slew rate of each driver be identical to each other. By using this structure, we downsize PSDs to skew the delay in the PSD tree. The result of this step is sent to a router and the final layout is completed.

IV. Evaluation and Results

A. Setup for Experiment

In order to evaluate effectiveness of the proposed approach, we designed a MIPS R3000 based CPU core, referred to as Geyser, by using the proposed design flow. Fine-grain PG was applied to internal function units such as ALU, shifter, multiplier and divider, and also to a coprocessor unit CP0 for interrupt processing. Only the unit executed in the current instruction is powered on and the rest is powered off. Wakeup operation does not occur at two or more units at the same time. Sleep and wakeup control is done by a sleep controller unit. In this CPU, a pre-wakeup technique was employed to reduce performance penalty due to the wakeup delay. The sleep controller detects a function unit to be woken up by pre-decoding an instruction [9] and fully wakes up the unit before the execution.

B. Results of Implementation

Table 1 summarizes the results from applying the proposed implementation scheme to the function units and the coprocessor unit. The number of local power domains varies from 29 to 170. ALU, shifter, divider and the coprocessor unit are controlled by their own unique enable signal, resulting in one logical domain. In contrast, the multiplier has two logical domains to dynamically control the power gating for the upper and lower halves of the array of carry-save adders.

Table 1. Implementation results of fine-grain power gating for function units

	ALU	Shifter	Multiplier	Divider	CP0
Area W[μm]xH[μm]	90x85	80x85	241x246	241x226	151x 146
# Logic Cells	1218	643	6405	6995	1633
# Local Power Domains	63	29	170	112	80

The layout result is shown in Fig. 3. The size of the CPU core is 540μm×570μm. The total area of the function units and the coprocessor unit (CP0) to which we applied the fine-grain PG occupies 49% in the entire core, while the area of the sleep controller shares 5%.

C. Rush Current and Ground Bounce

We evaluated effectiveness of the proposed approach through analyses on rush current and the ground bounce at the power-gated unit. Results are presented only for the unit with the largest cell count, the divider, due to the limit of space. The divider unit consists of 6995 logic cells and the total transistor width (Wp+Wn) of the cells is 29497μm.

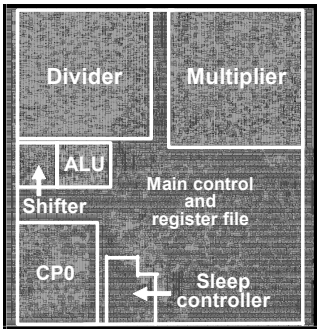


Fig. 3. Layout of CPU core Geyser in which four function-units and coprocessor unit are power-gated in fine-grained manner.

Power-switch sizing was done with physical partitioning into local power domains such that the VGND voltage may not exceed 100mV in the active mode. This process was done by using a commercial tool [10]. As a result, the divider unit was partitioned into 112 local power domains and the total transistor width of power switches is 845μm. For those optimized power switches, a PSD tree was constructed such that the output slew rate of each PSD be 300ps. We refer to this PSD tree as a "typical buffer tree". PSD directly driving the power switch is referred to as a leaf driver. A leaf driver may drive one power switch or may drive two or more power switches.

Under keeping the connectivity in the typical buffer tree, we experimented with three options for delay-skewing (Opt.1-3). In Opt.1, 8 of 39 leaf-drivers were downsized from X4 to X1 size. In contrast, in Opt.2 and Opt.3, 13 and 15 leaf-drivers were downsized, respectively. By these downsizings the maximum output slew rate at leaf drivers was changed to 500ps. As a comparison, we also experimented with an option to turn on all power switches exactly at the same time. In this option we assumed that the gate of power switches is directly driven by a signal with the slew rate of 500ps. This slew rate is identical to the maximum value of the output slew rate of PS in Opt.1-3.

To look at the effect of our delay-skewing, we investigated the voltage waveforms for the gate-input signals of power switches. By using the parasitic from the layout, we conducted SPICE simulations at V_{DD}=1V. Results are shown in Fig. 4. To quantify the spread of the distribution, we focus on the time at which the gate-input signal of a power switch rises. We defined this as the arrival time (AT) of the enable signal to the power switch. Hence, AT is defined as the delay from the time at which the root enable signal rises to the time at which the gate-input signal of a power switch rises. We measured this at the time at which the voltage waveform crosses $V_{DD}/2$. In the typical buffer tree, the difference between the earliest AT and the latest AT is around 685ps. In contrast, this difference is increased to 791ps, 1.19ns and 1.05ns in Opt.1-3, respectively, because of delay-skewing for the PSD tree.

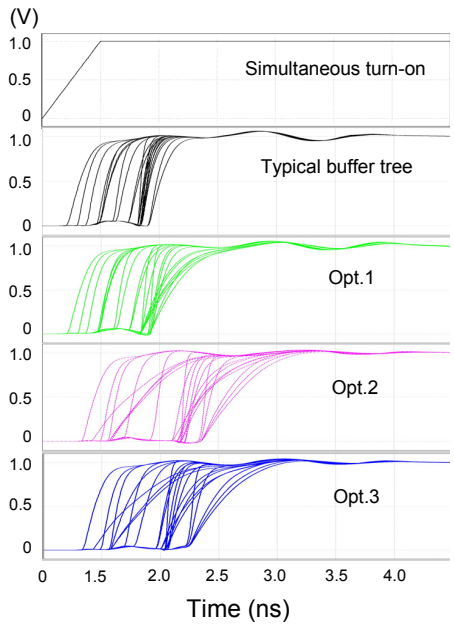


Fig. 4. Simulated voltage waveforms for the gate-input signals of power switches.

Next we demonstrate how the delay-skewing affects rush current and the ground bounce. Fig. 5 shows simulated waveforms for (a) rush current without L, R, C of the package interface, (b) the ground bounce and (c) rush current with above-described L, R, C. For electrical parameters of the package interface, we used L=2nH, R=70mΩ and C=0.2pF for the TQFP package [8]. Additionally, taking into account the actual assembly of the chip, we assumed that four bonding wires are connected to the ground pad of the chip in parallel. R and C of the on-chip power/ground rails were extracted from the layout by using a commercial tool [11].

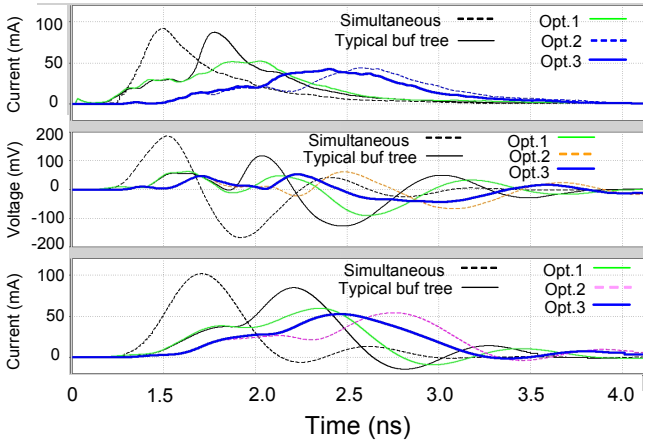


Fig. 5. Simulated waveforms for (a) rush current without L, R, C of the package interface (top), (b) the ground bounce (middle) and (c) rush current with L, R, C of the package interface (bottom).

Rush current induces the ground bounce and the ground bounce affects back the rush current, resulting in the rush-current waveform shown in Fig.5(c). To understand the original cause of the ground bounce, it is meaningful to observe rush current without L, R, C of the package interface. As shown in Fig.5(a), rush current waveforms for the "simultaneous turn-on of power switches" and the "typical buffer tree" show higher and sharper peaks than those for Opt.1-3. In particular, in Opt.2 and Opt.3, the height of the peak (i.e. the maximum rush current) is reduced to 48% and 47% over the simultaneous turn-on, respectively. As a result, the maximum voltage of the ground bounce is reduced, as demonstrated in Fig.5(b). The simultaneous turn-on induces the ground bounce of as much as 187mV, whereas Opt.1-3 reduces the peak voltage down to 53-61mV.

To investigate how the distribution of turn-on timings of power switches influences the ground bounce, we analyzed the standard deviation σ of the arrival times described above and plotted against the maximum values of rush current and the ground bounce. Results are shown in Fig. 6. As σ increases, the maximum values of rush current and the ground bounce get reduced until a certain point. It is interesting to observe that increasing σ beyond that point increases the rush current and ground bounce instead.

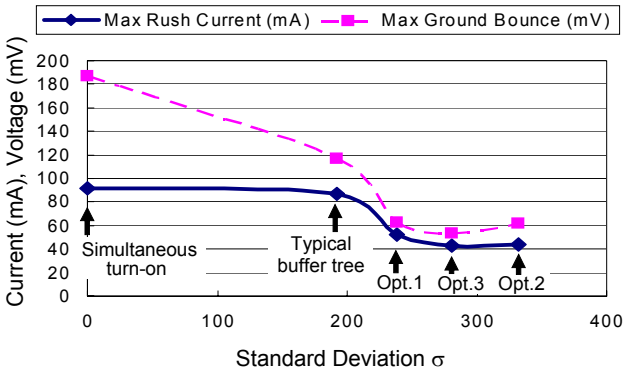


Fig. 6. Maximum values of rush current and ground bounce as a function of standard deviation σ of arrival times.

To understand these phenomena, let us compare the waveforms shown in Fig. 5 (a) and (b) for Opt.2 and 3. Compared to the rush-current waveform for Opt.3, the waveform for Opt.2 is broader but it consists of two peaks. This means that between the two peaks the occurrence of turning on the power switches got less. In contrast, in Opt.3 we downsized two more PSDs such that their arrival times come into the interval between the two peaks. Thus, the scheme aiming at increasing σ of arrival times is basically a good strategy but requires a fine-tuning as well to adjust them such that the arrival times are distributed as equally as possible.

Wakeup time is plotted in Fig. 7. The wakeup time is defined as the delay from the time at which the root enable signal rises to the time at which the VGND voltage reduces to 0.02V.

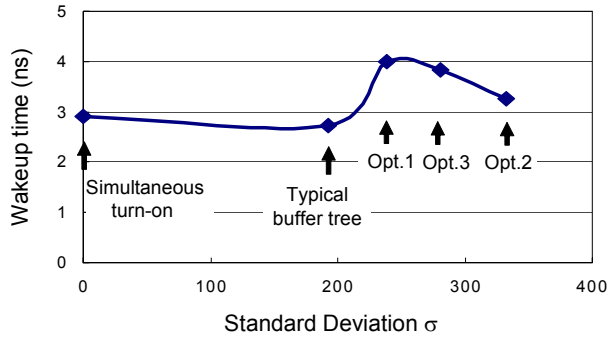


Fig. 7. Wakeup time as a function of standard deviation σ of arrival times.

As σ increases, the wakeup time increases up to a certain point but decreases beyond that point. In particular, in Opt.2 and 3 the wakeup times are 3.8ns and 3.3ns, respectively, which are smaller than that of Opt.1. This is because discharging current through the power switch at the wakeup is affected by the ground bounce and the influence in Opt.2 and 3 gets less due to the reduced ground bounce.

D. Power Dissipation

Based on extracted parasitic from the layout of the Geyser CPU core, we conducted power analysis. PowerCompiler and Hsim were exploited to analyze dynamic and leakage powers. We used two benchmark programs: (i) a quick sort program (QSORT) in MiBench [12] and (ii) a JPEG encode program (JPEG_E). To precisely reflect the influence of cache-miss, we created an RTL model for 8KB 2-way set-associative instruction/data caches. This model was combined with the RTL model of the CPU core and used at the simulation.

We implemented non-power-gated counterpart to the Geyser CPU core and compared power dissipations. Fig.8(a) and Fig.8(b) show the results from power analysis for QSORT and JPEG_E programs, respectively.

For QSORT, the total power dissipation for the five power-gated units was reduced by 9% at 25°C and by 62% at 100°C, respectively. Power overhead due to introducing the sleep controller is counted in these numbers. Instruction pre-decoding and detection of a function unit to be used are performed at the sleep controller. Power dissipation of power-switch drivers is included in that of power-gated units. For JPEG_E, power was reduced by 15% at 25°C and by 60% at 100°C, respectively. It should be noted that power dissipations for the multiplier and divider are reduced to 1/10-1/20 at 100°C. In contrast, power dissipations for ALU and shifter are not reduced remarkably. We investigated this reason through analyses for the break-even point and consecutive sleep cycles.

The break-even point for the sleep cycles to get gain in energy savings was investigated using transistor-level simulations. We analyzed energy dissipation E_{PG} for each power-gated unit while changing the sleep cycles. For non-power-gated counterparts, we evaluated energy dissipation E_{NPG} as well.

The ratio E_{PG}/E_{NPG} is plotted against the sleep cycles in Fig.9(a) and Fig.9(b) assuming the cycle time of 5ns. The sleep cycles at which the ratio E_{PG}/E_{NPG} crosses 1.0 is the Break-Even Cycles (BEC).

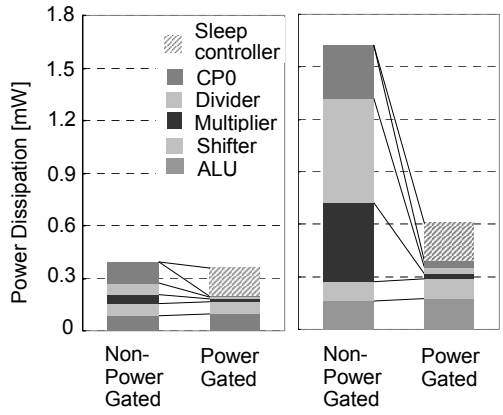


Fig. 8 (a). Power reduction by power gating for QSORT program at 25°C (Left) and at 100°C (Right).

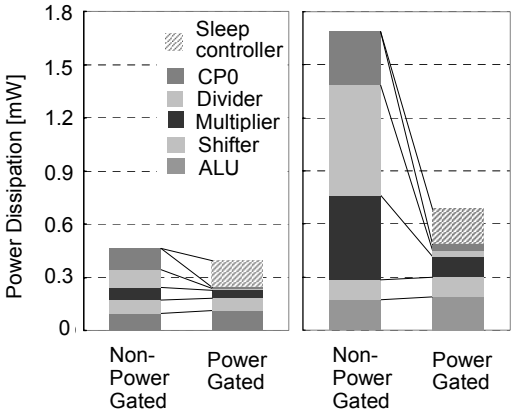


Fig. 8 (b). Power reduction by power gating for JPEG_E program at 25°C (Left) and at 100°C (Right).

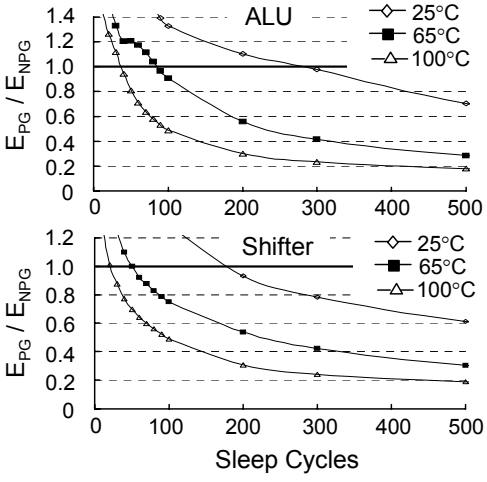


Fig. 9 (a). Energy ratio as functions of sleep cycles for ALU (Upper) and Shifter (Lower). Cycle time of 5ns is assumed.

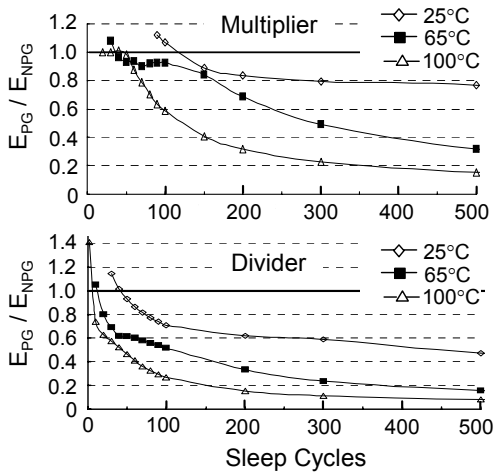


Fig. 9 (b). Energy ratio as functions of sleep cycles for Multiplier (Upper) and Divider (Lower). Cycle time of 5ns is assumed.

Fig.10 shows BEC as functions of temperature. It should be noted that BEC reduces with increasing the temperature. This is because sub-threshold leakage increases with the temperature, thereby enabling power-off at the short sleep to gain. At 100°C, BEC for ALU and shifter are 38 cycles and 22 cycles, respectively.

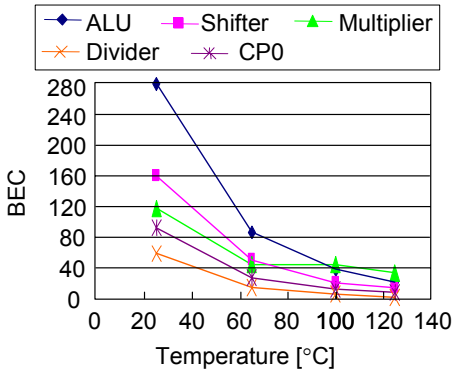


Fig. 10. Break-even cycles (BEC) as functions of temperature. Cycle time of 5ns is assumed.

Investigation showed that sleep events whose consecutive sleep times are over BEC for these units share only 2% of the total execution time. Powering off these units at shorter sleep cycles than BEC increases energy rather than saving it. In contrast, for the multiplier, BEC at 100°C is 44 cycles. Sleep events whose consecutive sleep time is over BEC share 74% of the total execution time. Moreover, for the divider, this number increases to 99%. Thus, for the multiplier and divider, majority of the sleep events is the sleep whose sleep cycles is over BEC. This results in reducing power effectively at these units.

IV. Conclusion

We presented a design and implementation methodology for fine-grain power gating. In the fine-grain power gating, the issue of the ground bounce induced at the wakeup is critical because a short wakeup time is required. We proposed a novel technique to employ delay-skewing in the power-switch driver tree to suppress rush current. Through a design of a CPU core, we evaluated rush current, the ground bounce and power dissipation. Simulation results demonstrated effectiveness of the proposed methodology. As the future work, we extend our delay-skewing approach to combine with a technique to unbalance the number of stages in the tree.

Acknowledgments

The authors thank VLSI Design and Education Center (VDEC), Synopsys, Cadence, Mentor, Sequence Design, STARC, and Japan Science and Technology Agency (JST) CREST for their support.

References

[1] D. Lackey, P. Zuchowski, J. Koehl, "Designing mega-ASICs in nanogate technologies," in Proc. DAC'03, 2003.
[2] P. Royannez, et. al., "90nm low leakage SoC design techniques for wireless applications", ISSCC, 2005.
[3] T. Lueftner, et. al., "A 90nm CMOS low-power GSM/EDGE multimedia-enhanced baseband processor with 380-MHz ARM926 core and mixed-signal extensions", ISSCC, 2006.
[4] T. Hattori, et. al., "Hierarchical power distribution and power management scheme for a single chip mobile processor", Proc. of ACM/IEEE Design Automation Conference, pp.292-295, 2006.
[5] Z. Hu, et al, "Microarchitectural techniques for power gating of execution units," Proc. ISLPED'04, pp.32-37, 2004.
[6] K. Usami and N. Ohkubo, "A design approach for fine-grained run-time power gating using locally extracted sleep signals," Proc. ICCD'06, pp.155-161, Oct. 2006.
[7] S. Kim, et al, "Understanding and minimizing ground bounce during mode transition of power gating structure", Proc. ISLPED'03, pp.22-25, 2003.
[8] P. Heydari and M. Pedram, "Ground bounce in digital VLSI circuits", IEEE Trans. VLSI Systems, vol.11, no.2, pp.180-193, April 2003.
[9] N. Seki, et al, "A Fine-grain Dynamic Sleep Control Scheme in MIPS R3000", Proc. ICCD'08, Oct. 2008.
[10] *CoolPower* by Sequence Design, Inc., www.sequencedesign.com.
[11] *Calibre* by Mentor Graphics, http://www.mentor.com/
[12] M. Guthaus, et al, "MiBench: A free, commercially representative embedded benchmark suite," Proc. IEEE 4th Annual Workshop on Workload Characterization, pp. 3-14, 2001.

A method for the Multi-net Multi-pin Routing Problem with Layer Assignment

Tuhina Samanta
Bengal Engg. & Sc. University, India
t_samanta@it.becs.ac.in

Prasun Ghosal
Bengal Engg. & Sc. University, India
p_ghosal@it.becs.ac.in

Hafizur Rahaman
Bengal Engg. & Sc. University, India
rahaman_h@it.becs.ac.in

Parthasarathi Dasgupta
Indian Institute of Management Calcutta, India
partha@iimcal.ac.in

Abstract

Interconnects are vital in deep sub-micron VLSI design, as they impose constraints, such as delay, congestion, cross-talk, power dissipation and others, and consume resources. These parameters affect the efforts for obtaining a feasible solution for the global routing of multiple nets. In addition, efforts are on for exploration and use of non-Manhattan routing architectures. In this work, we focus on the specific problem of multi-net multi-pin global Y-routing for custom-built design styles with several available routing layers. The problem is formulated as a minimum crossing Y-Steiner Minimal tree problem with multi-layer assignment. Experimental results are quite encouraging.

1 Introduction

The VLSI layout problem is usually solved in a hierarchical framework. Each stage of the hierarchy is optimized, while the problem becomes manageable for the subsequent stages. Global routing is a critical phase of this hierarchical design flow, particularly, the physical design flow. It assigns wires and vias to signal nets so as to obtain approximate interconnections of the pins of every net. Objectives in this phase include chip size and wire-length minimization, even congestion distribution, ensuring signal integrity, cost and ease of fabrication, time to market, and so on. Global routing is known to be a very difficult problem. In fact, finding a feasible routing of a two-pin net in the presence of congestion has been shown to be NP-Complete [1].

Traditional global routing architectures are Manhattan. With increasing dominance of interconnects, their is increasing research on and use of X- and Y-architectures. X-routing is now well appreciated in chip manufacturing circle; however, the research community has been recently investigating the Y -architecture [2, 3]. This refers to wiring with 0°, 60°, and 120° oriented wires for on-chip interconnects. Several benefits of Y-architectures vis-a-vis X-architectures have been elucidated in [3]. Routing of Y-interconnects for multi-net multi-pin nets is thus an interesting problem to be explored.

In this paper, we focus on the multi-net global routing over a number of wiring layers, for Y-interconnects such that the total length of all the routing trees is minimum. Since each crossing of two edges will yield the insertion of a via, yielding obstacles to routing, and increasing delays, we attempt to minimize the number of crossings of the trees as well.

The rest of the paper is organized as follows. Section 2 introduces some significant related works and the motivation of our work. Section 3 describes the formulation of the problem. Section 4 describes the proposed method of construction of Y-routing trees with minimum edge-crossings in a given number of layers. Section 5 describes the empirical observations , and Section 6 concludes the paper and discusses its possible extensions.

2 Literature survey

Global routing is a well-researched problem. [8], [7] and others present updated coverage of progress in global routing. [5] and [6] focus on global routing issues for a single net. The survey on multi-net global routing [4] discusses the recent global routing methods with emphasis on performance-driven multi-net routing. A recent report on academic global routing [14] provides a brief review of some of the best high-performance routing techniques of recent times.

[1]This work is supported by grants from the Department of IT, Govt. of India, New Delhi, Projects: SMDP-II and R & D in Microelectronics

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.30

The existing global routing algorithms can be broadly classified into the following categories: maze routing, Steiner tree construction, and 0-1 ILP [8]. The maze routing algorithm finds a shortest path connecting two pins in the presence of wiring blockages. A variant of the maze routing techniques is the group of line probe-based algorithms. It is important to note that Maze routing inherently can consider one net at a time. Thus, an extension to multi-net domain requires nets to be considered one at a time, in a particular order.

Maze routing and line probe algorithms are, however, applicable to only two-pin nets. In practice, routing problems consider nets with more than two pins. The wire length of a routing tree, in such cases, is usually reduced by constructing a Steiner Minimal tree [7]. Traditional VLSI routing problems use only rectilinear Steiner trees. However, recently, the use of wire geometries with other orientations are quite predominant. Many of the Steiner tree construction algorithms in literature focus on the optimization of a single net, and do not consider wire congestion issues explicitly. Nevertheless, these algorithms can be applied to serially route the nets, with the most critical nets being routed before non-critical nets. When the edge cost is defined according to congestion, the Steiner minimum tree algorithms may be applied directly to even out congestion while simultaneously restraining the wire length [9].

Global routing may be formulated as a special type of optimization problem, called a zero-one integer linear programming (0-1 ILP) problem. For a set of candidate routing trees $T_i = T_{i,1}, T_{i,2}, \ldots$ for net N_i, we use variable $x_{i,j}$ to indicate if tree $T_{i,j}$ is selected for net N_i.

Very recently, there has been a growing interest in the construction of obstacle avoiding Steiner trees [16, 15]. The work reported in [19], which has fair relevance to our work, proposes an algorithm for simultaneous escape routing within a set of components on a printed-circuit boards (PCB) that ensures minimum crossings of the nets in the subsequent area routing phase.

The use of diagonal wires was exploited on PCB and integrated circuits for more than a decade [12]. Y-routing and Y-architecture for integrated circuits were introduced through series of works of two different groups in [2] and [3]. The work in [3] gives an in-depth analysis of Y-architecture, and highlights the potential advantages of the use of Y-architectures vis-a-vis the X-architectures. Algorithms for construction of Y-routing trees appear in [11, 10].

2.1 Motivation of the work

To the best of our knowledge, not much work has been reported in the area of multi-net multi-pin routing, especially with the use of non-Manhattan interconnects. As such, there are several unexplored issues and unanswered questions in this area. In this work, we attempt to provide an answer the following question:

Given a set of pins of several different nets, and a set of wiring layers, can we obtain a global Y-routing of all these nets using these layers, such that the number of crossings of the routing trees of the different nets is as small as possible?

The following Section formulates the problem, and introduces the basic framework for the proposed method.

3 Problem formulation

In global routing, a signal net consisting of a set of fixed terminals $N_i = \{n_{i,0}, n_{i,1}, \ldots, n_{i,k_i}\}$ is connected by a routing tree $T(N_i)$. The cost of the tree $T(N_i)$ is the sum of costs of its constituent edges. Thus, the total cost $\chi(T(N_i))$ of a tree $T(N_i)$ of k_i nodes is given by $\chi(T(N_i)) = \Sigma_{j=1}^{p} t(e_j)$, where $t(e_j)$ is the cost of edge e_j in routing tree $T(N_i)$. In the multi-net multi-pin global routing problem, we consider a number κ, say, of signal nets, each net containing a number of pins = 2 to k_i.

In this case, we consider a fixed set of wiring layers Λ, say, for the placement of the routing trees, such that each layer can have at least one Steiner Minimum Tree (SMT) assigned to it. In order to ensure minimum routing resource consumption, the total length of the routing trees is to be minimum. The SMTs considered are Y-routing trees.

The Minimum length minimum edge-crossing Multi-net Multi-pin global routing ($MNMP$) problem can be formulated as follows:

Given a number κ, say, of multi-pin nets Π_j, j = 1 to κ, a set of wiring layers Λ, and an upper-bound on the layout area, construct a Y-routed Steiner Minimum Tree for every net Π_j, and assign the trees appropriately to the given layers such that (i) the total length of the trees is minimum, and (ii) total number of edge crossings of the Steiner trees is as small as possible.

4 Proposed method

In this paper, our approach broadly comprises the following steps:

- For every net, construct the Convex Hull of its constituent terminals.
- Construct a matrix, each element of which contains the amount of overlapping regions of a pair of nets. Sort the nets in decreasing order of their total amount of overlaps with other nets.
- Assign the minimum overlapping nets to the given set of wiring layers in an increasing order of the amount of overlap.

- If the number of wiring layers required is more than the available number of layers, redistribute the excess nets appropriately among the available wiring layers.
- Construct the Y-routed SMTs and modify them using a set of operators to minimize the number of edge crossings between all pairs of trees in every layer.

Let T_i, T_j respectively represent the Steiner Minimal Trees of the i^{th} and j^{th} nets, and CH_i, CH_j be the corresponding convex hulls. Then, the following observations are clear.

Observation 1 *Overlapping of CH_i and CH_j does not necessarily imply crossing of an edge of T_i with an edge of T_j.*

Observation 2 *If CH_i and CH_j do not overlap, then there cannot be any crossing of edges of T_i and T_j.*

From the set of pins for all the κ nets, find the coordinates of the leftmost-bottommost pin position, and the rightmost-topmost pin position. Let (x_{min}, y_{min}) and (x_{max}, y_{max}) respectively denote these coordinates. Then, the available routing area is bounded by a rectangular box within these set of coordinates.

We discuss the above steps below. However, the complete details could not be provided due to paucity of space.

4.1 Convex Hull creation

Consider a given net to be a set of points in a plane, where each point corresponds to a terminal of the net. A convex hull CH_j, j = 1 to κ is generated for each net Π_j (i.e., its associated points) using Graham's scan algorithm [17]. If Q_j is the number of points lying over the boundary of CH_j, then $1 \leq Q_j \leq p_j$, where p_j is total number of pins for the net Π_j.

Definition 1 *Two convex hulls CH_i and CH_j are said to overlap with each other if and only if a vertex on the boundary of one convex hull is bounded by the other convex hull.*

4.2 Layer assignment of nets

Once the convex hulls have been constructed, the amount of overlap of $\binom{\kappa}{2}$ possible pairs is obtained. Since a convex hull is a convex polygon, the following observation is clear.

Observation 3 *Intersection of a pair of convex hulls form a convex polygon.*

Thus, the amount of overlapping regions of a pair of convex hulls is obtained from the area of the polygon formed by their intersection. Figure 1 illustrates a pair of intersecting convex hulls with a convex polygon (a, b, c, d) created by their intersection. The circular nodes are the terminals, and the rectangular nodes are intersections of the boundaries of the convex hulls.

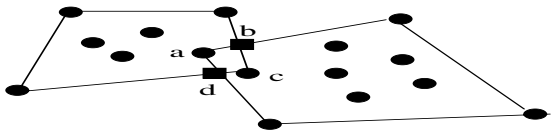


Figure 1. Overlapping Convex Hulls

Next, the convex hulls are arranged on the number of vertices along their boundaries, and are assigned to the layers in that order. Consider the layer numbers to be 1 to $\mid \Lambda \mid$ from bottom to top. The largest (in terms of the number of vertices along boundary) convex hull is the first to be assigned to layer 1. Next, the set of convex hulls, if any, which do not intersect this initially assigned convex hull, and are mutually non-intersecting with each other, are assigned to layer 1. Once the assignment to layer 1 is complete, layer 2 is the next to be considered. The same procedure is applied to layer 2 as well. The iterative assignment completes when either (i) all the nets have been considered, or (ii) all the layers have been considered, and some nets are yet to be assigned. In the second case, when some nets are yet to be assigned, for each of the leftover net, its most potential layer assignment is determined by finding the total overlapping area of this net with the already assigned nets for every layer.

Definition 2 *A wiring-layer Λ_j is considered to be most potential for assignment of a net Π_i if the total amount of overlap of the convex hull of net Π_i with the convex hulls of the nets already assigned to Λ_j is minimum among all the layers.*

For a net Π_i, once its most potential layer is determined, Π_i is assigned to this layer.

4.3 Construction of Y-Routing Steiner Minimum Tree

The Y-Routing Steiner Minimum Tree is constructed using a heuristic method. The proposed method depends on the computation of shortest paths between pairs of nodes in an underlying Hanan grid graph. Initially, a pair of terminals in the given problem that are farthest apart in Euclidean space, is obtained. It attempts to find the shortest path between these terminals in the underlying routing graph. An iterative sequence of steps is executed thereafter. At each iteration, the following steps are executed in sequence:

1. A partial SMT denoted by $G_{partial}$, is constructed.

2. For each of the remaining terminals, find the Euclidean distance of the terminal from its nearest Steiner node in $G_{partial}$.

3. Find the maximum of all the Euclidean distances obtained in the previous Step. Let p be the corresponding terminal, and q be the corresponding Steiner node in $G_{partial}$.

4. Find the shortest path between p and q in the underlying routing graph; augment $G_{partial}$ with this shortest path.

The iteration stops when all the terminals have been considered, and $G_{partial}$ is reported as the Y-routing Steiner tree. Figure 2 illustrates a partially grown Y-routing tree in an underlying Hanan grid for a net.

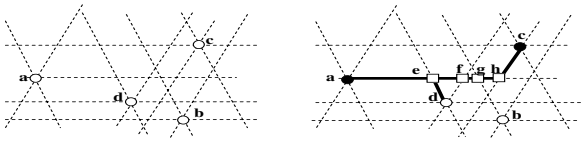


Figure 2. Example of Y-Routing SMT

We refer the readers to [10] for the details of the algorithm.

4.4 Transforming Steiner trees

Once the Steiner trees are constructed in the different layers, the pairs of Steiner trees with intersecting edges are identified. A set of operators [18] are applied on these Steiner trees for possible transformation to a different set of Steiner trees, such that the number of intersections is reduced. We apply the following set of operators:

- Operator $M1$: Flipping an edge of the Steiner tree (see Figure 3).
- Operator $M2$: Inserting detours (see Figure 4).
- Operator $M3$ Sliding an edge of the Steiner tree comprising two Steiner vertices.
- Operator $M4$: Moving a single Steiner node along an edge (see Figure 5).

Figure 3. Application of operator M1

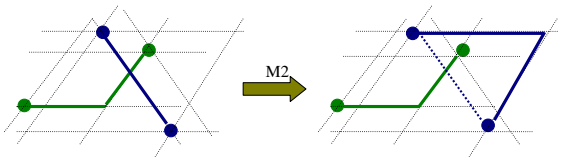


Figure 4. Application of operator M2

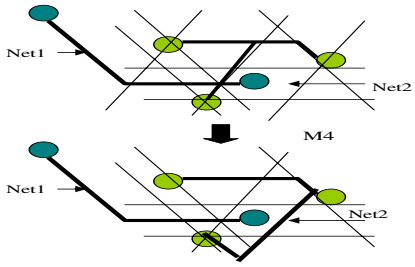


Figure 5. Application of operator M4

In Figures 3, 4 and 5, the figures on left and right respectively illustrate Steiner Minimal Trees before and after application of the operators. Consider Figures 3 and 4. The dashed lines illustrate the respective edges that are removed after application of the operators "edge flip", and "detour". Detailed explanation of all the operators could not be given due to paucity of space.

4.5 The algorithm

The overall algorithm for solving the $MNMP$ problem for a fixed number of layers is as described in Figure 6.

4.6 Time complexity

As mentioned above, let κ be the total number of nets, and N_i be the number of pins in i^{th} net, where i = 1 to κ. The number of layers is denoted by $\mid \Lambda \mid$. Let N = total number of pins for all the nets. Then worst-case time complexity of the proposed algorithm (Figure 6) is given by Lemma 1.

Lemma 1 *The time complexity of* $MNMP$ *is* $O(\mid \Lambda \mid \times\kappa^2)$ + $O(\kappa \times N^3 \log N)$, *where* N *is the total number of pins (terminals) of all the nets.*

Proof. The proposed algorithm (Figure 6) is divided into a number of phases. Time complexity of constructing the convex hull of the i^{th} net using Graham's scan is $O(\mid N_i \mid \log \mid N_i \mid)$, where N_i is the set of pins/terminals of the i^{th} net. Time complexity of finding the amount of overlap of convex hulls for a pair of nets, say, i^{th} and j^{th} nets, is $O((\mid N_i \mid + \mid N_j \mid)\log(\mid N_i \mid + \mid N_j \mid) + k\log(\mid N_i \mid$

Algorithm for the $MNMP$ problem

Input: a set of terminals $\mathcal{P}$ for a number of nets;A set Λ of wiring layers
Output: Minimum-intersecting $Y - routed$ Steiner Minimal Trees for the nets assigned to Λ.

1. $layout\_area$ = compute_layout_area()
(* compute the layout area from the pin coordinates *)
2. (* Convex hull construction for nets *)
3. for i = 1 to κ $C_i = Generate\_Convex\_Hull(\Pi_i)$
4. for i = 1 to κ - 1
5. for $j = i + 1$ to κ
6. Find_CH_Intersect(C_i, C_j)
(* Generate a matrix of overlaps for all pairs of nets *)
7. Sort the Convex Hulls (C_i, i = 1 to κ) in descending order of their number of vertices
8. (* assigning convex hulls to the layers *)
9. l = 1 (* layer index *)
10. c = 1 (* index of convex hull / net *)
11. while (not all nets assigned) and (not all layers considered) do
12. assign convex hull CH_c to layer l
13. $Area_l$ = area of CH_c; (* initialize consumed area *)
14. while ($\exists$ net p not intersecting any of the nets assigned to layer l) and (p is not already assigned)
15. if $temparea\_l$ (= $Area_l$ + area of CH_p) $\leq layout\_area$
16. assign net p to layer l; $Area_l$ = $Area_l$ + area of CH_p
17. $l = l + 1$
18. if (all nets assigned) and (not all layers considered)
19. Total number of layers = l; return
20. if (not all nets assigned) and (all layers considered)
21. (* distribute the remaining nets over the layers *)
22. for each remaining convex hull CH_c
23. find the most potential layer l_{min} (* min overlap *)
24. assign CH_c to l_{min}
25. Generate_MST();(* construct the Y-routing trees *)
26. for l = 1 to total number of layers
27. transform MSTs in layer l to minimize their mutual intersections

Figure 6. Algorithm for the MNMP problem

$+ \mid N_j \mid$)), where k = number of vertices of the polygon of intersection [17]. Then, the total time complexity for all the $\binom{\kappa}{2}$ pairs of convex hulls is $\Sigma_{i=1}^{\kappa-1}\Sigma_{j=i+1}^{\kappa}(\mid N_i \mid + \mid N_j \mid) \log(\mid N_i \mid + \mid N_j \mid) + k \log(\mid N_i \mid + \mid N_j \mid) \leq N^2 \log N$, hence $O(N^2 \log N)$. Time complexity of sorting the nets in decreasing order of the number of constituent vertices is $O(\kappa \log \kappa)$.

Assigning non-overlapping nets to $\mid \Lambda \mid$ layers requires checking one net per layer against all the other nets. Since we store the information on overlapping nets in a 2-dimensional symmetric matrix, then the time complexity for this part is $O(\mid \Lambda \mid \times \kappa^2)$. In the worst-case, this should also be the time complexity of finding the most potential layer for a net.

Time complexity of constructing a Y-routed Steiner Minimal Tree for the i^{th} net is $O(N_i^3 \log N_i)$ [10]. Total time complexity for constructing Steiner trees for all the nets is then $O(\kappa \times N^3 \log N)$. Finally, the worst-case time complexity of applying transformation operators on all the Steiner trees is $O(N^3)$.

Thus, the overall worst-case time complexity of the proposed algorithm for the $MNMP$ problem is roughly $O(N^2 \log N) + O(\mid \Lambda \mid \times \kappa^2) + O(\kappa \times N^3 \log N) + O(N^3) + O(\kappa \log \kappa) \simeq O(\mid \Lambda \mid \times \kappa^2) + O(\kappa \times N^3 \log N)$, for large N.

5 Experimental Results

The proposed algorithm is implemented in C on a P-IV machine at 2.5 GHz running on Linux. A set of benchmark instances available at [20] are used as inputs for the proposed algorithm. For construction of the grids, and computation of the layout area on each layer, we consider the technology node parameters for 65nm from [21]. The grid granularity is varied from bottom to top layers as per convention. The outcomes of the experiments are summarized in Table 1. Area shown in the table is in grid units for 65nm. The results indicate that our method can handle larger number of nets compared to [19] in much less CPU time. Figure 7, for a particular number of layers, confirms our expression of time complexity as a function of the number of nets. Figure 8 illustrates bar charts for different problem instances showing the overlap counts of nets before and after layer assignment.

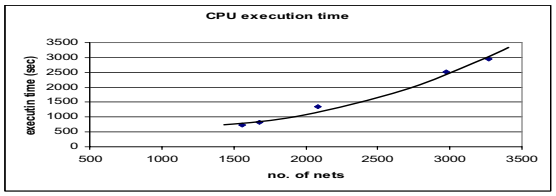


Figure 7. CPU time vs. Number of nets

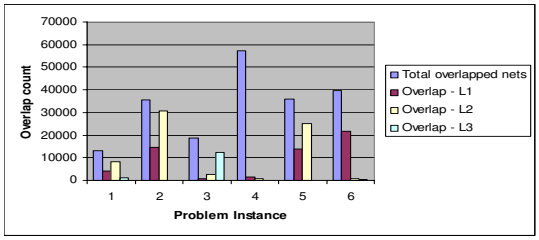


Figure 8. Overlap reductions of nets

6 Conclusion and Future Scope

In this work, we consider the problem layer assignment of multi-pin multi-nets with minimum edge crossings.

The proposed algorithm and the observations are very useful for the following reasons: (i) the $MNMP$ problem is of immense importance in VLSI physical design flow, (ii) the proposed algorithm is comprehensive, handles some critical issues related to multi-net routing, and handles a large number of nets in reasonable CPU time, (iii) to the best of our knowledge, not much has been reported in literature so far on this problem, and (iv) the $MNMP$ problem

Problem instance	# of nets	Layout area	# of layers used	CPU time (secs.)	Layer details		
					layer #	# nets placed	Area used
bench1	1557	120395000.0	3	730.01	1	1360	120394997.5
					2	184	120394218.5
					3	13	16723865.0
bench2	1676	121682961.0	4	824.39	1	811	121682960.5
					2	241	121682958.5
					3	605	121649628.5
					4	19	107261782.5
bench3	2691	121110000.0	2	2782	1	1579	121109999.0
					2	1112	77015672.5
bench4	3271	121010998.0	2	2954	1	3014	121010991.0
					2	257	55570606.0
bench5	2086	119289984.0	5	1349.14	1	1244	119289977.0
					2	148	119289980.5
					3	584	119289980.5
					4	72	119125470.5
					5	38	173328775.0
bench6	2973	120758121.0	2	2509.16	1	2814	120758111.5
					2	159	13179302.0

Table 1. Summary of results for some ISPD-2007 benchmark instances ($adaptec1.capo70.2d.35.50.90$)

for Y-interconnects does not seem to have been tackled in the existing literature.

The present work has wide scope of extension: (i) further minimization of crossings of the Steiner trees, (ii) certainly, the consideration of vias between layers with via minimization. Studying the variation of the extent of SMT edge-crossing minimization with order of the net assignments would be another interesting extension.

References

[1] M R Kramer and J van Leeuwen, "The complexity of wire routing and finding minimum area layouts for arbitrary VLSI circuits". In F P Preparata (Ed.), *Advances in Computing Research*, volume 2: VLSI theory, pp. 129-146, JAI, Reading, MA, 1984.

[2] M. D. Rostoker et al., *Hexagonal architecture*. U.S. Patent, No. US6407434B1, June 2002.

[3] A. B. Kahng, I. Mandoiu, Q. Wang, H. Chen, Chung-Kuan Cheng and Bo Yao, "The Y-architecture for on-chip interconnect: Analysis and methodology", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2005.

[4] J. Hu and Sachin S. Sapatnekar, "A Survey on Multi-net Global Routing for Integrated Circuits", *Integration, the VLSI Journal*, Vol. 31, pp. 1 - 49, November 2001.

[5] A. B. Kahng and G. Robins. On optimal interconnections for VLSI. Kluwer Academic Pub., Boston, MA, 1995.

[6] J. Cong, L. He, C.-K. Koh, and P. H. Madden, Performance optimization of VLSI interconnect layout, *Integration: the VLSI Journal*, Vol. 21, pp. 1-94, 1996.

[7] M. Sarrafzadeh and C. K. Wong, "An introduction to VLSI physical design", McGraw-Hill, New York, NY, 1996.

[8] N. A. Sherwani, "Algorithms for VLSI physical design automation". Kluwer Academic Pub., Norwell, MA, 3rd edition, 1999.

[9] C. Chiang, C. K. Wong, and M. Sarrafzadeh, A weighted Steiner tree-based global router with simultaneous length and density minimization, *IEEE Transactions on Computer-Aided Design*, Vol 13, No. 12, pp. 1461-1469, December 1994.

[10] T. Samanta, P. Ghosal, H. Rahaman and P. Dasgupta, A heuristic method for constructing hexagonal Steiner minimal trees for routing in VLSI, *Proc. of IEEE International Symposium on Circuits & Systems*, 2006.

[11] A. Thurber and G. Xue, Computing Hexagonal Steiner Trees using PCS, *IEEE International Conference on Electronics, Circuits & Systems*, pp. 381-384, 1999.

[12] E Lodi, "Routing multiterminal nets in a diagonal model", *Proceedings of the Conference on Information Sciences and Systems*, pp. 899 902. Dept of EE, Princeton University, 1988.

[13] J. F. Weng M. Brazil, D. A. Thomas and M. Zachariasen, "Canonical forms and algorithms for steiner trees in uniform orientation metrics", *Algorithmica*, Technical Report, pp. 2 22, 2002.

[14] M. D. Moffitt, J. A. Roy and Igor L. Markov, "The Coming of Age of (Academic) Global Routing", *Proceedings of the International Symposium on Physical Design (ISPD)*, pp. 148-155, April 2008.

[15] Chung-Wei Lin, Shih-Lun Huang, Kai-Chi Hsu, Meng-Xiang Li, and Yao-Wen Chang, "Efficient Multi-Layer Obstacle-Avoiding Rectilinear Steiner Tree Construction", *Proceedings of the International Symposium on Physical Design (ISPD)*, April 2008.

[16] Z. Feng, Y. Hu, T. Jing, X. Hong, X. Hu, and G. Yan, An O(nlogn) algorithm for obstacle-avoiding routing tree construction in the lambda-geometry plane, *Proceedings of the International Symposium on Physical Design (ISPD)*, pp. 48-55, April 2006.

[17] M. de Berg, O. Cheong, M. van Kreveld, M. Overmars, "Computer Geometry: Algorithms and Applications". Springer-Verlag, 2008.

[18] E. Bozorgzadeh, R. Kastner and M. Sarrafzadeh, "Creating and exploiting flexibility in Steiner trees", *Proc. of Design Automation Conference*, pp. 195-198, 2001.

[19] M M Ozdal and M D F Wong, "Simultaneous escape routing and layer assignment for dense PCBs", *Proc. of the International Conference on Computer-aided Design (ICCAD)*, pp. 822-829, 2004.

[20] http://www.sigda.org/ispd2007/contest/

[21] http://www.eas.asu.edu/p̃tm/

A New Hardware Routing Accelerator for Multi-terminal Nets

Mrs. Kaleem Fatima
Muffakham Jah College of Engg. and Tech.
Osmania University-Hyd

Dr. Rameshwar Rao
University College of Engineering (A)
Osmania University-Hyd

Abstract

This paper presents a new parallel processing wire routing machine, which finds a quasi-minimum Steiner tree for multi-point connections in a VLSI chip. A hardware implementation with concurrent time-multiplexed wavefront propagation from all terminals of a net is described. The new design requires fewer clock cycles to find the shortest path than the existing parallel routing algorithms. The time-multiplexed mode optimizes the number of interconnections. An RTL implementation has been developed in VHDL and the algorithm has been successfully tested for a prototype 4 × 4 and 8 × 8 single layer grid on an FPGA. The feasibility of the algorithm for larger size grid and nets with higher degree is demonstrated.

1. Introduction

Increasing system complexity is driving the solution time of design automation (DA) problems particularly, the routing phase to unacceptable levels. Since advances in CAD provide the capability for larger circuits and better design techniques, some combination of better algorithms and/or better machines is required, if design times are to remain reasonable. Considerable improvements have been made on many DA algorithms for conventional sequential machines; however, some problems like maze routing, have resisted significant run-time improvements. For some of these problems, the development of new hardware is a potential solution.

Several hardware accelerators that speed up the routing process have been proposed [3-8]. Nestor [7] re-demonstrated the feasibility of building hardware routers using ASIC, custom and FPGA technology. The full grid machines of [3,7] achieve linear run time (*O(n)*) for finding a path between two points. But no consideration is given to multi-point nets. These machines solve the routing of a multi-point net problem, by decomposing it into two-point sub problems. These sub-problems are solved sequentially, as the path found previously is used as the starting point. Depending on the order in which points are routed, this procedure sometimes results in redundantly lengthy paths as shown in Fig 1a (stared line). The Parallel Adaptable Routing algorithm (PAR-2) [5] is a popular hardware algorithm, which finds a quasi-minimum Steiner tree for multi-point nets. PAR-2 was implemented on a parallel SIMD processor, AAP-1.

Many software solutions exist for finding OARSMT (Obstacle avoiding RSMT). These have non-linear time complexity (i.e. $O(n^2)$ for n x n grid). The best known software solutions, for example [10] have tried to achieve a time complexity of $O(n \log n)$. The design of a hardware maze router with concurrent source/target wave propagation is described in [8]. Separate data lines are used for source and target wave propagations. This increases the number of interconnections by 33% for two terminal nets. And if this concept were extended to more than two terminals, the interconnection space would increase beyond practical limits. Hence for multi-point nets, the algorithm of [8] is again similar to a conventional hardware maze router and does not guarantee an optimal solution.

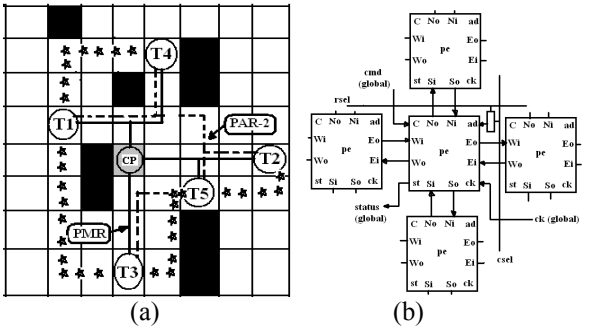


Figure 1. (a) Solution for a multi-point net routing - PMR path (solid line), PAR-2 (dashed line) and conventional router (stared line). Figure 1b. **The architecture of the processing element.**

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.44

This paper presents a new parallel processing algorithm called *PMR (Parallel Multi-point net Routing algorithm)*. The algorithm finds a quasi-minimum Steiner tree for multi-point connections. A hardware implementation with concurrent time-multiplexed wavefront propagation from all terminals of a net is described. The new design requires fewer clock cycles to find the shortest path between the terminals than the existing parallel routing algorithms (e.g. PAR-2). The time-multiplexed mode allows communication between adjacent processing elements to be carried on a single line, thus optimizing the number of interconnections as compared to [8]. The path so found is essentially independent of the order of connecting terminals and is optimal. To our knowledge, this enhancement has not been adopted by any of the hardware maze routers proposed in the literature. The performance of our algorithm is compared with PAR-2 [5] as these machines have *O(n)* time complexity. We show that our algorithm is faster by 18.75% compared to PAR-2 for three terminal nets and the quality of path obtained is same for both algorithms. Fig. 1a shows the results for the three type of routers: PMR (solid line), PAR-2 (dashed line), and conventional router (stared line). The length of the path for conventional router is 19 (non-optimal) and PMR and PAR-2 are 15(optimal). And number of corners for PMR is 4, conventional router is 5 and PAR-2 is 6. Performance results for PMR and PAR-2 are given in section 4.

A VHDL code is written and implemented to demonstrate the feasibility of the algorithm on a prototype 4×4 and 8×8 single layer grids. The algorithm has been successfully tested on a XC2VP30 Virtex-II Pro XUP development board. The algorithm can be easily extended to multi-layer grids.

This paper is organized as follows: Section 2 describes the operation of the PMR algorithm. A hardware description of the PMR accelerator and its operation is given in section 3. Section 3 also describes the architecture and operation of the PE. Section 4 gives results and Section 5 gives the summary and conclusion.

2. The PMR Algorithm

The PMR algorithm is based on the basic Lee's algorithm. The Lee algorithm [1] finds the shortest path between two points in a grid even in the presence of obstacles. It does so in three phases: *wavefront expansion*, *backtracking and clean up*. The software solution to the above problem has $O(n^2)$ time complexity for a *n x n* grid. But by parallel processing on a two-dimensional array processor, the wavefront expansion and backtracking are performed in linear time and clean-up is usually a one step operation. Thus in a hardware accelerator, the time complexity can be reduced from a square to a linear. As the maze router is known to be iterative in nature, the ordering of nets, and the selection of candidate path for rip-up is to be handled by the host PC. The hardware router helps in very fast rip-up and re-routing of nets.

2.1 Validity of the new algorithm

To prove the validity of the PMR algorithm, a modified Lee table called PMR-Lee table is shown in the Figure 2 for a three point net. The actual algorithm is described in section 2.2. Each cell in the PMR-Lee table has as many fields, as the number of points in the net. Thus for a three terminal net each cell has three fields, say F1, F2 and F3. In the first phase a "Center Point" (CP) corresponding to the minimum distance from all the three terminals is found. For this, let each terminal start wavefront expansion concurrently. Terminal T1 labels its adjacent cells in the field F1 only, terminal T2 in F2 and terminal T3 in F3 fields respectively. Let T1 stop its expansion phase once it hits the other two terminals -- T2 and T3. Terminal T2 stops expanding when it hits T1 and T3 and so does terminal T3. Now the resultant figure with cells being labeled with all the three terminals only need be considered to locate the CP, as shown in the Fig. 2. The field '4' at the right hand top corner of each cell indicates the sum of all the three labels (or fields). The cell in the 3rd row and 6th column (circled cell) has a minimum sum, of 10. This cell is the "Minimum *Center Point" (MCP) or simply CP (Center Point)*, and the values in its fields indicate that the cell is at a distance of 5 grid points from T1, 1 grid point from T2 and 4 grid points from T3. The minimum path for the three terminal-net is determined through this point as shown in the Figure 2.

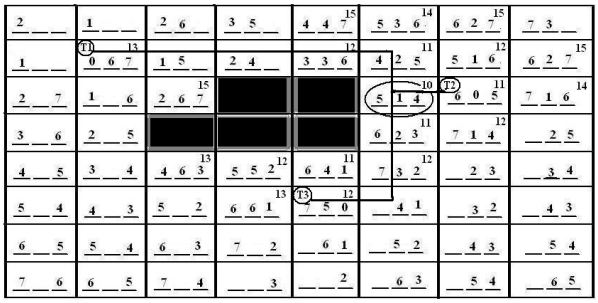


Figure 2. The modified Lee table (PMR-Lee Table) showing parallel labeling, The circled cell is the minmum Steiner point or the Center Point (P).

2.2 The new Parallel Multi-point net Routing algorithm (PMR algorithm)

A special-purpose application-specific SIMD processor is designed and developed that implements this algorithm in hardware (section 3). The architecture is an n x n array of identical cells called Processing Elements (PE). Each PE is a finite state machine and is directly mapped to a grid point during routing. This architecture is similar to [3, 7]. An entirely new PE and the Control Unit were designed to accommodate the algorithm. The details of implementation are given in section 3. This algorithm may also be implemented on other architectures as in [4, 6] to reduce the number of PEs, and extended to multi-layers. The PMR is implemented in two phases as described below.

2.2.1 Center Point search phase: For the sake of illustration we assume a 4 x 4 grid and a three terminal net. In the first phase the algorithm finds the CP. (Refer Figure 3. Initially, there is no target and all the terminals (T1, T2 and T3) will be the source terminals. The terminals T1, T2 and T3 start wavefront expansion concurrently. As each cell communicates with its orthogonal neighbors on a single line, the wavefront expansions are multiplexed in time in a round robin fashion as shown in the Fig. 3. First T1 expands (Fig. 3b). Its adjacent cells are labeled in the field F1. These labeled cells do not expand instantly. They do so in the next round for T1. T2 expands after T1 and its adjacent cells are labeled in the field F2 as in Fig. 3c, and then T3 expands (Fig. 3d). In the next round of expansion, cells labeled as a result of T1, expand into the next wavefront as in Fig. 3e. This process is repeated for the other terminals and is indicated in the Fig. 3(f) through Fig. 3(h). Now, during the expansion phase each cell may have been labeled once (due to T1, T2 or T3 in any one of the three fields), or may have been labeled twice (due to any two terminals in any two fields). The wave expansion phase stops as soon as a cell is labeled in all the three fields. This cell is the CP for the 3-point net (Fig. 3h, - circled cell). This point becomes the Source 'S' for the second phase, which is the *path determination phase*. In this phase path shown in Fig. 3h is found.

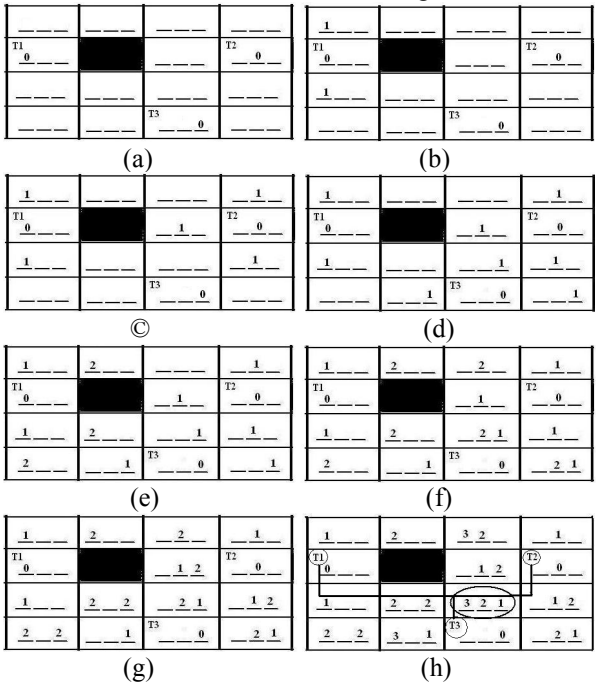


Figure 3. (a -- h) shows phase I- multiplexed expansion mode for the three terminal net.

2.2.2 Second Phase--Route through the CP: Fig. 4, shows the operation of phase-II of the PMR. As the backtrace is a linear operation, a path is iteratively determined from CP to all the terminals in question. In this phase all the terminals will be set as target terminals, *T,* and the CP will be the Source terminal *S* as in Fig. 4a. From this stage, the router operates as any conventional hardware maze router. Firstly, a path is determined between the CP and the nearest terminal, as in Fig. 4(b) and Fig 4(c). Then, all the points on this path are set as sources and the unused cells are cleared as in Fig. 4d. A second labeling starts from all the cells in the path as in Fig. 4e. The second target, which is the next nearer one, is hit by the expanding wavefront, Fig. 4e. During traceback, the source point in the earlier path, nearest to the target in question is hit, (in the above example it is CP), as seen in Fig. 4f. This point could be the Steiner minimal point (RSMP) itself, or any other point for optimal route. This procedure is repeated till all the terminals are routed, Fig. 4(g) through Fig. 4(j). In case phase I results in two (or more) CPs, which result in two (or more) different

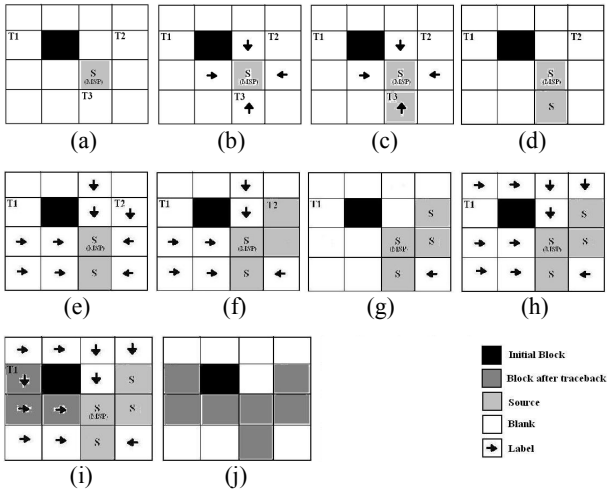


Figure 4. (a – j) Shows a few important stages of phase 2--the path determination phase.

shortest paths, then during traceback phase depending on the choice of heuristic only one CP will be chosen as a part of the path and the rest will be cleared.

3. Hardware implementation of the PMR algorithm:

The basic architecture of the PMR router and its operation is described briefly in this section. The hardware router (Fig. 5) is composed of a processor, an I/O unit and a control Unit. The processor unit consists of $n \times n$ array of identical cells, called Processing Elements (PE) (Fig. 1b). The I/O unit consists of encoders and decoders. Each PE is directly mapped to a grid point in a grid array.

The Control Unit (CU) can address any PE (for local operations) through the row and column decoders and identifies a cell using addr_out lines of the row and column encoders. In our implementation the CU sends 14 commands to the processor (RST, IDLE, BLOCK, SET_T1, SET_T2, SET_T3, LABEL_1, LABEL_2, LABEL_3, LABEL, TRACE_BACK, CLEAR, INT, RIP), some of which are local and others are global command. The processor sends its status to the CU via the status lines 'St'. The CU also communicates with the host PC using commands like RESET, BEGIN, E_BL (enter block cells), E_S_T (enter source terminals), PATH_FAIL, RIP_UP, PATH_FOUND, and END.

3.1 The Processing Element

The (i,j)th cell in the processor array and its four adjacent neighbors are shown in Fig. 1b. Each PE is a FSM and has 8 states, Bl (block), B (Blank), T1, T2 and T3 (terminal states), L (Label state), S (Source or CP) and the T(Target) state. Each PE (Fig. 1b) has four control inputs, Cmd = C (0:3) common to all cells which receive the command from the CU, and are internally decoded to decide its next course of action (next state). Each cell sends its status to the CU via the Status lines (St_0 and St_1 which are wire-OR'ed to a common bus going to all cells). The ClK and RST input runs common to all cells. Each PE has two address input lines 'rsel' and 'csel'. The CU selects any PE via these lines to set it to Block (BL), Source (S), Target (T) or Terminal (T1, T2, and T3) states as the case may be. Each PE can communicate with its neighboring PE in SIMD mode whenever the CU sends any LABEL or TRACE_BACK commands using the four unidirectional Input ($N_i + W_i + E_i + S_i$) and Output (N_o, W_o, E_o, S_o) lines. Each PE has three flip-flops, F1, F2 and F3, which store the information of the three

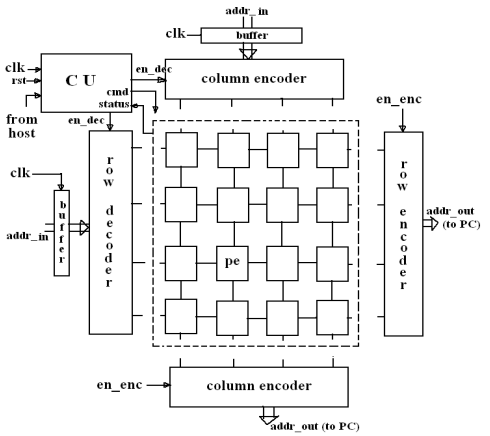


Figure 5. The architecture of PMR accelerator.

fields of our PMR-Lee table (Fig.2).

3.2 Hardware operation

The control unit governs the operation of the entire system. On reset, all the cells are set to BLANK state in SIMD mode. Next, all the Block cells are read sequentially followed by the Source terminals (T1, T2, T2) into the processor (Fig 3a). This is followed by Phase I, which is the CP search phase. The control unit sends three label commands Label_1 (for T1 expansion), Label_2 (for T2 expansion) and Label_3 (for T3 expansion), in three subsequent clock cycles. This is repeated in a round robin fashion. At each clock period, a blank cell adjacent to a labeled cell will be labeled if X ='1', depending on the Label command as described in section 2. A cell entering the expansion state is indicated to the control unit by asserting the status signal 'St_0' (see Table 1) connected to the global wired-OR status bus.

Labeling stops when the CU receives a HIGH on the Status signal St_1 from the processor unit. The signal St_1 is asserted as soon as a cell is labeled in all the three fields F1, F2 and F3. This indicates to the CU that the CP cell is found and Phase I is completed successfully. Table I shows the state table for PE in phase- I. There is no traceback in phase I. Table 1 describes the cell state sequencing for the PE in phase I. In Table1, PS is the present_state of PE, NS is the next_state. Columns 3 and 4 of table 1 gives the values of nx_state, St_0St_1 (status), F1F2F3 (field values indicating whether a cell is labeled as a result of expansion due to T1, T2 or T3) and '-' indicates a don't-care value. The signals X and Y in the Table I have the following Boolean relation: $X = N_i + W_i + E_i + S_i$ and $Y=N_o=W_o=E_o=S_o$. If X='1', it means the cell has received a signal from its neighbor on any one or more inputs. If Y='1', the cell is sending an expand

signal to its neighbor. In path determination phase, all the three terminals will assume the target state T, although Fig 4 retains the earlier names (i. e. T1, T2 and T3) for clarity only. Upon the next command from the CU, the source S (or CP) starts labeling its orthogonal neighbors in parallel SIMD mode using Akers method [2] till it reaches the nearest target. In this phase traceback direction is recorded. A shortest path between the S and the nearest terminal is determined. The remaining steps are as explained in Section 2.2.2. and illustrated in Fig. 4.

In phase II, the PE assumes the following state: S (source or CP), T (Target), L (Label), B (Blank) and Bl (Block). Labeling process is similar to [3] and is not described here. Table 2 describes the cell state sequencing in phase II during the traceback process. From Table 2, if signal 'Pc = 0', the cells in the traceback path will re-enter the 'S' state to start the labeling again as in Fig. 4e. Only if Pc=1 the cells in the traceback path gets blocked as in Fig. 4 (i) and Figure (j). St_0 is the status signal. Signals G_{in} and G_{out} are '1' if a cell is selected to be a part of the path. The phase II expansion takes (F1 + F2 + F3) clock cycles. Same number of clock cycles is required by the traceback phase. All unused cells are cleared in one clock cycle in parallel SIMD mode. The cell clearing has to be done three times for a three terminal net in phase II. The data loading initially is $O(N_{Bl} + N_T)$, where N_{Bl} is the number of block cells and N_T is the

Table 1. State table for PE during phase I. (For columns 3 and 4, values denote: nx_state, S0 S1, F1 F2 F3, for X='0' And X='1')

Command (Cmd)	PS, F1 F2 F3	X='0'	X='1'
Label_1	B, 000	B, 00, 000*	L, 10, 100*
	T1, 000	T1, 10, 100	-
	L, 0 - -	L, 00, 0 - -	L, 10, 1 - -
	Bl, 000	Bl, 000	Bl, 00, 000
Label_2	B, 000	B, 00, 000	L, 10, 010
	T2, 000	T2, 10, 010	-
	L, - 0 -	L, 00, - 0 -	L, 10, -1-
	Bl, 000	Bl, 000	Bl, 000
Label_3	B, 000	B, 00, 000	L, 10, 001
	T3, 000	T3, 10, 001	-
	L, - - 0	L, 00, - - 0	L, 10, - -1
	Bl, 000	Bl, 000	Bl, 000
Any Label	L, 111	CP, 01, 111	CP, 01, 111
S_{int}	Outputs: $N_o = W_o = E_o = S_o$ = '0'		

Table 2. State table for 'PE' during traceback in phase -II

Present State	Gin=0		Gin=1	
	Pc=0	*Pc=1*	*Pc=0*	*Pc=1*
B	B, 00	B, 00	-- --	-- --
L	L, 00	L, 00	S, 10	Bl, 10
Bl	Bl, 00	Bl,00	Bl, 00	Bl, 00
T	S, 10	Bl, 10	S, 10	Bl, 10
S	S, 00	S, 00	S, 11	S, 11

* Nextstate, G_{out} St_0

number of terminals. In the worst case this will be close to $O(n^2)$ for *n x n* grid. But in practice it is much less.

For a large net with say 200 terminals and 1,10,000 length, assuming 25MHz clock or 40ns time period, an estimated time of 0.37sec would be required by PMR. The storage elements will also increase, as 200 memory elements will be required in each PE. Assuming 4000 x 4000 grid an estimated 3.2GB of memory would be required. The size and complexity of the controller would appropriately swell, as it has to multiplex between 200 terminals.

Hence a more feasible solution would be to divide the problem of size 200 terminals into clusters 5 or 10 closely spaced terminals and iteratively route each cluster by the above method. A proper grouping of terminals is necessary. This would improve the speed and reduce the hardware. While the reduction in hardware is understandable, the speed also increases because the wavefronts for a 10 terminal net are much smaller than a 200 terminal net. Also the multiplexing will be within 10 terminals. After determining the route for all the clusters, they can be joined by again going for a CP in between the clusters and then joining them.

4. Results

The design of the multi-point PMR accelerator has been coded in VHDL and synthesized using the Xilinx ISE 8.2i targeting a Xilinx XC2VP30 Virtex-II Pro FPGA, on a XUPV2P Virtex-II Pro development board. The code has been simulated using Active VHDL 6.2. Both simulation and synthesis were performed with a 40ns timing constraint. Table 3 shows the device utilization summary for 4×4 and 8×8 grid sizes. The array unit includes the I/O unit.

Run Time Performance: From our simulation results, for PMR, Center point is found in (n*d) clock cycles where 'n' is the number of terminals and 'd' is the numerical value of CP. Maximum number of clock cycles to determine the path after the CP is found is 2(F1+F2+F3) for expansion and traceback together, where F1, F2, F3 are the distances of the terminals T1, T2, and T3 from the CP. Time for CP determination is,

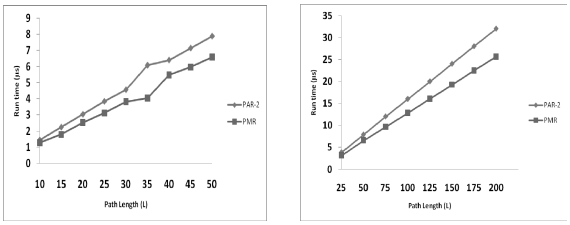
Tp1 = n d * 40ns and for path determination phase is Tp2 = 2(F1+F2+F3) * 40ns. Therefore total time Tp (PMR) = Tp1 +Tp2 = nd + 2(F1+F2+F3) * 40ns. If F1+F2+F3 is the Net-length (worst case), L, then Tp = nd + 2L. For the case indicated in the Fig. (3) and Fig. (4), for 25MHz clock the time taken is ((3*5)+2(5+1+4)) *40ns=35 * 40ns=1.4µsec.

We have theoretically evaluated the performance of PMR and PAR-2 on 60 test cases of three pin net circuits having average net length of 17 (net lengths ranging from 9 to 41) on a grid size of 10 x10. The obstacles (block cells) were uniformly distributed. The algorithms were evaluated with 0%, 10%, upto 50% block cells. The experiments were repeated for 15 x 15 and 20 x 20 grid sizes (nets of larger length) and a few cases of more than three terminal nets. We found that the mean of L/d =2.47 for three pin nets, for PMR. Thus for three pin nets, Tp = 3 * L/2.47 + 2L = 3.21 * L. The mean of L/d by our statistics is nearly independent of the length of the net for three pin nets. For terminal more than three, the value L/d was found to increase in PMR. Performance of PAR-2 was simultaneously evaluated for the above cases.

By our evaluation, PAR2 takes 2(F1+F2+F3) clock cycles, both for phase I and phase II. Assuming same clock of 40ns, PAR-2 takes 4*(F1+F2+F3) clock cycles. Thus Tp (PAR-2) = 4L * 40ns. By our statistical observations, PMR algorithm is faster than PAR-2 by 18.75%, as is indicated in the figure 6. The PMR is also observed to be faster than PAR2 for terminals of degree less than 5. For more than 5 terminals we do not have performance measurement at this point of time. But PMR can be very efficient if routing of clusters of pins, as discussed in the previous section can be adopted. The quality of routes obtained is same for both algorithms for three terminal nets. From our statistics, in 75% cases both algorithms yielded same result, in 10% cases, the paths were different, but had same length and corners (vias). In 10% cases, the PMR length was greater by 5%, and in 5% cases PAR-2 resulted in greater length by 8%.

Table 3. Device utilization summary of the total unit (clk 40ns)

Array size	Control Unit LUTs/FFs/ latches	Array LUTs/FFs/ latches	Total LUTs/FFs/ latches
4X4	95/21/112	1906/96/457	2001/117/569
8X8	95/21/112	7392/384/1792	7487/405/1904



(a) (b)

Figure 6. shows the performance of PMR and PAR-2 algorithms for three pin nets.

Thus, for PMR an overall increase in length is much less than 1%. Where as 11.7% nets resulted in more corners for PAR-2 against 8.3% nets of PMR. And from our knowledge PAR-2 becomes order dependent for more than three terminals and may not give an optimal solution, and PMR is not order dependent.

5. Summary and Conclusion

A new parallel processing wire routing algorithm for multi-point nets, is described. The PMR algorithm can find a path for multi-terminal nets having a quasi-minimum Steiner tree. The new design requires fewer clock cycles to find the shortest path between the terminals than the existing parallel routing algorithms.

7. References

[1] C. Y. Lee, "An Algorithm for Path Connections and its Applications," *IRE Transactions on Electronic Computers* vol. EC-10, no. 2, 1961, pp. 346-365.

[2] S. Akers, "A Modification of Lee's Path Connection Algorithm," *IEEE Trans. Electronic Computers* vol. EC-16, no. 2, pp. 97-98, 1967.

[3] M. Breuer and K. Shamsa, "A Hardware Router," *Journal of Digital Systems*, vol. IV, no. 4, pp. 393-408, 1981.

[4] T. Ryan, and E. Rogers, "An ISMA Lee Router Accelerator", IEEE Design of Test Computers, pp 38-45, October 1987.

[5] T. Watanabe, H. Kitazava, and Y. Sugiyama, "A Parallel Adaptable Routing Algorithm and its Implementation on a Two-Dimensional Array Processor", *IEEE Trans. CAD*, vol. CAD-6, no. 2, 1987.

[6] K. Suzuki, et. al., "A Hardware Maze Router with Application to Interactive Rip-Up and Reroute," *IEEE Trans. CAD*, vol. CAD-5, no. 4, pp. 466-476, 1986.

[7] J. A. Nestor, "A New Look at Hardware Maze Routing", *Proceedings Great Lakes Symposium on VLSI,* March 2002.

[8] W. Choi, and G. Sobelman, "Hardware Rip-up Router with Concurrent Waveform Propagation", Electronic Letters, Vol. 25 No. 6, pp373-374, 1989.

[9] M. Sarrafzadeh, and C. K. Wong, "An Introduction to VLSI Physical Design", McGraw-Hill, 1996.

[10] Chu, C., Yiu-Chung Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 27, Issue 1, pp.70 – 83, Jan. 2008.

Simultaneous Routing and Feedthrough Algorithm to Decongest Top Channel

Shashank Prasad
Cadence Design Systems
shasha@cadence.com

Anuj Kumar
University of Wisconsin Madison
kanujc@gmail.com

Abstract - In macrocell based SoC design, a routing plan to decongest top channel is an important step during floor planning. While previous approaches attempt at reducing congestion of chip as a whole, there is no attempt to specifically decongest top channel. We present an algorithmic approach to decongest top channel by using very few feedthroughs. Results show that compared to conventional methods, we can decongest top channel by using 20% lesser feedthrough buffers, and better top channel routing resource utilization.

1. Introduction

Due to enormous complexity of multi-million gate SoC designs, a top down hierarchical approach is useful for physical design. In this approach, a circuit is partitioned into a set of functional blocks (also called macrocells). The physical design flow of circuit (also called chip) is divided into following phases: floor planning/placement, pin assignment, partitioning, routing, assembly. In the floor planning/placement step, the dimensions and locations of the macrocells are determined. In the pin assignment step, the locations of pins (or crossing points) on the macrocell boundaries are determined. Partitioning divides the chip resources to macrocells. Routing is done individually in macrocells. It assigns routing regions for connecting the pins of the same net, and generates the actual geometric layout for the nets. Assembly merges macrocells back into a single chip. Usually, there is some routing channel available between macrocells for routing nets connecting two or more macrocells, or for routing nets connecting top glue logic with macrocells. In Figure 1A, macrocells are represented by M1-M5; top channel is represented by rectangular mesh outside macrocells. Nets *b*, *c*, *d, e* are connected to top glue logic, while net *a* connects logic in two macrocells M1 and M4.

As technology nodes are shrinking, SoC designs have larger number of transistors, while there is competitive pressure to reduce chip area. The routing resource in top channel is thus shrinking. It is important to ensure that there is no routing congestion in top channel. Otherwise, when placed and routed macro-cells are finally assembled together, there may not be enough routing resources to route all the nets between macrocells.

In order to reduce top channel congestion, it is imperative to push routes from top channel to channels inside nearby macrocells. In Figure 1, 1A shows a Steiner [8] chip routing plan. Figure 1B shows a no-feedthrough-scheme applied to 1A. In 1B, net *a*, which connects M1 and M4, is forced to go through congested top channel. On the other hand, if net *a* is subjected to feedthrough as in 1C, it can avoid top channel congestion by taking the path inside macrocell M3. Note that such feedthroughs also alter the hierarchical nature of the design, in that two new pins are created for net *a* in M3. Also most physical design tools require a feedthrough buffer to be added in M3 for net *a*, to avoid verilog assign statement, as they don't have a good strategy to deal with behavioral statements.

While feedthrough routes can effectively address top channel congestion, but it also leads to following issues:

- Macrocell M3 is no longer the original functional block, because now it has to route and optimize net *a* within itself. This undermines the functional nature of M3 and hierarchical nature of design.
- Too many feedthrough buffers could lead to over utilization of macrocell real estate. (Figure 1C: **E** in M3)
- Excess feedthrough buffers on very short nets could lead to over buffering of the timing path, leading to timing slack violation. (Figure 1C: net *e* near M4).
- Too many feedthrough routes could lead to under utilization of top channel. (For example, top channel in Figure 1C)

We have come up with an algorithm for chip route and feedthrough planning, which leads to no congestion in top channel while minimizing feedthrough routes and thus, feedthrough buffers.

Much research has been done on simultaneous global route planning and buffering. [1], [2] allow routes to go over macrocells but uses restricted buffer blocks for feedthroughs. A channel graph was used in [7] to perform the routing and allowed macrocell re-shaping. [4] develops a multi-level physical hierarchy generation algorithm integrated with fast incremental global routing for directly updating and optimizing congestion. A channel connection graph was used as the global routing graph in [6], and feedthrough paths inside macrocells are allowed in the algorithm. But, none of these methods deal with routing congestion in hierarchical physical design.

To handle multi layer layout pin assignment problem, [3] developed an algorithm based on an effective multi layer global router. It takes care of congestion hot spots while planning global routes. However, it does not pay any special attention or methodology to decongest top channel.

[5] describes a fast routing algorithm to generate a

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.83

hierarchically pure route topology (explained in Section III-A). Based on this, we have developed a two pass algorithm to decongest top channel. Using [5], we generate a hierarchically pure route topology. In this route topology, there is no over-the-macrocell route. However, as a result of this route topology, top channel is deeply congested. In the first pass, we sample over congested top channel grid cells, and collect data on nets which are passing through them. In the second pass, nets are then selectively ripped and re-routed through adjoining macro cells to decongest top channel. In doing so, very few feedthrough routes are used. The algorithm also pays due attention to address issues related to undue long routes, and timing violations on short routes due to excessive buffering.

The rest of the paper is organized as follows. Section II briefly reviews some feedthrough schemes on Steiner route topology. Section III describes our route topology and feedthrough algorithm. Section IV shows experimental results on some industrial circuits. Section V concludes the paper.

2. Steiner Routing and Feedthrough

Our goal is come up with an appropriate route topology and feedthrough scheme which work together to predict and decongest top channel. Figure 1A shows a Steiner route topology. Figure 1B, 1C, 1D show various feedthrough schemes applied on 1A.

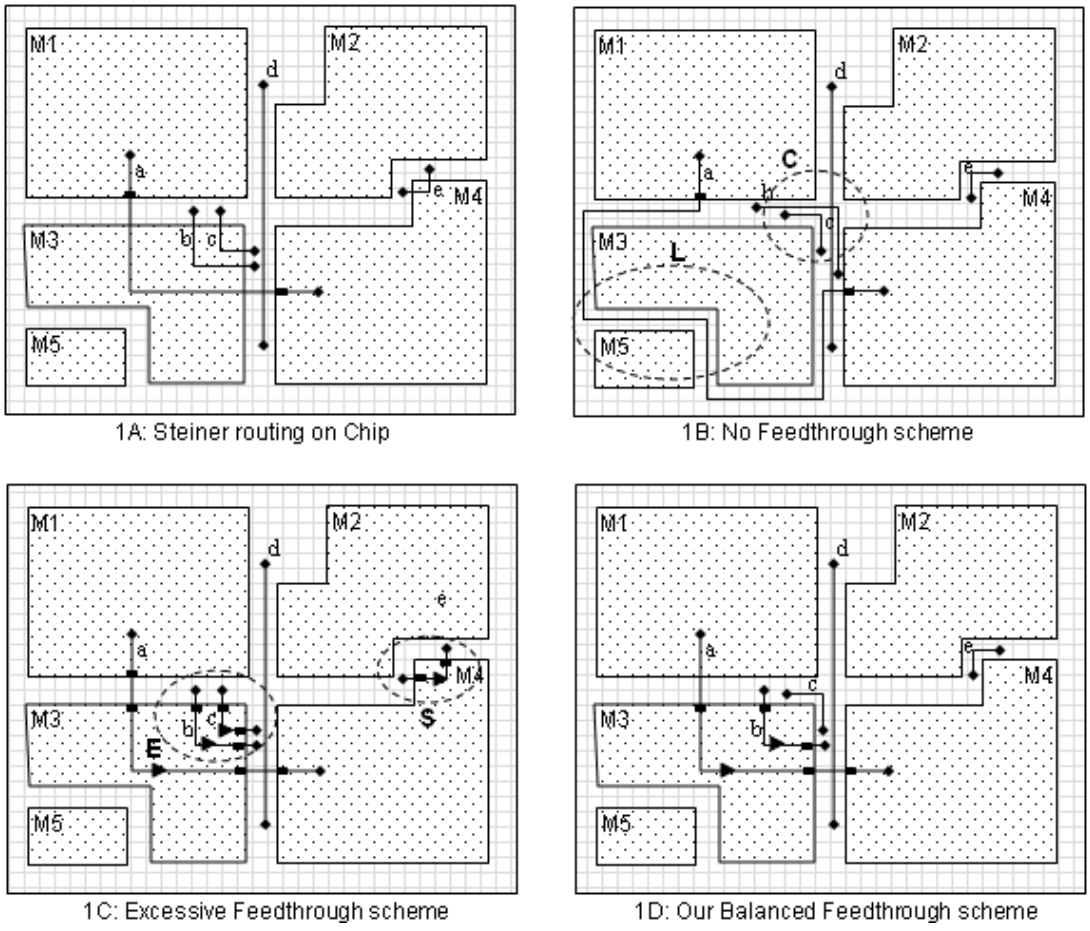


Figure 1: 1A, 1B, 1C, 1D show various feedthrough schemes on a Steiner routed chip design

Figure 1B shows the result of a no feedthrough scheme, wherein all pure top nets (like *b*, *c, d, e*) as well as nets interconnecting macrocells (like net *a*) are forced to avoid over-the-macrocell routing. This leads to congestion in top channel, denoted by **C** oval, and some long routes, denoted by **L** oval. Figure 1C shows excessive feedthrough scheme where all over-the-macrocell routes of 1A are subjected to feedthrough. This leads to excessive feedthrough buffers in M3, denoted by **E** oval, and feedthrough on short nets in M4, denoted by **S** oval. Figure 1D shows a balanced feedthrough scheme; where in only some over-the-macrocell-routing of 1A is subjected to feedthrough. For example:

- Net *e* is not subjected to feedthrough as that would create buffer on short net, potentially leading to timing

violation.

- Net *a* is subjected to feedthrough, otherwise it would lead to unduly long route (as shown in Figure 1B).
- Congestion shown in 1B is also avoided, as net *c* is subjected to feedthrough.

3. Our Route topology and Feedthrough algorithm

A. Hierarchically Pure Route Topology

[5] describes a PA (partition aware) routing model which basically emulates the route topology which would occur upon final assembly of routed macro cells (as explained in the next paragraph). [5] describes a routing algorithm to obtain this route topology upfront in the floor planning stage itself. Here, we call it hierarchically pure route topology. [5] shows that the run time to generate hierarchically pure routing is comparable to that of a prototype Steiner router used during in floor planning. It ensures hierarchically pure routing for at least 95% of pure top and inter macrocell nets. For rest of the nets, it follows Steiner route topology.

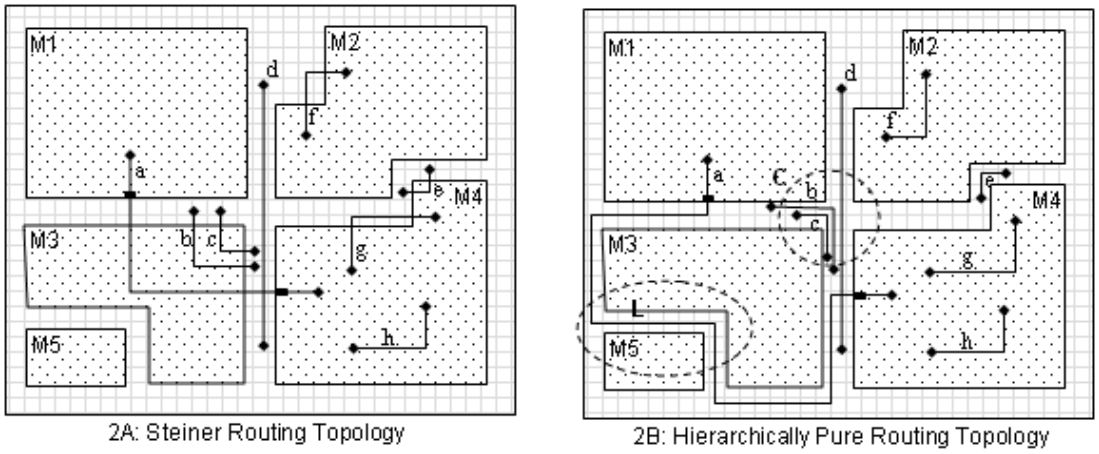


Figure 2: 2A, 2B show various route topologies on chip

Steiner routing is shown in 2A. Hierarchically pure routing is shown in 2B for the same design. In 2B, nets *b*, *c*, *d, e* which connect to only top connection points (not inside any macrocell), are routed in top channel only. Similarly nets *f*, *g*, *h,* which only connect to connection points completely inside a single macrocell, are routed within their respective macrocell routing resources. Net *a,* which interconnects two macro cells M1 and M4, is routed in M1, M4 and top channel only. Compared to 2A, 2B is useful in upfront prediction of top channel congestion and even chip congestion. [5] describes in detail the algorithm to come up with hierarchically pure route topology in a fast and fairly accurate manner. Note that 2B also creates unduly long routes, like net *a* (**L** oval), and excessive congestion, like **C** oval. However, there is no over-the-macrocell routing.

B. Hierarchically Pure Route Topology and Feedthrough

Our algorithm starts with hierarchically pure route topology as the starting point. It then, one-by-one, alters this route topology to create over-the-macrocell routes, with the aim to decongest top channel in the process. No short net feedthrough routes are created, neither is there any unnecessary under utilization of top channel.

Routing track is the smallest unit of routing resource. Grid cell is an abstract data structure used to represent a very small rectangular area. The chip routing resource could thus be represented by an array of grid cells (shown by the fine rectangular mesh in Figure 2A and 2B). Grid cell can be thought as to contain tracks which are within the boundaries of the grid cell across all routing layers. As part of routing the design, most routers also maintain congestion scores for every grid cell of the design. Also, given a route topology, routers can easily update congestion scores of grid cells. CongestionScore(g) tells how many extra routes are passing through grid cell g than what all its tracks can accommodate. Given a net's routing, it is very easy to extract as to which grid cells it is passing through.

Based on above terminology, the algorithm is as follows:

ROUTING _ALTERATION_AND_FEEDTHROUGH () {

1. Build hierarchically pure route topology.
2. Remove unduly long nets from top channel: If routing length of a net is greater than its Steiner routing length (by a tunable factor, default value is 1.2), then delete its routes. This step is done only for pure top net or nets inter connecting macro cells.

(i.e., Nets *b*, *c*, *d*, *e*, *a* in Figure 2B).

3. Update congestion scores of top channel grid cells to reflect congestion as per current route topology. If a grid cell is under utilized (i.e., not congested), then set its CongestionScore to 0.
4. Create a sorted list of congested top channel grid cells: say, *GCL*.
5. NET_PROPENSITY_TO_CONGEST(): Create a list of nets (say, *NL*) passing through any grid cell in *GCL*, and annotate congestion propensity scores on these nets, denoted by CongestionPropensity (*n*).
6. Sort *NL* in order of decreasing CongestionPropensity (*n*).
7. OPTIMAL_NETS_TO_FEEDTHROUGH(): Find an optimal subset of nets (say, *ONL*) such that when their routes are deleted, it reduces the CongestionScore(*g*) score of every grid cell in *GCL* to <= 0, i.e., it decongests top channel.
8. Delete routes of *ONL*.
9. Reroute these nets in *ONL* in a Steiner route topology, creating over-the-macrocell routes.
10. Feedthrough these over-the-macrocell routes into macrocells.

}

Step 2 and 3 can easily be done in linear time with respect to number of nets and grid cells respectively. Step 4 and 6 require sorting. Steps 5 and 7 are described below.

```
Algorithm NET_PROPENSITY_TO_CONGEST() {
1.  For each net n of the design {
2.      CongestionPropensity(n) = 0
3.  }
4.  For each grid cell g in GCL {
5.      For each n crossing over g {
6.          increment CongestionPropensity(n) by 1
7.      }
8.  }
}
```

```
Algorithm OPTIMAL_NETS_TO_FEEDTHROUGH () {
1.  For each net n of the design {
2.      Remove any bookmark on n
3.  }
4.  For each grid cell g in GCL {
5.  o Find set of nets from NL passing through g.
6.      Call this set as gN.
7.  o Suppose there are dN nets from gN
8.      which are already bookmarked to be deleted.
9.  o Decrement CongestionScore(g) by dN.
10. o If CongestionScore(g) > 0, then {
11.     o Sort gN in decreasing order.
12.     o Bookmark first CongestionScore(g) nets
13.         from gN to be deleted.
14.     }
15. }
16. For each net n of the design {
17.     If (n is bookmarked) add it to ONL
18. }
}
```

Algorithm OPTIMAL_NETS_TO_FEEDTHROUGH iterates over congested grid cells, and deletes just enough nets from each grid cell, so that its CongestionScore becomes <= 0. Care is taken to ensure that an already deleted net is accounted for while updating CongestionScore of grid cells. This makes it a one pass iteration on each grid cell of *GCL*. Also, nets with maximum propensity to congest the design, as reflected by CongestionPropensity(*n*), are deleted in priority over others.

ROUTING_ALTERATION_AND_FEEDTHROUGH thus deletes just enough nets to balance top channel routing congestion. By virtue of hierarchically pure route topology, no within-macrocell nets (like *f*, *g* and *h*) are routed in top channels. So, they are never picked for deletion and feedthrough. In Figure 2B, short nets, like *e* will normally have less CongestionPropensity than long nets like *a*. So, it is unlikely that *e* will be subjected to feedthrough unless they are really congesting the design. This ensures that the algorithm does not create any unnecessary short net feedthrough.

4. Results

We aim to demonstrate two things via experimental results. First, that utilizing feed through routes during chip global routing leads to significantly decreased top channel congestion. Second, that our feedthrough scheme indeed balances top channel congestion with significantly less number of feedthroughs. Actually, this latter point is the main thesis of this paper. To accomplish these objectives, we tested our program on industrial circuits shown in Table I. The net lists for these circuits already had a floor planning, placement and chip Steiner routing done. On the basis of this routing, we did pin assignment for macro cells, and then converted macro cells into hard blocks, and again routed the chip. Since the macro cells are now opaque hard blocks, this re-routing is done in top channels only. We call this Scheme 1 and measure top channel congestion at this point (and some more routing parameters as shown in Row1 of Table II). We repeat the above exercise once again, but this time before we convert macro cells into hard blocks, we insert feed through routes in macro cells, against those routes which go over the macro cells. Pin assignment and routing is then performed, after which top channel congestion is again measured (Scheme 2). Finally, once again, we repeat the above exercise, but this time, we use hierarchically pure route topology and ROUTING_ALTERATION_AND_FEEDTHROUGH to balance top channel congestion with feed through routes (Our Algorithm).

Table 1: Circuit Information

Circuit name	Macrocells	Standard cells (in millions)	Nets (in millions)	Pure Top and Inter Macrocell nets	Die area (sq mm)	Top Channel area (sq mm)
Ckt1	2	0.1	0.125	~700	9	0.9
Ckt2	6	1.25	1.35	~4000	64	6.1
Ckt3	8	6.1	7.5	25000	196	17.1

Table 2: Comparison of Top Channel Congestion, # of Feedthrough, Short Feedthrough, and Channel Utilization

Schemes	Top channel congestion (Ckt1, Ckt2, Ckt3)	# of Feed through buffers	# of Feed through on short nets	Top channel utilization	CPU run time (in minutes)
Scheme 1	(7%, 23%, 28%)	Not Applicable	Not Applicable	Over congested	(4, 22, 127)
Scheme 2	(0%, 0%, 0%)	(817, 4221, 5719)	(131, 719, 821)	(13%, 18%, 20%)	(6, 35, 191)
Our Algorithm	(0%, 0%, 0%)	(503, 3339, 4519)	(54, 158, 203)	(78%, 85%, 81%)	(8, 41, 217)

The top channel congestion for the three schemes is shown in Table II. Column 2 shows % of top channel grid cells which are congested. It clearly shows that there is no congestion in top channel when feedthroughs are used. It also shows that our scheme could achieve this with around 23% less feedthroughs compared to Scheme 2. Also, the channel routing resource utilization is better (80% vs. 20%) in our algorithm.

Table II also compares number of nets subjected to feedthrough, whose routing length is less than 400 microns. Doing feedthroughs on such short nets could result in timing slack violation due to over buffering of its timing path. Our algorithm does significantly better, over Scheme 2, by picking very few such nets for feed through (4% vs. 16%)

Table II compares CPU run time. Our algorithm takes around 10% more run time than Scheme2. Considering absolute run times involved, this is very much acceptable in floor planning stage.

5. Summary and Conclusions

We presented an algorithm to balance top channel congestion while minimizing the number of feedthrough routes. In the first pass, we analyze congestion hot spots and congested grid cells in top channel. Nets are given priority as per their propensity to pass through congested areas. In the second pass, a rip and re-route strategy is performed on high priority nets till top channel congestion is balanced. Previous approaches have not dealt with top channel congestion, and never made any attempt to do feedthrough for congestion balancing. This scheme ensures that minimum number of feedthrough routes is inserted. It also ensures that no undue feedthroughs are inserted on very short nets, which could otherwise lead to slack timing violation. The memory and run time of our scheme is comparable to that of prototype global router. It makes this scheme suited for industry designs with up to 40 macrocells and 5 million nets. While balancing top channel congestion, it can also be used as seed input for other floor planning steps like final pin assignment and timing budgeting.

6. References

[1] Hai Zhou, Wong D. F., Liu I-Min, and Aziz Adnan, "Simultaneous Routing and Buffer Insertion with Restrictions on Buffer Locations", in IEEE Transaction On Computer Aided Design of Integrated Circuits And Systems, Vol. 19, No. 7, July 2000, pp. 819-824

[2] ----, "Simultaneous routing and buffer insertion for high performance interconnect", in Proc. 6th Great Lakes Symp. VLSI, 1996, pp. 148-153.

[3] Liu L. E. and Sechen C., "A Multi-layer Chip-level Global Router," Fifth ACM/SIGDA Physical Design Workshop, 1996

[4] Chang C.-C., Cong J., Pan Z. D., Yuan X., "Physical Hierarchy Generation with Routing Congestion Control", in Proc. International Symposium on Physical Design, 2002

[5] Prasad Shashank, "Fast Congestion Aware Routing for Pin Assignment", IEEE VLSI Design 2008, pp. 343-347

[6] Wang L. Y., Lai Y. T., and Liu B. D., "Simultaneous pin assignment and global wiring for custom VLSI design", in Proc. IEEE Int. Symp. Circuits Systems, vol. 4, pp. 2128-2131, 1991.

[7] Koide T., Wakabayashi S., Yoshida N., "An integrated approach to pin assignment and global routing for VLSI building-block layout", in Proc. European Conf. Design Automation with the European Event in ASIC Design, Feb. 1993, pp. 24-28.

[8] http://en.wikipedia.org/wiki/Routing_(EDA)

Session 6C

Low Power Design

2009 22nd International Conference on VLSI Design

Metric Based Multi-Timescale Control For Reducing Power In Embedded Systems

Nitin Kataria, Forrest Brewer, João Hespanha, and Timothy Sherwood
Engineering I, University of California, Santa Barbara, CA 93106, USA
nitin@engr.ucsb.edu, forrest@ece.ucsb.edu, hespanha@ece.ucsb.edu, sherwood@cs.ucsb.edu

***Abstract*—Digital control for embedded systems often requires low-power, hard real-time computation to satisfy high control-loop bandwidth, low latency, and low-power requirements. In particular, the emerging applications of Micro Electro-Mechanical Systems (MEMS) sensors, and their increasing integration, presents a challenging requirement to embed ultra-low power digital control architectures for these lithographically formed micro-structures. Controlling electromechanical structures of such a small scale, using naive digital controllers, can be prohibitively expensive (both in power and cost for portable or battery operated applications.) In this paper, we describe the potential for control systems to be transformed into a set of co-operating parallel linear systems and demonstrate, for the first time, that this parallelization can reduce the total number of instructions executed, thereby reducing power, at the expense of controlled loss in control fidelity. Since the error tolerance of linear feedback control systems is mathematically well-posed, this technique opens up a new, independent dimension for system optimization. We present a novel Computer-Aided Design (CAD) method to evaluate control fidelity, with varying timescales on the controller, and analyze the trade-off between performance and power dissipation. A CAD Metric for control fidelity is proposed and we demonstrate the potential for power savings using this decomposition on two different control problems.**

I. Introduction

Digital feedback controllers make up a substantial part of modern embedded systems like ink-jet printers and anti-lock brake systems in cars. Their discrete-time realization, largely designed using classical unit-time sampling techniques, tends to dominate the design performance criteria because of real-time sampling and actuation constraints. These controllers are often implemented on micro-controllers for low bandwidth and digital signal processors or Field-Programmable Gate Arrays (FPGA) for higher bandwidth requirements. Their implementation exploits the linear systems approach for control realization by performing matrix-multiplication based updates of a set of state variables, or direct digital realization as an *Infinite Impulse Response* (IIR) digital filter.

In practice, however, such a strategy can lead to very inefficient, impractical, and expensive designs. For example, the performance of Atomic Force Microscopy (AFM) depends on the performance of the feedback control of the scanning cantilever, which is based on distributed FPGA implementations of the control algorithm to maintain the required loop latency. Another example is the design of portable MEMS sensors, whose diminutive physical size makes very high demands on loop-bandwidth requirements from the controller [1], [2]. Typically, a linear size scaling of a MEMS device linearly increases the required control-loop bandwidth, and with added requirements of noise shaping and filtering, makes the control algorithms computation intensive, running at sample rates of the order of a few MHz. These strict design requirements seriously hamper the development of small, portable MEMS based systems. Indeed, all current closed-loop MEMS designs are based on analog controller designs.

A promising alternative is to use a custom low-power hardware multi-threaded architecture [3]. Such an architecture enables processes that can be interrupted and interleaved at the granularity of single instruction cycles to meet timing (jitter) obligations. Further, we can substantially reduce latency and power by partitioning the control system based on bandwidth requirements and by performing only the most critical computations on the fastest thread, letting the slower parts run on slower threads. This nontrivial and difficult to analyze technique lowers the total number of instructions executed per second, which translates directly to significant power savings.

It is also possible to attempt algebraic manipulation [4] of the control algorithm to reduce latency and power without loosing significant control fidelity. However, this technique has several problems: First, even if the transformation is algebraically correct, round-off error and numerical stability issues often lead to unstable results [5]. Second, the space of possible control algorithms, which might be suitable for control decomposition, is very large – only a minuscule fraction of which are algebraically equivalent to yield the desired control law. Instead, we posit that it should be possible to use well defined controller performance metrics to evaluate the fidelity and power dissipation of a given embedded control design. This approach effectively increases the level of abstraction in the design since a candidate controller need not be equivalent in any sense other than

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.16

the evaluation of the performance metric.

In this paper, we take the first steps toward this approach by using a well defined metric for control stability and closed-loop performance. We use an exhaustive search based computer-aided design (CAD) method for exploring the design space by running a partitioned controller at different rates of computation. The metric allows us to explore trade-offs between performance and computation power and clearly shows that we can raise the level of abstraction for control design and optimization, beyond traditional methods. Rest of the paper is organized as follow: Section II present related work; Section III introduces the H-infinity framework, a basic control design methodology and performance metric with two case studies; Section IV presents our analysis of computation cost relative to performance within a bounding envelope of deviation from an ideal continuous controller. Finally, to conclude, we also demonstrate the use of *Voltage Frequency Scaling* (VFS) to reduce computation power significantly.

II. Related Work

Ad hoc techniques for low power multi-rate control have been used to implement a hard disk drive (HDD) controller in [6]. Ad hoc techniques for multi-rate control with interlacing of computations have been simulated and analyzed in [7], where the results show that the control performance deteriorates marginally with significant reduction in computation energy. Multi-rate digital control for motion control applications has been demonstrated, through various examples, in [8].

Software based power estimation for optimizing power in embedded systems is discussed in [9], [10]. An optimizing compiler, that minimizes execution time to save power has been proposed in [11]. A multiple clock and voltage domain, power-aware, tile-based embedded multiprocessor architecture has been proposed and analyzed in [12]. Architectural trade-offs for MEMS closed-loop control have been presented in [13], and a case study of hardware multi-threaded architectures for control has been discussed in [3].

III. Multi Rate Digital Controller Implementation

A. Control Design

The H-infinity framework has been used as the basic control design methodology and the control performance metric [14]. In this type of design, one starts by determining the points at which disturbances and noise are most likely to enter the system dynamics and then designs a feedback controller that minimizes the effect of these disturbances on specific signals that one wants to regulate, typically tracking errors (i.e., the difference between a signal and its desired value) and control signals. Specifically, denoting by w the vector of exogenous noise/disturbance signals that affect the dynamics of the process, sensors, and actuators and by z the vector of signals that one wants to regulate, an H-infinity controller minimizes the worst-case root-mean-square gain

$$J = \max_{w} \frac{\left(\int_0^\infty \|z(t)\|^2 dt\right)^{\frac{1}{2}}}{\left(\int_0^\infty \|w(t)\|^2 dt\right)^{\frac{1}{2}}}, \tag{1}$$

where the maximum is taken over all possible disturbance signals w. One can capture band-limited disturbances by introducing weighting pre-filters in w and one can also introduce weighting post-filters in z to capture the fact that one may only want to regulate this signal tightly over specifc bands of frequency. The value in equation (1) obtained by a specific controller is called the *H-infinity norm* of the closed-loop system.

For linear time-invariant continuous-time processes, the controller that minimizes the H-infinity norm is also linear time-invariant and operates in continuous time, thus requiring an infinite amount of computation. In practice, this cannot be implemented and thus one needs to approximate this continuous-time controller by a discrete-time controller that can be implemented digitally. Since this is not an optimal continuous-time controller, the discrete-time digital controller will not be able to achieve the minimum value for the H-infinity norm in equation (1), but hopefully does not increase it by too much. A side effect of this analysis is the generation of a lower bound for fidelity allowing derivation of sensible trade-offs.

Traditionally, a digital controller is obtained by discretizing a continuous-time controller using the *zero-order-hold* (ZOH) method with a fixed sampling interval T. For sufficiently large sampling rate, the resulting closed-loop exhibits an H-infinity norm very close to that of the continuous-time controller. However, if one has computational constraints that limit the sampling rate, then the performance may degrade significantly and may result in an unstable closed-loop system.

B. Multi-rate Digital Implementation

Here, we propose an alternative approach that is motivated by the observation that generally the optimal continuous-time controller exhibits a few high-frequency modes that need to be implemented at a high-sampling rate, while its low-frequency modes can be implemented at a lower sampling rate. In particular, suppose that the transfer function of the continuous-time controller is denoted by $K(s)$ and that we perform a partial fraction expansion to decompose this transfer function as a sum of two transfer functions as:

$$K(s) = K_1(s) + K_2(s)$$

where $K_1(s)$ is a first-order transfer function whose single pole is the fastest pole of $K(s)$ and $K_2(s) = K(s) - K_1(s)$. When the fastest pole of $K(s)$ has a non-zero imaginary part, then $K_1(s)$ is a second-order transfer function with the two fastest (complex conjugate) poles of $K(s)$. For digital implementation, we now have two degrees of freedom in selecting

the sampling rate, as we can discretize and implement $K_1(s)$ and $K_2(s)$ with two different sampling intervals T_1 and T_2, respectively.

C. Case Studies

We use two case studies to demonstrate the benefits that can be drawn from this approach: the position control of Quanser's DC servo motor [15] and the control of a MEMS tunneling accelerometer [16].

a) DC servo: Quanser's DC servo motor can be modeled by the following second-order transfer function from the input voltage to the shaft angle as:

$$G(s) = \frac{500}{s(s+53.5)}.$$

For the purposes of H-infinity design, we considered the effect of an additive input disturbance with energy concentrated below 100 rad/s (which is roughly twice the system's bandwidth) and flat-spectrum measurement noise with an amplitude 100 times smaller than that of the disturbance. The signals to be regulated by the H-infinity controller are the shaft angle and the applied voltage. Through the use of a weighting post-filter we mostly penalized the control signal at frequencies above 1000 rad/sec to prevent high frequency noise to flow through the feedback loop. These design choices led to the following H-infinity controller:

$$K(s) = \frac{-7.51(s+1001)(s+92.01)(s+73.99)}{(s+124.3)(s^2+317.9s+4.62e04)}$$

which achieved an optimal H-infinity norm of $J_{\text{opt}} = .1505$. For the multi-rate implementation, this controller was decomposed as:

$$K(s) = K_1(s) + K_2(s)$$

$$K_1(s) = \frac{-7.51(s+945.8)(s+32.72)}{s^2+317.9s+4.62e04}, \quad K_2(s) = \frac{-482.33}{s+124.3}$$

It is interesting to note that although the poles of $K_1(S)$ (with absolute value 215rad/sec) are not much faster than those of $K_2(s)$, this decomposition can still lead to significant computational savings, as we shall see below.

b) MEMS tunneling accelerometer: The MEMS tunneling accelerometer described in [16] is analyzed similarly. In this type of accelerometer, the feedback controller should use the applied voltage to cancel acceleration and one effectively gets the readout equation:

$$W_{\text{acc}}(s) \approx -187.6 V_{\text{app}}(s).$$

where Vapp(s) is the control voltage.

For the purpose of H-infinity design, the goal is to reject from the measured voltage the effect of acceleration, as well as noise. In constructing the vector w of exogenous inputs, noise terms were scaled to have unit standard deviation [16]. Additionally, a fictitious low-frequency noise term is added to the measured voltage, to make the controller insensitive to low frequency measurement biases such as 1/f amplifier noise. These design choices led to the following H-infinity controller:

$$K(s) = \frac{-1.21e04(s+6.29e04)(s+5.78e04)(s^2+5239s+6.95e08)}{s(s^2+1.14e05s+4.41e09)(s^2+2.23e05s+2.01e10)}$$

which achieves an optimal H-infinity norm of $J_{\text{opt}} = 4.6298$. For the multi-rate implementation, this controller is decomposed as:

$$K_1(s) = \frac{-1.41e04(s+2.91e04)}{s^2+2.23e05s+2.01e10}$$

$$K_2(s) = \frac{2.01e03(s+4.09e04)(s-1.85e04)}{s(s^2+1.14e05s+4.42e09)}$$

In this case as well, the poles of K1(s) (141881 rad/sec) are not much faster than those of K2(s) (66478 rad/sec) and yet significant computational savings can be achieved with this decomposition.

IV. Model for Computation Cost

The model for computation power cost focuses on the execution of the control algorithm and conservatively assumes that the cost of data acquisition and actuation is constant. Both examples are *Single Input Single output (*SISO) control systems, where the decomposed controllers share the same input, and their outputs are added together to generate the control output. The two controllers run as two separate threads of execution on two similar processors with no communication except the summation of the outputs, which is done in the faster thread. For the purpose of implementing digital control, the above two example controllers are discretized using *zero-order-hold* (ZOH) and implemented as shown in Figure 1.

A. Instruction Count

As a first order approximation of the computation cost, we determine the total number of processor instructions per second executed to implement a given control algorithm. For the purpose of our model it is convenient to implement the controller as cascaded, direct form II, structured second order sections (SOS), also known as *Biquads (Figure 2)* on a processor.

To evaluate instruction count, we used a Texas Instruments TMS320C2801 (Digital Signal Controller), which is a 32-bit, 8-level deep pipelined architecture, offering up to 100

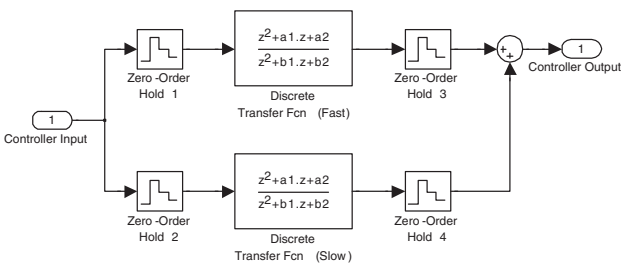


Figure 1. Parallel form decomposed controller.

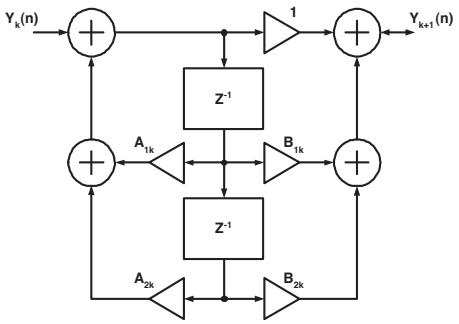


Figure 2. Structured second order biquad.

MIPS of performance and an ARM7TDMI, a 32/16-bit, RISC architecture offering up to 130 MIPS of performance. The total number of instructions required to implement a biquad section on each processor is determined through their individual library assembly routines [17], [18]. In Table I we list the total number of biquad sections needed to implement our control examples and the total number of instructions executed per sample time on the two processors. For a single-rate controller running at a frequency Fs and its decomposed version running at frequencies $Fs1$ (fast) and $Fs2$ (slow), the total number of instructions per second is determined as follows:

- Single-rate: Fs * instructions per biquad * number of biquad sections
- Multi-rate: $Fs1$ * instructions per biquad * number of biquad sections (fast) + $Fs2$ * instructions per biquad * number of biquad sections (slow)

Searching the sampling parameter space (sample time pairs for the fast and slow controllers) of the H-infinity framework, we generate a family of curves for the power/performance trade-off. It is then easy to determine Pareto optimal points suitable for controller realization. Figure 3 shows the computation-cost/performance trade-off for the DC servo motor control, where, the H-infinity norm is plotted against the total number of instructions/second (TMS320C2801) for a single-rate controller, and a multi-rate controller running at different rates of computation. The plot highlights the family of curves and the Pareto optimal points for single-rate and multi-rate implementation of control.

B. Power Estimation Model

The power estimation model measures power consumption of each thread, assumed to be running on individual processors in an architecture similar to *Synchroscalar* [12]. Although different strategies for realizing digital control would result in a different mix of instructions, and hence different power consumption, choosing the well studied biquad ensures that realistically optimized estimates can be determined. It is important to note that by decomposing the control, we benefit from reduced computation order in addition to enabling multi-rate computation. Therefore, once we know the total number of instructions/second and the instruction distribution for a control realization, we can use the results from a cycle accurate energy consumption measurement of a processor instruction set [19] to estimate the power dissipation. Figure 4 illustrates the Pareto optimal points for the case study of MEMS accelerometer control where H-infinity norm is plotted against the total power dissipation for an ARM7TDMI processor.

Table I
TOTAL NUMBER OF BIQUAD SECTIONS AND INSTRUCTION COUNT PER BIQUAD

Example	Controller order	Biquad sections	Instr. count C2801	Instr. count ARM7TDMI
DC servo	3	2	56	54
MEMS Accel.	5	3	79	81

In Table II we show the computation cost for DC motor control, and, compare the results for the single-rate uniformly sampled controller against a few configurations of the multi-rate implementation, for near similar values of H-infinity norm. It is interesting to note that performance is non-monotone with sample rates and that careful selection of rates can lead to higher overall fidelity than is achievable by single-rate sampling. In Table III we show the power consumption and the H-infinity norm for the MEMS single-rate controller versus a few configurations of multi-rate controller, implemented on an ARM7TDMI.

C. Voltage Frequency Scaling

In this section we postulate that an architecture similar to *Synchroscalar* [12] can be used to implement decomposed control algorithms to further reduce power. Each thread of computation is mapped onto a different processor core running at its optimized voltage and frequency requirements. The voltage/frequency pairs are statically scheduled after determining the real-time requirement of control loops. This

Table II
COMPARISON OF ESTIMATED TOTAL NUMBER OF INSTRUCTIONS/SECOND EXECUTED BY SINGLE-RATE AND MULTI-RATE CONTROLLERS USING TMS320C2801

Single-rate			Multi-rate			
Ts (ms)	H-inf.	Instr. / sec.	Ts1(ms) (fast)	Ts2(ms) (slow)	H-inf.	Instr. / sec.
1	0.24	53000	1	4	0.25	41250
			1	6	0.26	38478
			1	14	0.31	35343
2	0.34	28000	2	4	0.33	24750
			2	8	0.35	20625
			2	11	0.37	18843
3	0.45	18648	3	5	0.41	17589
			3	16	0.47	10593
			3	18	0.48	10065
4	0.56	14000	4	8	0.51	12375
			4	14	0.55	10593
			4	18	0.61	10065
5	0.69	11200	5	8	0.64	10725

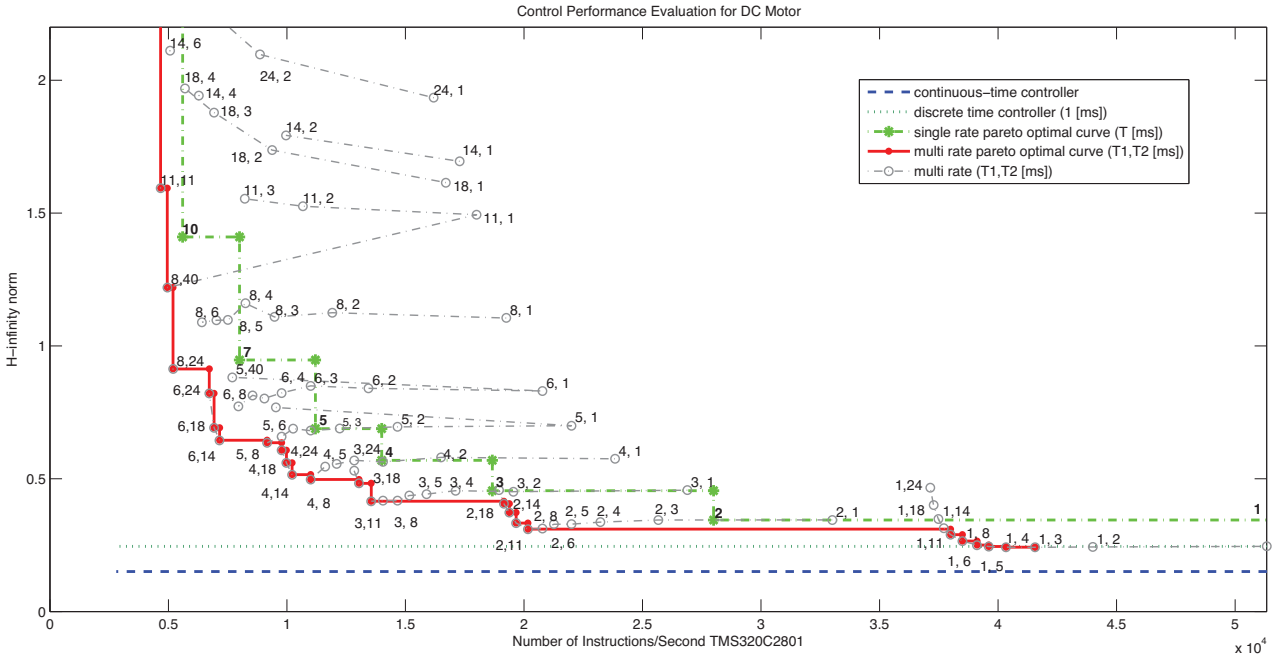


Figure 3. DC Motor performance versus instruction count trade-off. Increasing values of H-infinity norm indicate decreasing control performance.

Table III
COMPARISON OF ESTIMATED POWER CONSUMED BY COMPUTATIONS IN SINGLE-RATE AND MULTI-RATE CONTROLLERS USING ARM7TDMI

Single-rate			Multi-rate			
Ts (μs)	H-inf.	Power (μW)	T$s1$ (μs) (fast)	T$s2$ (μs) (slow)	H-inf.	Power (μW)
5	6.08	10782	5	10	7.46	7188
			5	15	10.59	6090
10	11.46	5391	10	20	11.83	3594
			10	25	16.55	3292
			10	30	20.05	3234
15	30.65	3494	15	20	23.92	2994
			15	25	18.71	2635
			15	30	17.79	2395
75	43.37	718	75	100	44.85	638

approach is different from *Dynamic Voltage Frequency Scaling* (DVFS) [20].

Advantages of statically scheduled voltage frequency pairs are more prominent if we consider, for example, a computation architecture that needs to control a 3-Dimensional (3-D) MEMS accelerometer. From our MEMS control example, we needed a fifth-order controller for 1-Dimensional (1-D) control, therefore, for a 3-D MEMS accelerometer, we would need to execute at least three times the total number of instructions executed for the 1-D controller, within the same execution deadline. Such a controller would require a very fast processor, consuming a lot of power. Instead, we can partition each fifth-order controller, like before, into two separate threads, and run them individually as three fast and three slow threads on *Synchroscalar*. Table IV shows the power consumption of a 3-D MEMS controller implemented as a single thread of execution versus six separate threads of execution, running at different rates on *Synchroscalar*.

V. CONCLUSIONS

In this work, we have shown that it is possible to raise the level of abstraction in the design power analysis for linear feedback controllers in a way that directly relies on well-known metrics for the design of such controllers. This is in contrast to algebraic methods which can reach only a small fraction of the same design space. The new techniques exploit simple multi-thread or multi-core architecture tricks to allow practical implementation where dependence of the partitioned controllers is minimal. Using these techniques, very substantial power savings are obtained in a way that is independent from conventional algebraic tricks such as sub-expression decomposition and related high-level synthesis techniques. Although the current approach is exhaustive, for practical scale designs it can be performed in a very reasonable amount of time. Further work in this area will likely target more efficient exploration techniques as well

Table IV
ESTIMATED POWER CONSUMED BY A 3-D MEMS CONTROLLER

Controller type	Thread count	Total cycles	Cycle time	Volt. (V)	Freq. (MHz)	Power (mW)
Single-rate Ts (5μs)	1	531	9(ηs)	0.8	120	23.17
Multi-rate Ts1 (5μs) Ts2 (10μs)	3 (fast) 3 (slow)	117 354	85(ηs) 85(ηs)	0.7	10	14.14

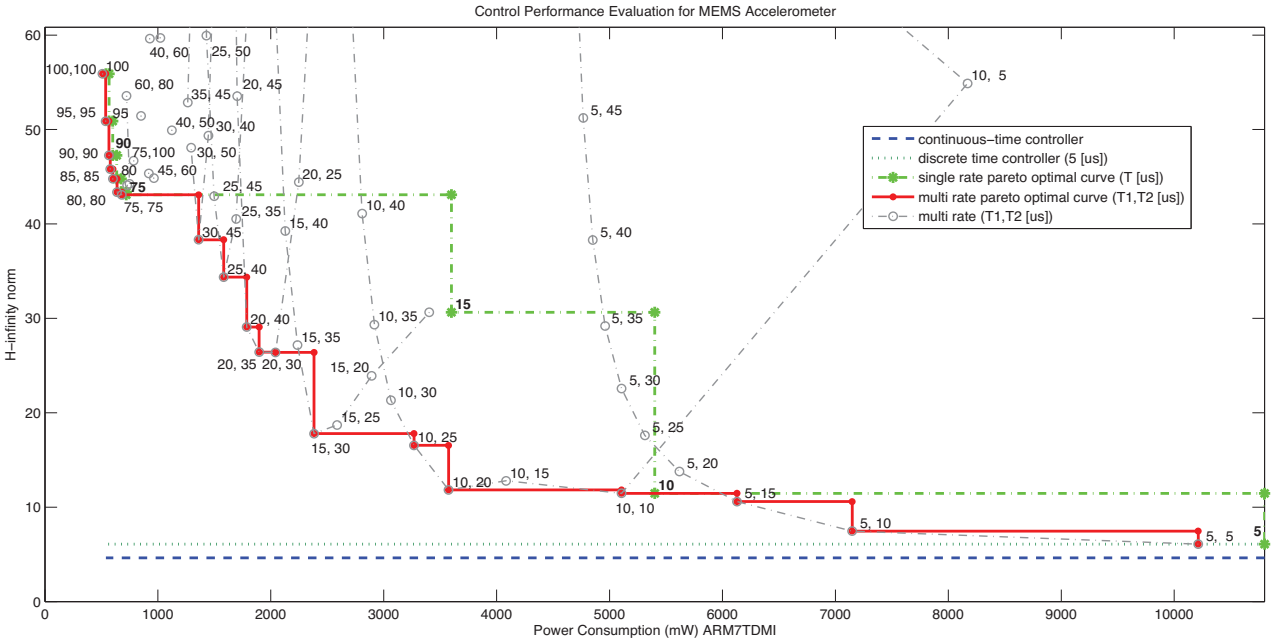


Figure 4. MEMS accelerometer performance versus power consumption trade-off. Increasing values of H-infinity norm indicate decreasing control performance.

as alternative design metrics suitable for sensors and other related designs.

REFERENCES

[1] M. K. L. O.-R. K. Turner, "Robust feedback control design of an ultra-sensitive, high bandwidth tunneling accelerometer," *American Control Conference, 2005. Proceedings of the 2005*, vol. 6, pp. 4176– 4180, June 2005.

[2] J. Bryzek and E. Abbott, "Control issues for mems," in *Proc. of the 42nd IEEE Conference on Decision and Control*, 2003, pp. 3039–3047.

[3] G. Hoover, T. Sherwood, and F. Brewer, "A case study of multi-threading in the embedded space," in *CASES '06: Proceedings of the 2006 international conference on Compilers, architecture, and synthesis for embedded systems*, 2006, pp. 357–367.

[4] A. Hosangadi, F. Fallah, and R. Kastner, "Common subexpression elimination involving multiple variables linear dsp synthesis," in *ASAP '04: Proceedings of the Application-Specific Systems, Architectures and Processors, 15th IEEE International Conference*. Washington, DC, USA: IEEE Computer Society, Sept. 2004, pp. 202–212.

[5] N. Sureshbabu, B. Powell, and M. Dunn, "An integrated procedure for fixed-point control implementation," *American Control Conference, 1998. Proceedings of the 1998*, vol. 5, pp. 3096–3100 vol.5, 21-26 Jun 1998.

[6] J. Ding, F. Marcassa, S.-C. Wu, and M. Tomizuka, "Multirate control for computation saving," *Control Systems Technology, IEEE Transactions on*, vol. 14, no. 1, pp. 165–169, Jan. 2006.

[7] S.-C. Wu and M. Tomizuka, "Multi-rate digital control with interlacing and its application to hard disk drive servo," *American Control Conference, 2003. Proceedings of the 2003*, vol. 5, pp. 4347–4352 vol.5, 4-6 June 2003.

[8] M. Tomizuka, "Multi-rate control for motion control applications," *Advanced Motion Control, 2004. AMC '04. The 8th IEEE International Workshop on*, pp. 21–29, 25-28 March 2004.

[9] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: a first step towards software power minimization," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 2, no. 4, pp. 437–445, Dec 1994.

[10] M. T.-C. Lee, V. Tiwari, S. Malik, and M. Fujita, "Power analysis and minimization techniques for embedded dsp software," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, no. 1, pp. 123–135, Mar 1997.

[11] J. Russell and M. Jacome, "Software power estimation and optimization for high performance, 32-bit embedded processors," *Computer Design: VLSI in Computers and Processors, 1998. ICCD '98. Proceedings., International Conference on*, pp. 328–333, 5-7 Oct 1998.

[12] J. Oliver, R. Rao, P. Sultana, J. Crandall, E. Czernikowski, I. Jones, L.W., D. Franklin, V. Akella, and F. Chong, "Synchroscalar: a multiple clock domain, power-aware, tile-based embedded processor," *Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on*, pp. 150–161, 19-23 June 2004.

[13] G. Hoover, T. Sherwood, and F. Brewer, "Towards understanding architectural tradeoffs in mems closed-loop feedback control," in *CASES '07: Proceedings of the 2007 international conference on Compilers, architecture, and synthesis for embedded systems*, 2007.

[14] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. New Jersey: Prentice Hall, 1996.

[15] "Rotary servo plant with encoder, srv02." [Online]. Available: http://www.quanser.com/

[16] C. B. K. Å. F. B. L.A. Oropeza-Ramos, N. Kataria and K. Turner, "Noise analysis of a tunneling accelerometer based on state space stochastic theory," in *Solid-State Sensor, Actuator, and Microsystems Workshop, Hilton Head, SC June 2008*.

[17] C. W. Andrew N. Sloss, Dominic Symes, "Arm system developer's guide designing and optimizing system software," http://www.arm.com, Advanced RISC Machines Ltd., Tech. Rep.

[18] "Filter library, module user's guide, c28x foundation software," http://www.ti.com, Texas Istruments, Tech. Rep., 2002.

[19] N. Chang, K. Kim, and H. G. Lee, "Cycle-accurate energy consumption measurement and analysis: Case study of arm7tdmi," *Low Power Electronics and Design, 2000. ISLPED '00. Proceedings of the 2000 International Symposium on*, pp. 185–190, 2000.

[20] G. Magklis, G. Semeraro, D. Albonesi, S. Dropsho, S. Dwarkadas, and M. Scott, "Dynamic frequency and voltage scaling for a multiple-clock-domain microprocessor," *Micro, IEEE*, vol. 23, no. 6, pp. 62–68, Nov.-Dec. 2003.

2009 22nd International Conference on VLSI Design

Code Transformations for TLB Power Reduction*

Reiley Jeyapaul, Sandeep Marathe and Aviral Shrivastava

Compiler and Microarchitecture Laboratory, Arizona State University, Tempe, AZ 85281 USA
Email : {reiley, sandeep.marathe, aviral.shrivastava}@asu.edu

Abstract

The Translation Look-aside Buffer (TLB) is a very important part in the hardware support for virtual memory management implementation of high performance embedded systems. The TLB though small is very frequently accessed, and therefore not only consumes significant energy, but also is one of the important thermal hot-spots in the processor. Recently, several circuit and microarchitectural implementations of TLBs have been proposed to reduce TLB power. One simple, yet effective TLB design for power reduction is the Use-Last *TLB architecture proposed in [9]. The Use-Last TLB architecture reduces the power consumption when the last page is accessed again. While very effective for instruction TLB, this technique is not as effective for the data TLB. In this paper, we propose compiler techniques (specifically, instruction and operand reordering, array interleaving, and loop unrolling) to reduce the page switchings in data accesses. Our comprehensive page-switch reduction algorithm results in an average of* 39% *reduction in the data-TLB page switching, and therefore power with negligible variation in performance on benchmarks from MiBench, Multimedia, DSPStone and BDTI suites.*

1. Introduction

Power, energy and thermal issues in current and near future digital systems form the crux of the biggest challenge that the semiconductor industry faces today. In high-end computing, power consumption limits the amount of achievable performance because of exorbitant increase in the cost of heat removal mechanisms. In battery operated portable systems, the battery is the single largest factor in device cost, weight, recharging time, frequency and ultimately the usability of the system. Translation Look-aside Buffer or TLB is an important component of high-end multi-tasking embedded processors, like the Intel XScale. The TLB performs virtual to physical address translation and determines page access permissions. Most modern processors, including the Intel XScale implement virtually-addressed caches, in which the cache lookup is directly performed on the virtual address provided by the processor, and therefore the TLB lookup comes in the critical path. Elkman et al. [11] note that the TLBs can consume 20-25% of the total $L1$ cache energy. Kadayif et al. [14] observed high power densities of the data-TLB, as compared to the data-$L1$ cache. Thus reducing the power consumption of TLBs is an important research problem.

Several TLB designs have been proposed to trade-off the TLB lookup delay, area and power consumption [12, 15]. One simple, yet effective technique for TLB power reduction proposed in [10, 9], is the *Use-Last* TLB architecture. Observing that there is a high probability that instruction access will refer to the same page as the last one, they store the previous page translation information into a latch, and thereby reduce the TLB lookup power. The Use-Last TLB architecture is able to reduce the instruction TLB power by 75%. However, since data accesses do not exhibit as high locality as instructions, this microarchitectural technique was not effective for data TLBs.

We develop compiler techniques to reduce the power consumption of the Use-Last TLB architecture by improving the locality of data accesses. We propose a novel instruction scheduling and operand reordering technique, heuristic for deciding when to perform array interleaving, and loop unrolling to minimize the page switchings between consecutive TLB accesses while minimizing performance loss. Our combined technique can reduce the TLB switches by 39%, with minimal performance impact on benchmarks from MiBench, Multimedia, DSPStone and BDTI suites. Note that this improvement is above and beyond what the Use-Last hardware technique alone could achieve.

2. Related Work

Several researchers have proposed efficient circuit-level, microarchitectural and software techniques to reduce the power consumption of the TLB and the Memory Management Unit.

2.1. Compiler based Approaches

A compiler-directed array interleaving technique [17] was proposed to save energy in multi-bank memory architectures with power control features. In this, the arrays used in separate banks are interleaved such that only one of the banks is active and the other can be powered down, thus saving energy. The energy reduction achieved by this technique does not account for the leakage power of the SRAM cells during standby mode. Parikh et al in [18] schedule instructions within a block based on the minimum obtainable value for a weighted cost function:*circuit-state cost*. One recent work is [19], where energy reduction is achieved through effective utilization of resources by switching between two processor modes based on the cache misses.

2.2. Closest Approach

The work closest to our approach, is by Kandemir et al. [13]. Their compiler technique is to increase the effectiveness of a previously proposed architectural technique that uses *Translation Registers* or TRs. The addition of TRs requires changing the ISA, which may not be desirable in many cases. In contrast,

*This work was partially funded by grants from Raytheon and Startdust Foundation.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.39

our approach is to improve the effectiveness of Use-Last TLB architecture, which exists in the Intel XScale processor. They have to profile the code to find out which page will be accessed frequently in the near future, and then generate code to load the translations to that page into TRs. In comparison, our approach is a static technique. We do not need/use profile information. Not only that profile-based compilation is limited in application and scope, it has huge overhead in terms of compilation time. Our technique does not have any such overheads. Finally, in their technique, the code is modeled as nodes which represent loop nests that access data from a particular page region. Code transformations to enhance the use of TRs are directed at scheduling these loop nests (nodes that access data from a particular page region) together. In contrast, our approach is to schedule and transform instructions so that the accesses to the same page are grouped together. Our technique operates at a finer granularity than theirs, and could therefore co-exist, and enhance the effectiveness of each other.

3. *Use-Last* TLB Architecture

Our compilation approach enhances the effectiveness of an already effective and popular TLB architecture, the *Use-Last* TLB architecture. Proposed in [9], the Use-Last TLB architecture utilizes a modified TLB-CAM structure. The virtual address input is matched with the TLB tag through the CAM cells (which has reduced power consumption). The TLB tag is then used to retrieve the mapped physical address from the register files. The lookup on the register files is a power consuming process because of the bit-line and word-line drivers and other associated circuitry involved in its operation. The key factor in this architecture design is the latch used to store the tag address of the previously accessed address. If the two TLB tag addresses match, the page address and access information at the output is the same for both. In this case, the word line (WL) of the register files are not activated and the switching energy of the RF cells and associated circuitry is eliminated. The effectiveness of this technique was demonstrated on instruction TLB, and it was shown to reduce the power consumption by 75%. However, this technique was deemed un-useful for data caches, as data accesses in general do not exhibit high data locality as compared to instruction TLB. Our work aims to enhance the effectiveness of this architectural technique on data caches through code transformations and achieve power savings through reduction in the number of page-switches during successive data accesses.

4. Experimental Setup

We explore and develop compiler techniques for the Intel XScale processor [8] on which the *Use-Last* architecture was implemented(Section 3). Intel XScale is an out-of-order, 7-stage superpipelined high-end embedded processor, which runs at up to 1 GHz. The Intel XScale uses TLBs to implement virtual memory support. The Intel XScale is intended to be used in wireless and handheld applications and therefore we execute benchmarks from MiBench [3], MultiMedia [6], DSPstone [4], Spec2000 [5], and the BDTI [7] benchmark suites. The *sim-outorder* cycle-accurate simulator of the SimpleScalar toolset [16] was modified to model the Intel XScale memory configuration and to determine the total number of page switches in the data TLB in a program.

The remainder of the paper is organized as follows: Section 5 describes our instruction scheduling and operand reordering technique. Section 6 describes our array interleaving implementation. Section 7 describes the conditions for our implementation of loop unrolling. Section 8 forms a comprehensive algorithm for TLB page switch reduction.

5. Page Switch-Aware Instruction Scheduling

Instruction scheduling can aggregate instructions that access the same pages consecutively, thereby reducing page switches in the data TLB. In addition, for commutative operations, it is also possible to reorder the operands, and effect the memory access pattern. We develop a combined instruction scheduling and operand reordering technique to reduce TLB page switching.

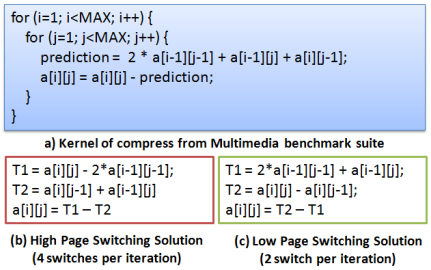


Figure 1. Impact of code generation on TLB page switching

5.1. Motivation

We motivate the applicability and effectiveness of fine-grain instruction and operand reordering on TLB page switches using a kernel from the *compress* benchmark, shown in Figure 1(a). The kernel accesses elements from a two-dimensional array. If the array size is much larger than the page size(which is typically small in embedded systems), elements from the higher dimensions may reside in different pages. In this example, there are high chances that $a[i]$, and $a[j]$ may be in different pages, if $i \neq j$. Assuming this, the two code sequences generated by the compiler, illustrated in Figure 1(b) and (c), may result in the same performance, they may differ significantly in the number of TLB switches they cause. When executed, the code in Figure 1(b) will result in accesses in the sequence: $a[i][j]$, $a[i-1][j-1]$, $a[i][j-1]$, $a[i-1][j]$, and $a[i][j]$, which will result in 4 page switches per iteration, while the code in Figure 1(c) will result in only 1 page switch per iteration. Note that depending on the cache size and page size, the page switches can vary, but if there is no performance impact, it will be better to generate the code as in Figure 1(c). In the rest of this section, we first formulate the problem of minimizing the page switches by instruction scheduling and operand reordering. Finding the problem to be NP-complete, we propose a heuristic for the same.

5.2. Problem Formulation

Input: Data Flow Graph (DFG) is a directed acyclic graph (DAG) $D = (V, E)$ of a code sequence. The nodes $v \in V$ represent instructions $i \in I$. An instruction i is represented by a ordered $(k+2)$-tuple $i =< op, d, s_1, s_2, ...s_k >$, where op is the opcode, d is the destination, and there are k source

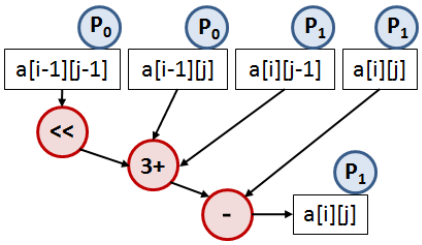


Figure 2. DFG and page mapping of compress kernel

operands, $s_1, ...s_k$, $d, s_1, s_2, ...s_k \in O$, where O is the set of program variables, or operands. There is a directed edge $e = (v_1, v_2) \in E, \exists\ v_1, v_2 \in V$, from v_1 to v_2 if the destination of the instruction represented by node v_2, is the same as any of the source operands of the instruction represented by node v_1. i.e., $(v_1.i.d = v_2.i.s_1) \vee (v_1.i.d = v_2.i.s_2) \vee ... \vee (v_1.i.d = v_2.i.s_k)$. The data flow graph will also have nodes at the beginning of the graph, representing loading of operands, and nodes at the end of the graph, representing storing of operands, or intermediate values that will be carried over to the next loop. The DFG of the compress kernel is illustrated in Figure 2.

Output: Instruction Sequence represented by the function $Time : I \rightarrow \mathbb{N}$ such that all data dependencies are maintained. i.e., if there is an edge from instruction i_a to i_b, then $Time(i_a) < Time(i_b)$.

Objective: Minimize Page Switches in the instruction sequence. To estimate page switching at the compiler level, we define a function $Page : O \rightarrow P$, which maps operands $o \in O$ to pages $p \in P$, where P is the set of all the pages accessed by the application. A source operand may be a scalar, or an array, and can be defined in a local scope or a global scope. We define $Page(s)$ thus:

- $Page(s) = undefined$ if the operand s is a local scalar variable. This is because most probably all the local scalar variables will be allocated to registers and therefore will not involve in memory access.
- $Page(s) = p_0$ if s is a global scalar variable. We assume that all the global scalars are allocated to a single page.
- For the global or local arrays, we assume that each array, irrespective of it's size is mapped to exactly one unique page.

Page Switch Model In addition, we also need a page switch model, i.e., given a sequence of instructions, how many page switches will occur. We assume that when an instruction i executes, its operands are accessed in the order $\{i.s_1, i.s_2, ..., i.s_k, i.d\}$. Assuming that the page accessed just before the execution of an instruction i is p, then, we define the page switching function, $PS_I(p, i_1, ...i_n)$ to be the number of page switches when a sequence of instructions $i_1, ...i_n$ is executed.

$$\begin{aligned} PS_I(p, i_1, ...i_n) &= PS_O(p, i_1.s_1, i_1.s_2, ..., i_1.s_k, i_1.d, \\ &= i_2.s_1, i_2.s_2, ..., i_2.s_k, i_2.d, \\ &= ..., \\ &= i_n.s_1, i_n.s_2, ..., i_n.s_k, i_n.d) \end{aligned}$$

The total page switch count between operands can be recursively computed,

$$\begin{aligned} PS_O(p, o_1, ..., o_m) = &PS_O(p, o_1) \\ &+ PS_O(LP_O(p, o_1), o_2, ..., o_m) \end{aligned}$$

where $PS_O(p, o) = 1$, when both p and $Page(o)$ are defined, and $p \neq Page(o)$. $LP_O(p, o)$ is the last page accessed when operand o_1 is accessed after accessing page p. The last page function $LP(p, o) = Page(o)$, if $Page(o)$ is defined, otherwise, it is p.

5.3. Solution for Page Switch Minimization

To minimize page switches by instruction scheduling and operand reordering, we define a Page Switching Graph $PSG\_full = (I, S)$, which is a directed graph, whose vertices are instructions $i \in I$, and there is an edge from instruction i to instruction j if instruction j can be scheduled immediately after instruction i. We attach a weight attribute to each edge $w(i, j)$, which is the minimum increase in the page switches when instruction j is scheduled immediately after instruction i. Thus,

$$w(i,j) = \begin{cases} min \begin{cases} PS_O(p, j.s1, j.s2, j.d) \\ PS_O(p, j.s2, j.s1, j.d) \end{cases} & \text{if } j.op \text{ is comm} \\ PS_O(p, j.s1, j.s2, j.d) & \text{otherwise} \end{cases}$$

where p is the last page that has been accessed after instruction i is executed. We add a dummy source node, and a sink node so that there is an edge from the source node to all the instructions that do not have any predecessors in DDG, and there are edges all nodes that do not have successors in DDG to the sink node. Dummy nodes access only $undefined$ pages.

The problem of finding the instruction sequence and operand ordering that minimizes the number of page switches is exactly equal to the problem of finding the shortest hamiltonian path from source node to sink node. This implies that if we can solve the problem of page switch minimization in polynomial time, we can also solve the hamiltonian problem, which is a well known NP-Complete problem in polynomial time. This is quite unlikely, therefore the problem of scheduling for page switch minimization is NP complete. Therefore we focus our efforts on developing scheduling heuristics for page switch minimization.

5.4. Heuristic for Page Switch Minimization

For heuristics, we first construct a Page-Not-Switching Graph $PNSG = (I, D, S)$, where the nodes (I) are instructions, and there are two kinds of edges, first is the set of data dependence edges D, and the second S is the set of inter-instruction page not-switching edges. Thus there is a an edge $s = (i, j) \in S$ between two instructions: $i, j \in I$, if there is NO inter-instruction page switch when instruction j is scheduled immediately after instruction i. In other words, $(i, j) \in S, \forall i, j \in I$, iff $Q_{ps} \geq 1$, where

$$Q_{ps} = \begin{cases} min \begin{cases} PS_O(p, undefined, i.d, j.s1) \\ PS_O(p, undefined, i.d, j.s2) \end{cases} & \text{,if } j.op \text{ is comm} \\ PS_O(undefined, i.d, j.s1) & \text{otherwise} \end{cases}$$

An example of a $PNSG$ is shown in Figure 3. The nodes 1 through 7 are instructions, and the solid edges represent data dependencies. The dashed edges represent the inter-instruction page not-switching edges. We now perform our scheduling on this graph representation. We first developed a greedy algo-

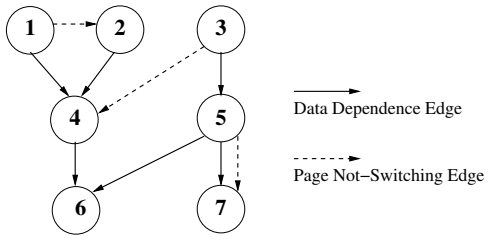


Figure 3. Problem in greedy solution

rithm. In the greedy algorithm, in every iteration, the *last scheduled instruction*, l is maintained, and list of instructions that are now ready to be scheduled, R is created. If there is a page-not-switching edge between l and any instruction $r \in R$, then r gets priority, as it minimizes the page switches. Thus suppose instructions $1, 2$ and 3 are scheduled, with $l = 3$. Then R can be computed as $R = \{4, 5\}$. Out of these, the greedy heuristic will pick up instruction 4.

Figure 3 illustrates one problem with this simple approach. In the first iteration, the greedy solution can pick up either instruction 1, or instruction 3. Picking up instruction 3 is a bad choice, because it is not possible to schedule instruction 4 as the second instruction. Instruction 3 should only be scheduled only if instruction 4 can be scheduled next. We fix this problem by adding that - when picking an instruction which is the source of a page-not-switching edge, we pick up a pair of instructions to schedule; plus, we give priority to pick up instructions that are not connected through page-not-switching edges. This gives us more opportunities to pick up instruction pairs with page-not-switching edges.

5.5. Experiments

We have implemented this page-aware instruction rescheduling algorithm as a compiler post-pass [1]. We compile our benchmarks with GCC -O3 optimization, to ensure that the benchmarks are compiled and scheduled for the maximum performance. We disassemble the generated object file, discover the basic blocks, and re-create the control flow graph (CFG), and the data flow graph. We perform this modified list scheduling heuristic on basic blocks. This fine grain instruction scheduling approach is applicable to any program. The effectiveness of this approach could be increased by performing our scheduling on hyperblocks, and/or superblocks. We observed that our scheduling gains from performing local reordering of load instructions. There is not much increased opportunity to move load instructions across basic blocks, because of tight data dependencies.

We modified the sim-outorder [16] simulator to count the page switches for an application execution. Figure 4 plots the page switch count, after implementing our page-aware instruction scheduling and operand re-ordering transformations normalized to the baseline page switch count. On an average, our technique achieves 23% reduction in the page switch count as indicated by the right-most bar in Figure 4. As a matter of fact, we observed an average performance improvement of 4%. This reduction in page switches directly translate into 23% power savings in the Use-Last TLB. Note that this is over and above what Use-Last TLB architecture achieves on its own.

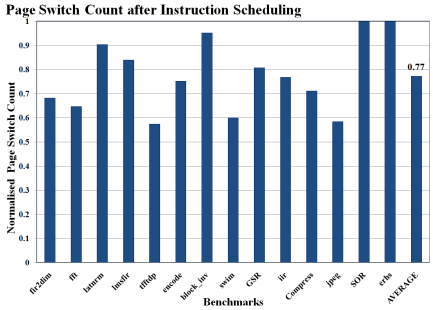


Figure 4. Impact of Instruction Scheduling on Page Switch Count

6. Page-Switch Aware Array Interleaving

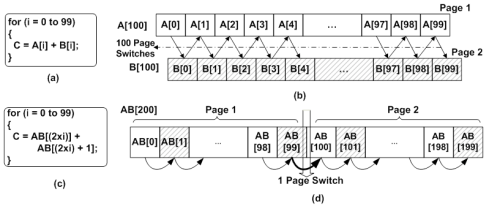


Figure 5. Array Interleaving through example: (a)Example loop (b)Array allocation and access pattern (c)Loop block with interleaved arrays (d)Array allocation and access pattern of interleaved array

6.1. Motivation

Figure 5 shows how array interleaving can reduce the TLB page switching over data accesses in the program. The code in Figure 5(a) shows a loop which accesses elements from two different arrays A and B, which are mapped to different pages. Figure 5 (b), shows that when this loop executes, there is a page switch between consecutive memory accesses in the program. Figure 5(c) shows the transformed code after interleaving. Array interleaving places the elements of the two arrays as alternate elements of the array AB. Figure 5(d) shows that there is no page-switching between consecutive access to AB.

6.2. Which Arrays to Interleave

The problem of reducing TLB page switching is localized to consecutive memory accesses, therefore interleaving of arrays

need only be directed to decrease the page switching in the innermost loop. Consider a nested loop of 3 levels, whose iterators are i, j, and k, in which are there are references to arrays A and B. Suppose in the innermost loop, the reference functions are affine functions of the iterators, i.e., the access function can be represented as a linear combination of the iterators, $f_A = a_0 + a_1 i + a_2 j + a_3 k$, and similarly $f_B = b_0 + b_1 i + b_2 j + b_3 k$.

We consider two arrays A and B as interleaving candidates only if i)the access functions of the arrays are the same. Thus, $a_0 = b_0, a_1 = b_1, a_2 = b_2, a_3 = b_3$ ensuring minimized page switches after interleaving. ii)the arrays of the same size. For example, we will interleave an array of integers with another same size array of integers. It is important to note that while it is possible to interleave arrays with slightly different access patterns also, it results in an overhead in terms of extra addressing instructions. However, the innermost loop may contain several references to the same array. Two arrays will be interleaving candidates if the conditions are satisfied for any pair of references to the arrays. We perform this analysis on all the important loops of the application, and find pair of arrays, which are interleaving candidates, we take the union of interleaving candidates. Thus if arrays A and B are found to be interleaving candidates from one loop, while B and C are interleaving candidates from some other, then all the three arrays will be interleaved.

6.3. Interleaving

The process of interleaving r arrays of the same data type $A_1, A_2, ...A_r$ is a three step transformation. The first is to replace the individual array declarations with a single array A of r times the size of each array, and second is to fix the access functions of all the array references. The access function $f_m = A_m[a_m i + b_m j + c_m k + d_m]$ of the m^{th} array is replaced by $f_m = A[r \times (a_m i + b_m j + c_m k + d_m) + (m-1)]$ in three-level nested loop. At the end of the day, it is important to schedule the instructions that access the interleaved array in the same pattern consecutively. This is done by moving the result of the first instruction in a new temporary variable, and replacing all its uses by the temporary variable. Interleaving of r arrays of different data types is done by declaring a new structure, say s, which contains an element from each of the arrays. We then declare an array A of the same size as all the previous arrays consisting of elements of data type s. Then we replace the access function of the m^{th} array $f_m = A_m[a_m i + b_m j + c_m k + d_m]$ by $f_m = A[a_m i + b_m j + c_m k + d_m].m$.

6.4. Experiments

We translate the source code into the FORAY format [2], which essentially consists of just the loop structure and the array access functions as affine functions of the loop iterators. We analyze the code in this format, and, perform our page-aware array interleaving transformations in this format, and then convert it back to the source code. The application is compiled again, and our instruction scheduling for page switch minimization is applied to enhance the impact of array interleaving. Figure 6 plots the page switch count after performing array interleaving and instruction scheduling on all the benchmarks. The plot thus shows that our page-aware array interleaving is a very effective transformation, and reduces the data-TLB page-swith count by an

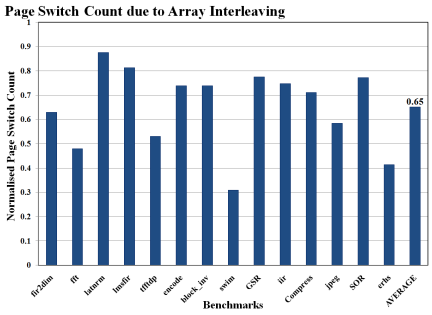


Figure 6. Impact of Array Interleaving and Instruction Scheduling on Page Switch Count

average of 35% (indicated by the right-most bar) with an overall average of 11% increase in performance. This performance improvement is inherent to array interleaving, as it inherently increases the spatial locality of data, leading to improved cache behavior. In *swim*, two global arrays and 5 local arrays were accessed together in the loop bodies. Interleaving was possible on all the arrays, thereby forming two interleaved arrays (one global, and other local). This transformation enhanced the opportunities for instruction scheduling and therefore 70% page switch reduction was observed. Since the TLB power is directly proportional to the number of accesses, we can expect a concomitant 35% reduction in TLB power due to the combined impact of array interleaving and page switch-aware scheduling.

7. Impact of Loop Unrolling

Loop unrolling is a loop transformation in which the loop body is replicated a finite number of times, thereby reducing the loop overhead instructions. It is important to observe that loop unrolling by itself does not reduce TLB page switching, but, it may increase the effectiveness of instruction scheduling, by providing more opportunities to schedule instructions and thereby reduce inter-instruction page switching.

Unrolling a loop may reduce page switches if there is atleast one instruction, such that if we schedule two copies of the instruction belonging to different iterations when scheduled consecutively, will not result in inter-instruction page switching. In other words, loop unrolling can be performed if $\exists i \in I$ such that,

$$\begin{cases} min \begin{cases} PS_O(undefined, i.d, i.s1) \\ PS_O(undefined, i.d, i.s2) \end{cases} & \text{if } i.op \text{ is comm} \\ PS_O(undefined, i.d, i.s1) & \text{otherwise} \end{cases} = 0$$

We have implemented our page-switch aware loop unrolling transformation also as source code transformation. Figure 7 plots the effect of loop unrolling on the page switch count of various benchmark applications. The normalized page-switch count for the case when page-switch aware instruction scheduling and array interleaving are performed is plotted as the dark bar (to the left for each benchmark), and the lighter graphs indicate the page-switch count for unrolling factors of $2, 4$ and 8 times respectively. The right-most set of bars in Figure 7 indicate the average values for the cases plotted. On an average,

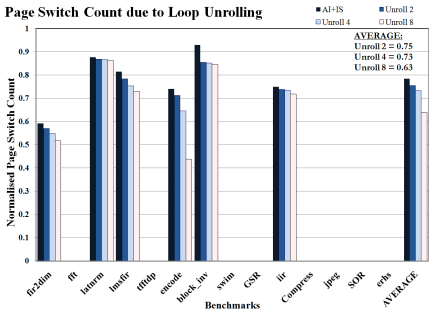


Figure 7. Impact of Loop Unrolling on Page Switch Count

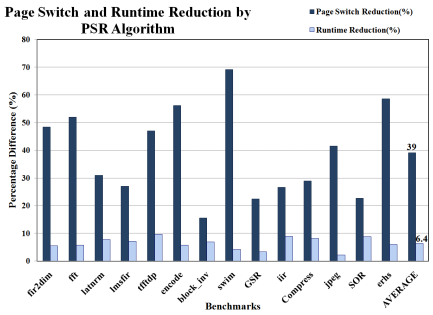


Figure 8. Page Switch Count and Runtime reduction by our Page-Switch Reduction Algorithm

for an unrolling factor of 8, we obtain a reduction of 37% in the page switch count for the applications on which page-aware loop unrolling was possible with 9% performance improvement.

8. Comprehensive Page Switch Reduction

Finally we study the impact of all the three transformations together. The ordering of the transformations is an interesting issue. Instruction scheduling and array interleaving are the fundamental transformations that reduce data TLB page switches. Loop unrolling will be most effective when all the opportunities for page switch reduction achievable after re-scheduling, are exploited. Our page switch-aware instruction scheduling is done at a more fine-grained level, and therefore has to be performed only after array interleaving and unrolling to maximize the effect. We first perform *Page-Switch Aware Array Interleaving* to group the memory allocation of varied arrays together into one overlapped page, and then *Loop Unrolling* on the instructions such that all the instructions capable of being implemented without page-switch are executed together. Our fine-grain instruction scheduling is then performed as a post-pass.

The dark bars on the left in Figure 8 plot the percentage reduction in the data TLB page switch count for each application. The reduction is calculated as compared to the data TLB page switch count when the application is compiled using $GCC - O3$ alone. The rightmost dark bar shows that there is an average 39% data TLB page switch count reduction over all the benchmarks. The light bars on the right in Figure 8 plot the reduction in runtime for all the applications. The rightmost light bar shows that there is an average 6.4% reduction in runtime. In conclusion, the effect of page switch reduction techniques is additive, and the effect is realized after each step of the *Page Switch Reduction* algorithm.

9 Summary

The *Use-Last* TLB architecture proposed in [9] reduces the TLB power consumption, if the same page is accessed successively. This approach was ineffective for data TLB, because data accesses do not exhibit high locality as compared to instructions. In this paper, we have introduced a novel, *page-aware instruction scheduling* algorithm, and proposed heuristics to decide when to perform array interleaving, and loop unrolling to reduce the TLB page switching. Our experiments on benchmarks from MiBench, Multimedia, DSPStone and BDTI suites show a 39% reduction in the TLB page switches with a negligible increase in performance, over what is possible by the GCC compiler. Since the dynamic power of the TLB is directly proportional to the number of page switches in the *Use-Last* TLB architecture, we can expect a concomitant 39% reduction in the TLB power. Our future work is to investigate the impact of other transformations, e.g., instruction selection on TLB power reduction.

References

[1] A. Shrivastava et al., "Operation tables for scheduling in the presence of incomplete bypassing," *In CODES+ISSS*, pages 194199, 2004.

[2] I. Issenin et al., "FORAY-GEN: Automatic Generation of Affine Functions for Memory Optimizations," *In DATE 05*, pages 808813, 2005.

[3] M. R. Guthaus et al., "MiBench: A free, commercially representative embedded benchmark suite," *In WWC 01*, pages 314, 2001.

[4] V. Zivojnovic et al., "DSPstone: A DSP-oriented benchmarking methodology," *In Proceedings of Signal Processing Applications and Technology*, Dallas, 1994.

[5] J. L. Henning, "SPEC CPU2000: Measuring CPU Performance in the New Millennium," *Computer*, 33(7):2835, 2000.

[6] H. Balakrishnan et al., " Multimedia Benchmarks: A Performance Comparison of Multimedia Programs on Different Architectures" *citeseer.ist.psu.edu/233784.html*

[7] BDTI Suite: Berkeley Design Technology Inc. "The BDTI Benchmark suites", *bdti.com/products/benchmark_overview.html*

[8] Intel Corporation, "Intel XScale®Technology Overview", *intel.com/design/intelxscale*

[9] J.R.Haigh et al., "A Low-Power 2.5 GHz 90 nm Level 1 Cache and Memory Management Unit," *In IEEE Journal of Solid State Circuits*, pages 11901199. IEEE Press, 2004.

[10] L. T. Clark et al., "Reducing translation lookaside buffer active power," *In ISLPED 03*, pages 1013, 2003.

[11] M. Ekman et al., "TLB and snoop energy reduction using virtual caches in low-power chip-multiprocessors," *In ISLPED 02*, pages 243246,2002.

[12] X. Zhou et al., "Low-power cache organization through selective tag translation for embedded processors with virtual memory support," *In GLSVLSI 06*, pages 398403, 2004.

[13] M. Kandemir et al., "Compiler-Directed Code Restructuring for Reducing Data TLB Energy," *In CODES+ISSS 04*, pages 98103, 2004.

[14] I. Kadayif et al., "Optimizing instruction TLB energy using software and hardware techniques," *ACM Trans. Des. Autom. Electron. Syst.* 10(2):229257, 2005.

[15] P. Petrov et al., "Energy-effcient physically tagged caches for embedded processors with virtual memory," *In DAC 05*, pages 1722, 2005.

[16] T. Austin. "SimpleScalar LLC". *simplescalar.com*

[17] V. Delaluz et al., "Compiler-directed array interleaving for reducing energy in multi-bank memories," *In ASP-DAC 02*, page 288, 2002.

[18] A. Parikh et al, "Instruction scheduling for low power," *The Journal of VLSI Signal Processing*, 37(1):129149, 2004.

[19] A. Chiyonobu et al., "Energy-efficient instruction scheduling utilizing cache miss information," *SIGARCH Comput. Archit. News*, 34(1):6570, 2006.

Simultaneous Peak Temperature and Average Power Minimization during Behavioral Synthesis

Vyas Krishnan and Srinivas Katkoori
Department of Computer Science & Engineering, University of South Florida, Tampa, USA
{krishnan, katkoori}@cse.usf.edu

Abstract

With continuous CMOS scaling, and increasing operating frequencies, power and thermal concerns have become critical design issues in current and future high-performance integrated circuits. Elevated chip temperatures adversely impact circuit performance and reliability. On-chip thermal gradients can lead to unpredictable clock skew variations and timing failures. Chip temperatures are influenced by design decisions at the behavioral and physical-synthesis levels. Existing low-power design techniques cannot adequately address thermal issues since their optimization objectives fail to capture the spatial nature of on-chip thermal gradients. We present an algorithm for thermally-aware low-power behavioral synthesis that concurrently minimizes average power and peak chip temperature. Our algorithm uses accurate floorplan-based temperature estimates to guide behavioral synthesis. Compared to traditional low-power synthesis, our method reduces peak temperatures by as much as 23%, with less than 10% overhead in chip area.

1. Introduction and Motivation

Steady scaling of CMOS process technologies over the past three decades have enabled feature sizes to shrink continuously, allowing the integration of millions of transistors in modern VLSI chips. However, with decreasing feature sizes and increasing transistor counts, power density in VLSI circuits has increased dramatically. Since the heat generated by a VLSI circuit is proportional to its power density, the corresponding rise in on-chip temperatures adversely impacts reliability, circuit performance, and cooling costs. According to ITRS [1], thermal management is projected as one of the most challenging issues in the design of future high-performance integrated circuits. Power-aware design alone fails to adequately address thermal issues, thus creating a need for *temperature-aware* low-power synthesis.

Different functional units in a circuit can have different switching activity rates, leading to uneven power dissipation among the various logic blocks on the chip. Due to low thermal conductivity of silicon, the rate of lateral heat propagation in a chip is slow, causing localized heating to occur much faster than chip-wide heating. This can cause uneven temperature distribution on the chip, creating on-chip thermal gradients, and "thermal hot spots" due localized areas of high power densities.

Traditional low-power behavioral synthesis techniques do not take on-chip thermal distributions into account during synthesis. With power densities and thermal gradients projected to increase rapidly in future technologies [1][2], there is a need for temperature-aware low-power design techniques that directly target the spatial nature of on-chip temperature distributions. This makes temperature-awareness an essential part of low-power behavioral synthesis.

Elevated temperatures have several adverse effects on a circuit. Higher temperatures adversely impact performance due to decreased transistor switching speeds and increased wire resistance, which can lead to timing violations [2]-[4]. Leakage power in CMOS circuits increases exponentially with temperature[5]. This coupling between static power and temperature can lead to thermal runways if not adequately managed [2]. Higher temperatures also cause reliability problems due to electro-migration in wires, and thermal-stress related gate-oxide breakdown in transistors [4]. In addition, chip packaging costs increase with higher thermal budgets.

Temperature-aware high-level synthesis is challenging because of the multi-objective nature of he problem, with several conflicting objectives – throughput rate, power, peak temperature, and chip area. Figure 1 illustrates the different factors that affect power dissipation and temperature of an integrated circuit. Task scheduling and resource binding in high-

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.78

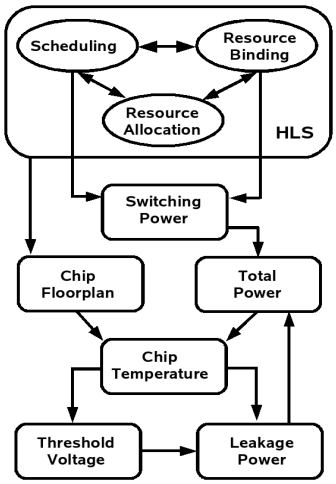


Figure 1. Factors affecting total power and on-chip temperature during behavioral synthesis

level synthesis significantly impacts the static and dynamic power of datapath functional units, and hence their temperature. An increase in temperature increases sub-threshold leakage, leading to further increases in temperature.

Floorplanning also significantly affects on-chip thermal distribution. The temperature of a functional unit is determined not only by its power density but also by the temperature of other functional units in its vicinity. Changing functional unit positions on a floorplan to balance power density and thereby reduce chip temperatures may increase chip area. Conversely, placing highly connected functional units bound to timing critical tasks, close to each other to minimize wire delay and power, could lead to localized areas of high chip temperatures, especially if the modules also have high switching activity.

An effective temperature-aware design technique must tightly couple floorplanning and high-level synthesis, with some form of feedback from thermal analysis. In this paper, we propose an integrated approach to temperature-aware high-level synthesis that simultaneously performs temperature-aware scheduling, binding, and floorplanning, using feedback from an accurate and fast thermal simulation, to guide synthesis decisions.

To perform a multi-objective optimization of delay, chip area, power dissipation, and peak temperature, we use a *two-stage* simulated annealing algorithm, that uses different optimization objectives in each stage. In particular, the first stage of the algorithm uses a fast annealing strategy to optimize for power, throughput, and chip area, and provide a good starting point for the second stage that improves on this solution to optimize average power dissipation while minimizing peak chip temperature. To estimate the on-chip thermal distributions, a thermal analysis of the floorplan is performed. Using a multi-stage annealing approach significantly reduces synthesis run-times by avoiding expensive thermal analysis during early stages of search, when solutions are far from optimal. We use ISAC [7], an accurate architectural-level thermal modeling tool to estimate on-chip thermal distributions and module temperatures. ISAC uses the power traces of functional units and their floorplan locations, to compute module temperatures. The use of thermal modeling during design space exploration allows weeding out potential designs that have a high probability of encountering thermal problems.

The remainder of the paper is organized as follows. Section 2 gives an overview of related work. In Section 3, we discuss our proposed temperature-aware low-power high-level synthesis approach. Section 4 presents our experimental results, and Section 5 concludes the paper.

2. Related Work

Several researchers have addressed the problem of thermal modeling and temperature-aware design. Micro-architecture level thermal models were proposed in [8]. The authors of [8, 9] consider thermal effects during processor-based micro-architectural design space exploration. Thermal issues have also been considered during physical synthesis [10]-[12].

A number of behavioral synthesis works have appeared in the literature addressing average power during datapath synthesis. However, there are few research works addressing thermal issues during high-level synthesis. In high-level synthesis, researchers have considered thermal effects during resource binding [13]-[15] and scheduling [16]. Gu *et al*., [17] use floorplanning and voltage islands to alleviate thermal-hotspot formation during high-level synthesis.

Our work differs from these in several respects. Our approach tightly integrates high-level and physical-level synthesis steps at all stages of synthesis, to provide accurate estimates of thermal distribution. The works in [13] and [14] do not use floorplan-level information in their temperature-minimization algorithm, and therefore does not account for lateral heat flow between modules. The authors of [15] separate the floorplanning and binding steps during thermal optimization. The authors of [16] use a series of scheduling and binding moves in an attempt to minimize temperature, without changing the floorplan.

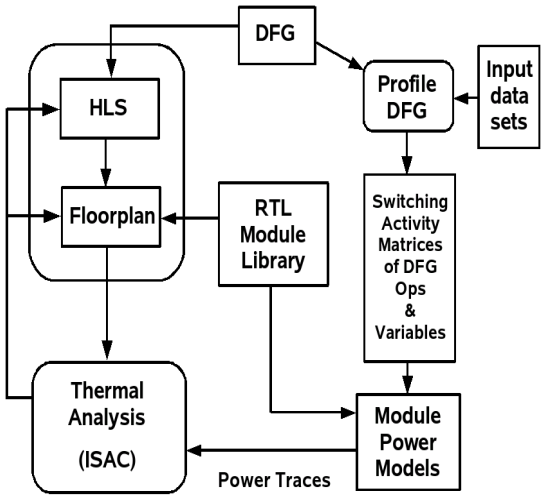


Figure 2. Temperature-aware behavioral synthesis algorithm

3. Temperature-Aware Low-Power High-Level Synthesis

In this section, we give an overview of the proposed technique, a resource-constrained high-level synthesis (HLS) algorithm that considers the impact of task scheduling, resource binding, and floorplanning, on average power and on-chip temperature distribution. Our low-power synthesis method uses a two-stage simulated annealing algorithm (SA) to concurrently perform the tasks of scheduling, resource binding, floorplanning, and thermal analysis, yielding solutions that have lower peak module temperatures than traditional temperature-unaware low-power high-level synthesis techniques.

Figure 2 illustrates the main steps used in our approach. First, the input dataflow graph (DFG), is simulated with typical input traces to create input switching power tables for each resource type in the target datapath circuit. Task schedules, resource bindings, power models, an RTL design library, floorplanner, and a thermal model of the IC package are then used to evaluate the IC temperature profile, power, area, and performance of designs synthesized by our algorithm.

The main algorithm comprises of an HLS synthesis system tightly integrated with an incremental floorplanner and a thermal analysis tool. The inputs to the synthesis algorithm are (1) a behavioral description of the design in the form of a dataflow graph, (2) an RTL module library, and (3) profile-based input switching tables for the RTL modules modeling their switching power for different task and variable bindings. A simulated annealing based search algorithm is then used to concurrently perform task scheduling, resource binding, floorplanning, and module power minimization. Module temperature profiles are then determined through a thermal analysis of the solutions. We use ISAC [7], a fast and accurate temperature analysis tool to perform a static thermal analysis of solutions examined by our algorithm. ISAC takes a floorplan and module power traces as inputs, and computes the module temperatures. Our synthesis algorithm concurrently minimizes delay, power, area, and peak module temperatures.

3.1. Two-stage Simulated Annealing

To address the multi-objective nature of the temperature-aware low-power synthesis problem, we use a two-stage simulated annealing algorithm, where different optimization objectives, moves, and cooling rates are used in different stages of annealing. The objectives optimized in each annealing stage are:

- *Stage-I (high-temperature annealing)*: A *Layout-aware low-power high-level synthesis*, where the optimization objectives are power, schedule-length, and chip area.
- *Stage-II (low-temperature annealing)*: A *Temperature-aware layout-driven low-power synthesis*, where the optimization objectives are peak module temperature, power, and chip area.

In first stage of annealing, we perform a floorplan-driven low-power high-level synthesis, given the resource constraints for the datapath. The second stage is a *low-temperature* annealing stage, that takes the best low-power solution returned by the first stage, and uses a set of thermally-aware SA moves to minimize peak module temperature and create an even thermal distribution across the chip.

There are two main reasons for adopting a two-stage annealing strategy in our algorithm. First, our experiments indicate that for the problem addressed in this paper, using a single weighted-sum of all objectives during the search process often yields solutions far from the Pareto-optimal front. However, a two-stage annealing algorithm is able to sample the multi-objective solution space more efficiently, consistently yielding nearly optimal solutions. Secondly, use of a two-stage annealing approach significantly reduces run-times by avoiding expensive thermal analysis of solutions in earlier stages of search, when they are far from optimal.

3.2. Solution Encoding

The solution encoding used by our two-stage SA consists of two parts – (1) a set of DFG node-lists (or variable-lists) specifying the tasks (or variables) bound to each datapath resource (datapath unit or register), and (2) a sequence pair [18] (*S1*, *S2*) representing the relative location of datapath resources on a chip floorplan. SA neighborhood search moves that explore the HLS search space of task schedules and resource bindings perturb the node-list and variable-list portions of a solution encoding. A node-list associated with an RTL module represents node priorities for DFG tasks bound to the module. Node priorities are used by a modified list scheduling algorithm to schedule tasks in the DFG. The search space of chip floorplans is explored via SA moves defined on the sequence pair. Solution cost functions are evaluated by using algorithms described in Section 3.4.

3.3. Neighborhood Moves

Simulated Annealing moves are defined in both, the HLS search space (task schedules, module and register bindings), and the layout search space (chip floorplans).

3.3.1. Floorplan Moves. Five SA moves, similar to the ones defined in [6], are defined on the sequence pair representing a solution's floorplan:

- Rotate datapath module,
- Shift datapath module in S1 string,
- Shift datapath module in S2 string,
- Swap two datapath modules in S1 string,
- Swap two datapath modules in S1 string.

These moves are used in both stages of annealing used in our approach.

3.3.2. HLS Moves Used in Stage-I of the SA. Four SA moves are defined to operate on the node-lists associated respectively with computational units in the solution datapath. These moves are designed to explore different task schedules and bindings encoded by the node-lists. These moves are termed as *schedule-length variant* moves, because they could result in task schedules with differing schedule lengths. The main objective of these moves is to find a solution that minimizes the total switching power among datapath functional units and interconnects, while at the same time meeting the user-specified delay constraint on the schedule. The SA moves used are:

- Change a DFG task priority in a module node-list,
- Swap two DFG tasks in a module's node-list,
- Change a DFG tasks module binding,
- Swap the module bindings of two DFG tasks.

The first two move operations change a task's priority in the node-list used by the list scheduler, resulting in different schedules of possibly different schedule-lengths. The remaining two moves randomly change a DFG node's or variable's resource binding, with goal of minimizing interconnect and multiplexer costs.

3.3.3. HLS Moves Used in Stage-II of the SA. The schedule length of the best solution found by the two-stage SA is used as a constraint in Stage-II of annealing. The HLS moves defined in Stage-II are designed to explore different schedules and bindings, without changing the length of these schedules (*schedule length invariant* moves). Five neighborhood search moves are defined:

- Shift a task within its mobility range,
- Migrate a task to another compatible module,
- Swap module bindings of two compatible tasks,
- Migrate a variable to another compatible register,
- Swap register bindings of two compatible variables.

All these three moves are designed to preserve the schedule-length, while exploring a variety of schedules and bindings. The first of these moves changes the time-step when a task is executed, without changing its resource binding. The next two moves change the resource bindings of tasks, and also possibly the time-step when they are executed. The remaining two moves change the bindings of variables to registers. The main objective of Stage-II is to simultaneously minimize peak module temperature and average power.

3.3.4. Thermally aware Floorplan Moves. To improve on-chip temperature distributions, our algorithm also incorporates floorplan-level thermal optimization moves. In solutions where functional units with high power densities are physically close to each other, thermal hotspots can occur. To mitigate this situation, a thermal-aware swap operation is defined on the sequence pair of a floorplan. Under this operation, the functional units are sorted in order of increasing temperature, and the positions of a randomly chosen high-temperature functional unit in either the *S1* or *S2* sequence pair is exchanged with that of a low-temperature functional unit. This SA move allows more even thermal distribution over the chip, and prevents high-temperature modules from clustering in one area, leading to thermal hotspots. These moves are used during Stage-II of annealing.

3.4. Cost Function Evaluation

Figure 3 illustrates the main steps used to decode a solution encoding and evaluate the cost function. The

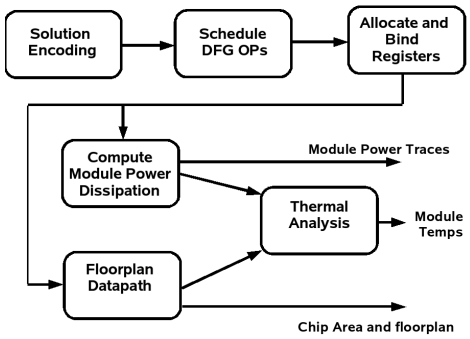


Figure 3. Solution cost function evaluation

node-list associated with each functional unit is used by a list scheduler to schedule tasks in the DFG. Once a feasible schedule is determined, we use the classical left-edge algorithm to allocate and bind registers in the datapath. The sequence-pair maintained by each solution encoding is used to pack the datapath units into a floorplan. The DFG schedule and the profiled switching probability data are used to compute module switching power. The power traces for functional unit, together with the chip floorplan, is then used by ISAC to perform a static thermal analysis to determine module temperatures and chip thermal profiles.

The cost functions used in Stages I and II differ in their objective functions. In Stage-I, we use the following cost function:

$$\alpha * D + \beta * P + \gamma * A$$

where, D is the normalized value of latency (schedule-length), P is the normalized value of average power, and A is the normalized chip area. The terms are normalized with respect to the best schedule-length, power, and chip-area values seen during search, which are dynamically updated during search. In our experiments, we set $\alpha = 250$, $\beta = 125$, and $\gamma = 125$. In Stage-II, where module temperatures of a low-power datapath are minimized, we use the following objective function:

$$\theta * T + \lambda * P + \varphi * A$$

Here, T refers to normalized peak module temperature in the datapath, while P and A have the same meaning as in Stage-I. In our experiments, we set $\theta = 250$, $\lambda = 125$, and $\varphi = 125$.

4. Experimental Results

The proposed algorithm was tested on a Linux-based workstation using a 1.86GHz Intel CoreDuo processor with 2GB memory. The overall flow used in our experiments is shown in Figure 2. Our experiments were performed on a set of several real-life large-sized benchmarks drawn from MediaBench suite [19]. Each of these benchmarks was specified as a DFG capturing the behavioral description of the architecture to be synthesized. The DFGs used in experiments range in size from 51 nodes to 333 nodes, and were obtained from [17]. The RTL resource set used in our experiments comprised of multipliers, ALUs, registers, and multiplexers synthesized using a TSMC 180nm technology library.

For our thermal analysis, we assume that each chip is attached to a copper heat sink using forced air cooling. Heat dissipates from the silicon die, through the cooling package to the ambient environment, and through the package to the printed circuit board. We assume an ambient temperature of 45^{o}C and a silicon thickness of 200μm.

In our experiments, the objective was to minimize the peak temperature among the functional units in a datapath during low-power behavioral synthesis. Our temperature-aware low-power synthesis method was compared with two other temperature-unaware low-power behavioral synthesis methods:

- Method-A: A low-power floorplan-aware synthesis methodology that minimizes average power,
- Method-B: A low-power floorplan-aware synthesis methodology that minimizes peak module power.

Method-A is an SA-based layout-driven low-power high-level synthesis algorithm that tightly integrates a floorplanner within the HLS synthesis loop. The optimization function used in Method-A minimizes the average power and throughput, together with the traditional floorplanning objectives of chip area and total wire length. Method-B is similar to Method-A except that it minimizes the peak power consumption of the datapath units, instead of average power. Comparing our algorithm with these low-power synthesis techniques allows us to highlight the advantages of a temperature-driven synthesis technique over a low-power design methodology. A thermal analysis is performed on the solutions found by these methods, and the peak module temperatures are compared with the solutions found by our technique. The intuition behind using Method-A is that lowering the total power dissipation in a circuit would hopefully lower overall on-chip power density and hence the on-chip temperatures. The intuition behind Method-B is

that by constraining peak module power dissipation, one could mitigate the formation of on-chip thermal-hotspots.

Table 1 compares the peak chip temperatures and average power using our temperature-aware technique and the low-power methods A and B. The peak temperature and average power values are averaged over ten independent SA runs using different random number seeds. For all the benchmarks, our technique found solutions with lower peak temperatures when

Table 1. Comparison of Peak Chip Temperature and Average Power

Benchmark		Proposed		Method-A		Method-B	
ID	Name	Max T (C)	Power (W)	Max T (C)	Power (W)	Max T (C)	Power (W)
1	h2v2-smooth_downsample	69.4	0.367	79.9	0.364	79.3	0.386
2	feedback_points	76.1	0.928	89.5	0.921	91.4	1.002
3	collapse_pyr	76.6	0.852	86.4	0.847	97.1	0.919
4	write_bmp_header	75.6	0.539	78.9	0.541	82.6	0.585
5	interpolate_aux	78.4	2.142	86.7	2.135	93.2	2.469
6	matrix_mult	80.7	2.270	89.8	2.263	94.7	2.355
7	idct_col	74.3	1.654	87.2	1.655	89.3	1.732
8	jpeg_idct_fast	72.4	1.370	91.3	1.371	92.6	1.459
9	jpeg_fdct_islow	79.8	1.415	95.8	1.418	97.1	1.544
10	smooth_color_z_triangle	71.2	1.022	88.6	1.019	91.9	1.114
11	invert_matrix	88.4	3.638	102	3.671	103.7	3.715
	Average	76.6	1.473	88.7	1.382	92.1	1.571

compared to a temperature-unaware low-power synthesis technique, highlighting the advantages of a temperature-aware technique over temperature-unaware low-power methods. The average peak chip temperature for the MediaBench benchmarks using our method was 76.6 C. Using Method-A, the average peak temperature was 88.7 C, while using Method-B resulted in an average peak temperature of 92.1 C.

Our temperature-aware synthesis methodology is able to achieve significant reductions in peak temperature over power minimizing methods A and B. Peak temperature improvements over Method-A averaged 13.5%, with a maximum improvement of up to of 20.7%. The average peak temperature improvement over Method-B was 16.7%, with a maximum improvement of 22.5%. Unless a synthesis methodology accounts for lateral thermal diffusion between modules on a floorplan, only minimizing peak module power or average power alone is inadequate in minimizing peak chip temperature.

The improvements in peak temperatures using our method are achieved with only a modest area overhead that averages to less than 10% for the tested benchmarks, with a peak area overhead of 12.6% for the largest benchmark. In addition, the difference between the average power of solutions found by our method and a traditional low-power synthesis method is less than 1% on average, for all the benchmarks tested.

5. Conclusions

Traditional low-power behavioral synthesis methods cannot sufficiently address non-uniform temperature distributions in an integrated circuit. Unless a synthesis methodology accounts for lateral thermal diffusion between modules on a floorplan, only minimizing peak module power or average power alone is inadequate in minimizing peak chip temperature. In this paper, we present an integrated temperature-aware high-level synthesis technique that tightly couples the HLS tasks of scheduling and binding, with physical-level estimates from an incremental floorplanner, to concurrently optimize power and peak chip temperature. Our experimental results indicate that compared to conventional power-minimization approaches to HLS, our synthesis technique reduces peak module temperatures by an average of 15%, with less than 1% difference in average power. These improvements in peak temperatures are achieved with less than 7% average increase in chip area over power-optimized designs.

6. References

[1] International Technology Roadmap for Semiconductors, http://public.itrs.net

[2] K. Banerjee, S.-C. Lin, and N. Srivastava, "Electrothermal engineering in the nanometer era: from devices and interconnects to circuits and systems", *ASP-DAC* 2006, pp. 223-230.

[3] A.H. Ajami et. al., "Analysis of non-uniform temperature-dependent interconnect performance in high-performance ICs," *DAC 2001*, pp. 567-572.

[4] M. Pedram and S. Nazarian, "Thermal modeling, analysis, and management in VLSI circuits: Principles and Methods," *Proc. IEEE*, August 2006, pp.1487-1501.

[5] K. Roy, S. Mukhopadhyay, and H. Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicron CMOS circuits," *Proc. IEEE,* 2003, vol. 91, pp.305-327.

[6] H. Murata et a., "VLSI module placement based on rectangular packing by sequence pair," IEEE Trans. CAD, Dec. 1996.

[7] Y. Yang, Z. Gu, C. Zhu, R.P. Dick, and Li Shang, "ISAC: Integrated space and time adaptive chip-package thermal analysis," *IEEE Trans. CAD*, vol 26, no.1, 2007.

[8] L. Skandron, K. Sankaranarayanan, S. Veluswamy, and D. Tarjan, "Temperature-aware microarchitecture modeling and implementation," *ACM TACO* 2004.

[9] M. Healy et al., "Microarchitectural floorplanning under performance and thermal tradeoff," *DATE* 2007.

[10] B. Goplen and S. Sapatnekar, "Efficient thermal placement of standard cells in 3D ICs using a force-directed approach," *ICCAD* 2003.

[11] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," *ICCAD* 2004.

[12] A. Gupta *et. al.*, "Thermal aware global routing for VLSI chips for enhanced reliability," *ISQED* 2008.

[13] R. Mukherjee, S.O.Memik, and G. Memik, "Temperature-aware resource allocation and binding in high-level synthesis," *DAC* 2005.

[14] R. Mukherjee, S.O.Memik, and G. Memik, "Peak temperature control and leakage reduction during binding in high-level synthesis," *ISLPED* 2005.

[15] P. Lim and T. Kim, "Thermal aware high-level synthesis based on network flow method," *CODES + ISSS* 2006.

[16] R. Mukherjee and S.O. Memik, "An integrated approach to thermal management in high-level synthesis," *IEEE Trans. VLSI*, November 2006.

[17] http://express.ece.ucsb.edu/benchmark/

Session 7A

Analog and Mixed Signal III

Low-Power Low-Voltage Analog Circuit Design using Hierarchical Particle Swarm Optimization

R. A. Thakker, M. Shojaei Baghini, M. B. Patil
Department of Electrical Engineering, Indian Institute of Technology, Bombay.
E-mail: rajesht@ee.iitb.ac.in, mshojaei@ee.iitb.ac.in, mbpatil@ee.iitb.ac.in

Abstract

This paper presents application and effectiveness of Hierarchical particle swarm optimization (HPSO) algorithm for automatic sizing of low-power analog circuits. For the purpose of comparison, circuits are also designed using PSO and Genetic Algorithm (GA). CMOS technologies from 0.35 µm down to 0.13 µm are used. PVT (process, voltage, temperature) variations are considered during the design of circuits. We show that HPSO algorithm converges to a better solution, compared to PSO and GA. For CMOS Miller OTA, even performance of the circuit designed by HPSO algorithm is better than the performance of recently reported manually designed circuit. For the first time, design of this OTA, in 0.4 V supply voltage, is also presented. For this new design, HPSO algorithm has taken 23.5 minutes of CPU time on a Sun system with 1.2 GHz processor and 8 GB RAM.

1. Introduction

The problem of analog circuit design and, in particular, MOS transistor circuit design has become very complex with the down-scaling of technology and with the increasing complexity of physical models. Various optimization techniques have been reported in the past and the recent times for automatic design of analog circuits. The gradient-based optimization methods [1] need to calculate derivatives and also require good initial guess for the design variables. In the absence of initial guess close to the globally optimum solution, these algorithms would generally stick at a locally optimum solution. Convex optimization techniques [2], which guarantee globally optimum solution, require a very good knowledge of circuit design and also of physical models to prepare constraints, which would be very difficult looking at the current state-of-the-art MOSFET models.

The evolutionary algorithms, which can be used to solve multimodal optimization problems to explore the solution space more efficiently, do not suffer from difficulties associated with the gradient-based and convex optimization methods. The Genetic Algorithm (GA), developed by Holland [3] with an inspiration from biological evolution, has been reported several times for automatic analog circuit design. Few recent citations are also found in [4]. Particle swarm optimization (PSO), proposed by Kennedy and Eberhart [5], has been observed to give better accuracy compared to GA in most of the applications. PSO algorithm was used for circuit design applications in [6]. This paper shows the effectiveness of hierarchical PSO (HPSO) [7], an extended version of PSO algorithm, for analog circuit design problem.

2. Analog Circuit Sizing Problem

In analog circuit design, after choosing a proper circuit configuration, values of circuit elements, called design variables, need to be determined to achieve the desired specifications. In case of MOS transistor circuits, generally these variables include the width (W) and length (L) of transistors, values of resistors, capacitors and inductors. Integrated analog circuit designers prefer to use technologies with longer minimum channel lengths as compared to digital circuits. However, technology scaling, high performance demands, and system-on-chip applications force analog modules to be implemented in the same or at most few technology nodes behind as that of digital circuits. The increased complexity of physical models and process variations with down-scaling of technology has made the manual design of analog circuits to be a challenging and time-consuming task. Therefore, efficient automatic design techniques are required.

The block diagram of the automatic analog circuit design, implemented in this work, is shown in Fig. 1. The optimizer module minimizes the error between desired specifications and simulator-returned performance measures using a suitable optimization algorithm (in this study, algorithm is: GA, PSO, or HPSO). The error function is defined as,

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.14

$$F_\varepsilon = \sqrt{\sum \left(\frac{Spec_{Desired} - Spec_{Sim}}{Spec_{Desired}} \right)^2}, \quad (1)$$

where $Spec_{Desired}$ represents desired specifications, and $Spec_{Sim}$ denotes the specifications returned by a circuit simulator for a particular solution provided by the optimizer. In each circuit design evolution, the specifications which satisfy the required criteria, will not contribute to the error function in Eq. (1).

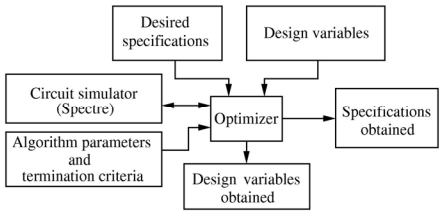


Fig. 1 The block diagram representation of automatic circuit optimizer.

3. Particle Swarm Optimization

In PSO algorithm [5], for a problem with n variables, $x_1, x_2, \ldots, x_n$, a population of N particles is initially generated by randomly assigning positions and velocities to each particle for each variable. In the case of analog circuit sizing, variables are the circuit design variables, e.g., sizes of transistors. If the position and velocity of the i^{th} particle are denoted by vectors $\mathbf{x}_i$ and $\mathbf{v}_i$ respectively, then

$$\mathbf{x}_i \equiv \left(x_i^1, x_i^2, \ldots, x_i^n\right), \text{ and } \mathbf{v}_i \equiv \left(v_i^1, v_i^2, \ldots, v_i^n\right) \quad (2)$$

Particles are moved towards the fittest particle and in this process; algorithm finds a better solution and is expected to reach the desired solution over time. Each particle keeps in memory the best position (denoted by $\bar{\mathbf{x}}_i$) it has attained during its trajectory. The velocity of a particle is updated on the basis of weighted addition of three vectors (Eq. 3), shown in Fig. 2(a): (i) particle's own velocity (**A**), (ii) the displacement of the particle from its past best position (**B**), (iii) the displacement of the particle from the globally best particle (**C**). The velocity update is given by,

$$\mathbf{v}_i(t+\Delta t) = w(t)\,\mathbf{v}_i(t) + p_1 r_1 (\bar{\mathbf{x}}_i - \mathbf{x}_i) + p_2 r_2 (\bar{\mathbf{x}}_g - \mathbf{x}_i), \quad (3)$$

$$\mathbf{x}_i(t+\Delta t) = \mathbf{x}_i(t) + \mathbf{v}_i(t+\Delta t)\Delta t, \quad (4)$$

$$w(t) = (w_i - w_f)\frac{(t_{\max} - t)}{t_{\max}} + w_f, \quad (5)$$

where i is the particle index, t is the current iteration number, Δt is equal to 1. r_1 and r_2 are random numbers uniformly distributed in the range [0, 1]. The parameter w is "inertia", and p_1 and p_2 are the acceleration coefficients. The $\bar{\mathbf{x}}_\mathbf{g}$ represents the best position attained by globally best particle. The velocities computed with Eq. (3) are used to move the particles as specified by Eq. (4). The commonly reported linear approach to update w parameter is used in this work and given in Eq. (5). w_i *and* w_f represent the initial and final values of w, respectively, and $t_{\max}$ is the maximum number of iterations.

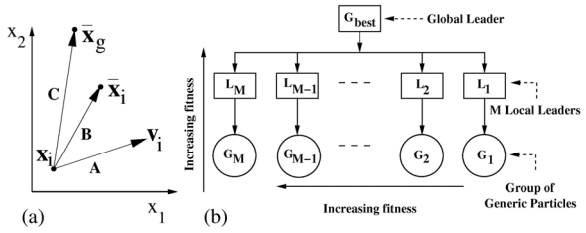


Fig. 2 (a) Two-dimensional representation of the components involved in velocity update of particles. (b) Arrangement of particles in HPSO algorithm.

In HPSO [7], population of N particles is first arranged in ascending order according to their fitness, with the globally best particle (the "global leader") at position N, as shown in Fig. 2(b). The next M best particles are designated as the "local leaders." The remaining (N-M-1) particles (the "generic particles") are divided into M groups. The M local leaders are assigned to the M groups. The generic particles follow their local leader and the local leaders follow the global leader. This feature enables enhanced exploration of the search space and shows better consistency in finding the optimum solution.

Algorithm parameters used in this work- For PSO and GA, N = 20. The crossover and mutation probabilities in GA are taken 0.8 and 0.2, respectively. For PSO, w_i = 0.9 and w_f = 0.4. For HPSO, w_i = 0.73 and w_f = 0.4 [7], and N = 21, M = 5. For both, PSO and HPSO, p_1 = p_2 = 1.49 [7].

4. Analog Circuit Examples and Results

In this section, we demonstrate the application of HPSO algorithm for automatic design of four analog circuits. The Cadence Spectre circuit simulator with BSIM3v3 MOSFET models is used to simulate circuits. To improve efficiency of the automatic design process, the simulator is asked to evaluate the fitness of a particular particle (chromosome), only if it has changed on at least one of the design variables by more than 1%. This restriction prevents unnecessary circuit

simulations. GA, PSO, and HPSO algorithms are compared with respect to two criteria.

- Quality of the solutions (measured by the error function F_ε given in Eq. (1)).
- Consistency of the algorithm in finding the best solution is measured by $\overline{F}_\varepsilon$ (average value of F_ε) over independent trials for each design problem.

In addition to the calculation of F_ε, the CPU time taken by each algorithm is recorded, and averaged (denoted by $\overline{T}_{CPU}$) over multiple design trials. The specifications during circuit evolutions are also recorded and normalized with the desired specification values for illustration purpose. All simulations are performed on a Sun system with 1.2 GHz dual core processor with 8 GB RAM.

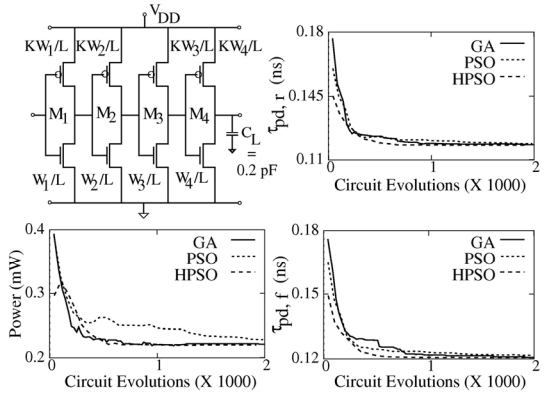


Fig. 3 CMOS buffer chain and plots of rise ($\tau_{pd,\ r}$) and fall ($\tau_{pd,\ f}$) propagation delay, and power dissipation versus circuit evolutions. One design solution obtained with HPSO is: W_1 = 0.13 µm, W_2 = 0.5 µm, W_3 = 1.7 µm, W_4 = 6.5 µm, and K = 2.0.

4.1 CMOS Buffer Chain

CMOS buffer chain (Fig. 3), commonly used for driving large capacitive loads, is designed using standard 0.13 µm CMOS process at V_{DD} = 1.2 V and for L = 0.13 µm. Input stage M_1 is chosen with minimum size. The width of remaining three NMOS transistors, and the ratio between the width of NMOS and PMOS transistors are considered as design variables. The buffer chain is designed using GA, PSO, and HPSO algorithms to minimize the rise ($\tau_{pd,\ r}$) and fall ($\tau_{pd,\ f}$) propagation delays, and power dissipation of the circuit. To examine the consistency of algorithms, five independent designs are carried out using each algorithm. Performance measures, evolved during the design cycles and averaged over five design trails, are plotted in Fig. 3. HPSO and GA algorithms are close in performance with HPSO algorithm slightly better than GA. PSO algorithm is not able to minimize the power dissipation as compared to HPSO and GA.

To evaluate the buffer chain designed by HPSO algorithm, the circuit is also sized manually using the logical effort theory (LET) with equal stage efforts assigned to each stage. It was observed that the results of HPSO algorithm not only match those of LET design, but also slightly better.

4.2 Two-stage CMOS Operational Amplifier

The two-stage CMOS op-amp, shown in Fig. 4 (a), is designed using 0.13 µm CMOS process at V_{DD} = 1.2 V, T = 27 ˚C, and for specifications given in Table 1. Op-amp specifications are of different orders in terms of values, which require different weights to be assigned to each of them during automatic design process. We have used constant weight approach and to ensure stability of the designed circuit, chromosomes in GA (or particles in PSO) are considered to be elite (or leaders in PSO), only if they have phase margin greater than 55˚.

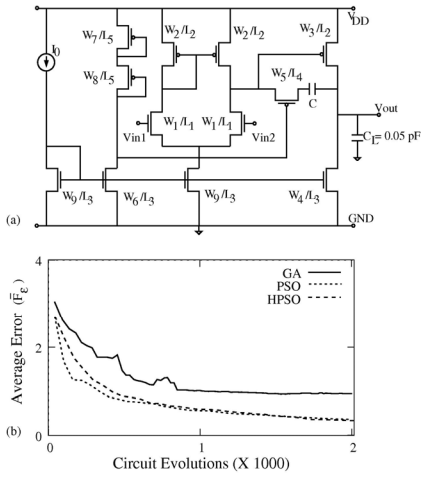


Fig. 4 (a) Two-stage CMOS operational amplifier and (b) Plot of average error $(\overline{F}_\varepsilon)$.

The design variables in this circuit are: W and L of transistors, the compensation capacitor, and the biasing current I_0. The range for each design variable is shown in Table 1. Systematic offset is taken into consideration during design cycle. Ten independent design trials are carried out. Five process corners taken into account during the design are: TT, FF, SS, FS, and SF, where 'T' stands for typical, 'S' for slow, and 'F' for fast. The plot of $\overline{F}_\varepsilon$ versus number of circuit evolutions is shown in Fig. 4(b). PSO and HPSO algorithms are close in performance; whereas the performance of GA is poor. Two (out of ten) design solutions obtained

with HPSO are given in Table 1. All of them are showing similar performance though sizes of transistors are different in some cases.

Table 1. Design solutions and specifications at TT process corner for two-stage op-amp designed by HPSO.

Design variables	Variable range	Solution 1	Solution 2
W_1/L_1	W: 0.5 to 10 (μm) L: 0.13 to 1 (μm) Transistor dimensions are in μm.	2.5/0.75	3.0/0.75
W_2/L_2		1.5/0.5	1.5/0.5
W_3/L_2		3.8/0.5	3.8/0.5
W_4/L_3		7.0/0.75	7.0/0.75
W_5/L_4		1.5/0.25	3.5/0.25
W_6/L_3		3.0/0.75	2.0/0.75
W_7/L_5		4.0/0.75	3.5/0.75
W_8/L_5		4.0/0.75	5.0/0.75
W_9/L_3		5.5/0.75	5.5/0.75
I_0 (μA)	0.01 to 10 (μA)	4.5	4.8
CF (pF)	0.1 fF to 10 pF	0.09	0.1
Desired specifications		**Specifications obtained**	
Gain ≥ 86 dB		86.16	86.13
Phase margin ≥ 65°		61.79	60.9
UGF ≥ 100 MHz		101	97
Power dissi. ≤ 20 μW		21	21.2
Rise slew rate ≥ 40 V/μs		50.33	48
Fall slew rate ≥ 40 V/μs		37.79	38

4.3 Ultra-low-voltage CMOS Miller OTA

The third example is an ultra-low-voltage, bulk-driven, rail-to-rail CMOS Miller OTA; recently reported in [8] and shown in Fig. 5(a). We show automatic design of this circuit in 0.35 μm TSMC mixed-mode process and at V_{DD} = 0.6 V, which are the same process and V_{DD} level used in [8]. The circuit is designed for specifications given in Table 2, which are equivalent or better than that reported in [8].

To carry out the comparison with the results reported in [8], the process, supply, and temperature (PVT) variations are not taken into account during the design. The ranges of design variables are given in Table 2. The circuit is designed for ten independent design trials. The plot of $\overline{F}_\varepsilon$ is shown in Fig. 5(b). It can be seen that the HPSO algorithm is better in performance in comparison to PSO and GA.

The specifications and sizing of transistors reported in [8], and that obtained using HPSO algorithm (one out of ten solutions), are given in Table 2. The following observations can be made from the table for the circuit designed using HPSO algorithm in comparison to that reported in [8].

(a) The open loop gain is greater by 9 dB, which means it is superior by 2.8 times.
(b) The unity gain frequency is close to four times better.
(c) It has given better phase margin and slew rate.
(d) The performance in terms power dissipation and third component of THD is equivalent.
(e) The total transistor area is less by 80 %.

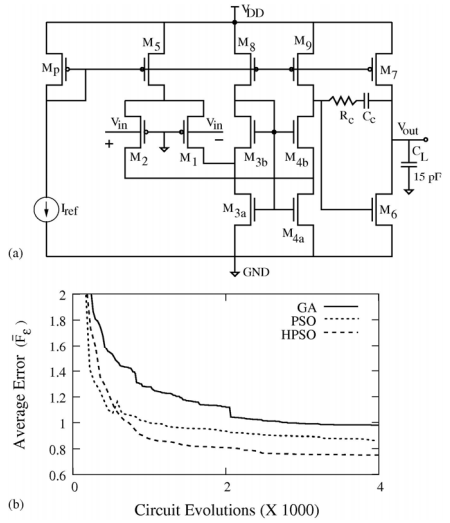


Fig. 5 (a) Ultra-low-voltage, ultra-low-power CMOS Miller OTA. (b) Plot of error $\left(\overline{F}_\varepsilon\right)$.

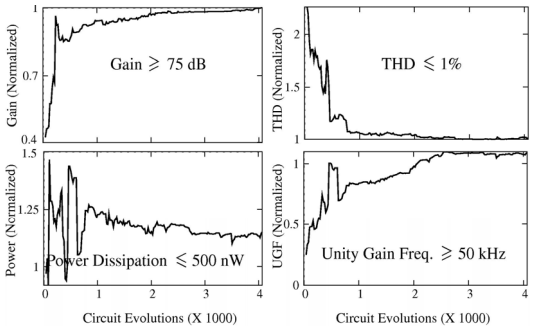


Fig. 6 Plot of specifications versus circuit evolutions for CMOS Miller OTA (Fig. 5(a)) designed by HPSO algorithm in 0.18 μm and V_{DD} = 0.4 V.

For future ultra-low-voltage applications, this circuit is also designed for standard 0.18 μm mixed-mode process at V_{DD} = 0.4 V using HPSO algorithm. The NMOS and PMOS threshold voltages are 403 and 440 mV, respectively. The desired specifications are set to the same values as used for 0.35 μm technology, and the circuit is designed for five independent design trials. The progress of performance measures (normalized) versus circuit evolutions averaged over

five independent design trails is shown in Fig. 6. Two design solutions with obtained specifications are shown in Table 3. The following major observations can be made about the circuit reuse in V_{DD} = 0.4 V (Table 3) compared to the circuit in V_{DD} = 0.6 V (Table 2):

(a) The open loop gain is reduced.
(b) The unity gain frequency is increased.
(c) The total transistor area is larger.

Table 2. Design solution and specifications reported in [8], and those obtained by HPSO algorithm for CMOS Miller OTA in 0.35 µm technology.

Design variables	Ref [8]	HPSO Algorithm	
		Variable Range	Solution
$(W/L)_{MP}$	200/9	W: 1 to 200 (µm)	34.5/9
$(W/L)_{M5}$	200/9		87/9
$(W/L)_{M1,}$	250/1	L for transistors is equal to the values reported in [8].	130/1
$(W/L)_{M8,}$	100/9		18/9
$(W/L)_{M3b,}$	400/1	Transistor dimensions are in µm.	86.5/1
$(W/L)_{M3a,}$	100/1		19.5/1
$(W/L)_{M7}$	800/9		104/9
$(W/L)_{M6}$	400/1		33/1
Rc (kΩ)	73.1	1 to 100 kΩ	81
Cc (pF)	5	1 to 10 pF	1.9
Iref (nA)	130	100 nA to 10 µA	109

Desired specifications	Specifications obtained	
	Ref [8]	HPSO
Open loop gain ≥ 75 dB	73.5	**82.67**
Phase margin ≥ 65˚	54.1˚	**58.01**˚
UGF ≥ 50 kHz	13.02	**48.61**
Power dissipation ≤ 500 nW	550	545.49
THD_3 ≤ 1% at 520 mV_{p-p}, 1 KHz	0.13 %	0.15 %
Rise slew rate ≥ 15 V/ms	14.7	**21.77**
Fall slew rate ≥ 15 V/ms	14.7	**23.11**
Total Trans. Area (µm²)	**14500**	**2858.5**
Total Area (µm²)	**20772**	**5586.92**

4.4 High-gain low-power three-stage Op-Amp

The circuit schematic of high-gain low-power three-stage CMOS op-amp, commonly used in transimpedance amplifiers and recently reported in a highly precise 1-V CMOS current reference generator [9], is shown in Fig. 7(a). A small modification was applied to the circuit to provide a proper topology for low-voltage design. For V_{bias}, we used $V_{DD}/2$. We also used drain of M_{22} to bias gate of M_{25} and M_{26}. This circuit is designed for 0.18 µm UMC technology. The process variations (TT, FF, SS, FS, SF), supply variations (± 10 % at 1.0 V), and temperature range 27-70 ˚C are taken into account. The desired specifications and the range of design variables are given in Table 4. To reduce the design time, the following steps are used in designing the circuit.

(a) Initially, the circuit is designed for TT process corner with supply variations considered and at T = 70 ˚C.
(b) The solution obtained in step (a) is refined to consider process and supply variations.
(c) The design solution obtained in step (b) is refined to consider PVT variations.

Table 3. Design solution and performance measures for CMOS Miller OTA in 0.18 µm technology.

Design variables		Solution 1	Solution 2
$(W/L)_{MP}$	Transistor dimensions are in µm.	110.5/9	105/9
$(W/L)_{M5}$		122.5/9	100/9
$(W/L)_{M1, M2}$		17/1	11.5/1
$(W/L)_{M8, M9}$		15/9	11.5/9
$(W/L)_{M3b, M4b}$		138/1	90/1
$(W/L)_{M3a, M4a}$		38.5/1	65/1
$(W/L)_{M7}$		194/9	143.5/9
$(W/L)_{M6}$		98/1	151/1
Rc (kΩ)		99	98
Cc (pF)		2.7	1.9
Iref (nA)		350	420
Desired specifications		**Specifications obtained**	
Open loop gain ≥ 75 dB		75	76.84
Phase margin ≥ 65˚		59.7˚	56.5˚
UGF ≥ 50 kHz		58	58.83
Power dissipation ≤ 500 nW		571	581
THD ≤ 1% at 400 mV_{p-p}, 1 KHz		1 %	1 %
Rise slew rate ≥ 15 V/ms		30.87	30.58
Fall slew rate ≥ 15 V/ms		16.4	21
Total Transistor Area (µm²)		4598	3827.5
Total Area (µm²)		7520.24	6046.19

For better clarity, the comparison between algorithms is carried out after completion of step (a) and is shown in Fig. 7(b). It can be seen that GA is quicker in reducing the error F_ε. But, HPSO is able to catch GA in terms of performance and is observed to keep on reducing F_ε. The plots of specifications averaged over five independent designs for HPSO algorithm after step (a) are shown in Fig. 8 and can be seen that they attained the specified values. One of the design solutions obtained with HPSO algorithm at the end of step (c), is shown in Table 4 along with the performance measures for the extreme supply voltages and temperatures at TT process corner.

The average CPU time ($\overline{T}_{CPU}$) taken by algorithms for the design of circuit examples considered in this work is given in Table 5. GA has taken slightly more CPU time, compared to PSO and HPSO.

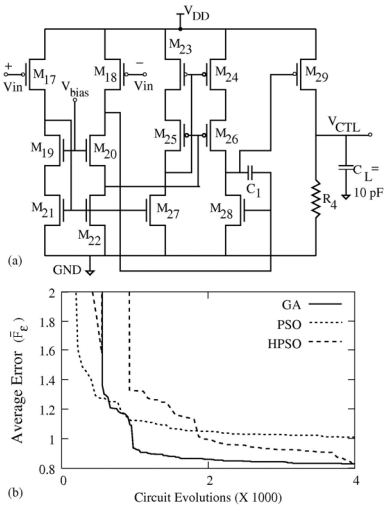


Fig. 7 (a) High-gain low-power three-stage op-amp (Vbias = V_{DD}/2). (b) Plot of error $(\overline{F}_{\varepsilon})$.

Table 4. Design solution and performance measures at TT process corner for three-stage CMOS op-amp.

Design variables	**Variable range**		**Solution**	
$(W/L)_{M17, M18}$	W: 1 to 50 (μm) L: 1 to 10 (μm) Transistor dimensions are in μm.		3.0/8.25	
$(W/L)_{M19, M20}$			12.0/7.5	
$(W/L)_{M21, M22}$			3.0/2.75	
$(W/L)_{M23, M24}$			3.0/2.25	
$(W/L)_{M25, M26}$			9.5/2.0	
$(W/L)_{M27, M28}$			15.5/5.5	
$(W/L)_{M29}$			50.5/1.0	
C_1 (pF)	0.001 to 10 pF		4.3	
R_4 (kΩ)	1 to 75 kΩ		59	
Desired specifications	**Specifications obtained**			
Temp	27° C		70° C	
V_{DD}	**1.1 V**	**0.9 V**	**1.1 V**	**0.9 V**
Gain ≥ 100 dB	119	114	118	114
Phase margin ≥ 65°	64	71	65	72
UGF ≥ 300 kHz	687	509	658	507
Power dissi. ≤ 10 μW	14	7	14	7.3
Systematic input offset voltage ≤ 50 μV	19	4.3	28.7	11.22

Table 5. The average CPU time $(\overline{T}_{CPU})$ taken for the design of circuits by algorithms.

Design Example	**GA**	**PSO**	**HPSO**
	h: hours, m: minutes, s: seconds		
Buffer Chain	4 m 21 s	3 m 54 s	**3 m 48 s**
Two-stage opamp	1 h 43 m	1 h 38 m	**1 h 37 m**
CMOS Miller OTA	26 m 28 s	**22 m 27 s**	23 m 4 s
Three-stage opamp	2 h 56 m	**2 h 41 m**	2 h 48 m

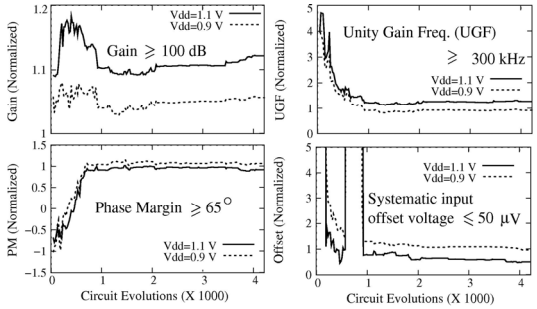


Fig. 8 Plots of specifications of three-stage op-amp designed by HPSO at TT process corner, T = 70 °C.

5. Conclusions

In conclusion, application of HPSO algorithm is demonstrated for automatic design of low-power low-voltage analog circuits. It is observed that in general, HPSO algorithm finds solutions with better repeatability, compared to GA and PSO. For ultra-low-voltage CMOS Miller OTA designed in 0.35 μm standard technology, the specifications of HPSO-designed circuit are significantly better than that of recently reported manual design. The OTA is also designed for the first time in 0.18 μm technology at V_{DD} = 0.4 V. For this new design, HPSO algorithm takes 23.5 minutes of CPU time.

6. References

[1] A. Savio, et al., "Automatic scaling procedures for analog design reuse," *IEEE Cir .and Sys.*-I, 2539-2547, Dec. 2006.

[2] M. Hershenson, et al., "Optimal design of a CMOS op-amp via geometric programming," *IEEE Trans. Comp. Aided Des. of Int. Cir. and Sys.*, vol. 20, pp. 1-21, Jan. 2001.

[3] D. Goldberg, "*Genetic Algorithm in Search, Optimization, and Machine Learning*," 1989.

[4] A. Somani, et al.., "An evolutionary algorithm-based approach to automated design of analog and RF circuits using adaptive normalized cost functions," *IEEE Trans. Evol. Comp.*, Vol. 11, 336-353, June 2007.

[5] J. Kennedy, et al., "Particle swarm optimization," *Proc. IEEE Int. Conf. on Neural Networks*, pp. 1942-1948, 1995.

[6] J. Park, et al.., "Parasitic-aware RF circuit design and optimization," *IEEE Cir. and Sys.*-I, 1953-1966, Octo. 2004.

[7] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE Trans. Sys., Man, and Cyber. – Part B:* vol. 35, 1272–1282, Dec. 2005.

[8] L. Ferreira, et al.., "An ultra-low-voltage ultra-low-power CMOS Miller OTA with rail-to-rail input/output swing," *IEEE Cir. and Sys.*-II: *Exp. Briefs*., 843-847, Octo. 2007.

[9] A. Bendali, et al., "1-V CMOS current reference with temperature and process compensation," *IEEE. Cir. and Sys.*-I., 1424-1429, July 2007.

Variation-Aware Macromodeling and Synthesis of Analog Circuits using Spline Center and Range Method and Dynamically Reduced Design Space

Shubhankar Basu , Balaji Kommineni and Ranga Vemuri
Electrical and Computer Engineering, University of Cincinnati
Cincinnati, OH 45221, USA
basusr,komminb,ranga@ece.uc.edu

Abstract

Manufacturing and process irregularities in nanometer technologies can degrade yield and severely slow down the design cycle time. Process variation aware methodologies can help in yield improvement and meeting time-to-market requirements for system-on-chip designs. Analog circuits are extremely sensitive to device mismatches and exhibit non-linear variations in their performance under the influence of manufacturing irregularities. Performance variation in blocks can lead to degraded system performance. In this work, we present a variation-aware performance macromodeling technique for analog building blocks that is fast and accurate and gurantees convergence during synthesis. The improvements in accuracy and time complexity of the macromodel generation process is achieved by constructing a target design region graph and dynamic reduction of the design space. The target design region also helps in reducing time during re-synthesis and achieving faster convergence. Experimental results demonstrate the accuracy of the macromodels and the reduction in synthesis time compared to spice based simulation-in-the-loop evaluations and static and adaptive sampling based techniques.

1. Introduction

Increase in the number of analog blocks in system-on-chip design, together with the faster time-to-market requirements has motivated the adoption of automated analog synthesis in the design flow. However, state-of-the-art commercial solutions still require computationally expensive numerical simulations-in-the-loop. Under the influence of random variations occuring due to manufacturing irregularities, conventional simulators require expensive Monte Carlo iterations to measure the effect of device mismatch. This makes the sythesis process prohibitively expensive.

Accurate performance macromodels allow designers to explore several design alternatives at negligible computation cost during the synthesis process. The effectiveness of performance macromodels is determined by the measure of their accuracy when compared to numerical simulators. Accuracy of macromodels in turn is driven by the choice of samples, dimensionality of the modeling problem and the choice of the modeling technique. While choice of sampling technique is an active area of research, it can be contextually improved through a combination of modeling technique, sample space selection and nature of dependent variables.

Analog systems are built using several lower level blocks like the operational amplifiers, filters etc. The variation in block performance under the influence of random process conditions can percolate up the hierarchy, thereby affecting the performance of systems like PLL, ADC etc. The use of hierarchical design approach starting with the development of variation-tolerant analog component blocks can reduce the overhead of mismatch in the system level design.

In this work, we propose a variation-aware performance macromodeling technique for analog component blocks like operational amplifiers. Our method employs a fast spline center and range interpolation technique for the performance functions, while preserving the accuracy of measuring process variation effect using limited number of numerical simulations in the inner loop of the data generation process. We propose a *dynamic design space reduction* scheme and a *target design region graph* construction during the macromodel generation steps. The target design region graph and dynamic design space reduction scheme progressively improves the accuracy of the macromodels while detecting the sub-space which can be used for repeated synthesis.

The remainder of the paper is organized as follows. Section 2 presents the background of the problem and discusses some of the related work in the domain. In section 3, we define the key concepts used in this work. We present our variation-aware macromodeling and synthesis methodology in section 4. Section 5 discusses the result on three well known benchmark circuits belonging to the operational amplifier class and finally Section 6 presents the conclusion and future work.

2. Background and Related Work

As semiconductor industry continues to adopt Moore's Law, it is challenged by the random and systematic defects introduced during the manufacturing process. In Fig. 1, we demonstrate the effect of such random process variation on a synthesized single-ended operational amplifier **(SEO)** circuit. The circuit is initially synthesized in 65nm technology, with the nominal values of process parameters. We subject the sized circuit to a random variation in four process parameters V_{th_0}, T_{ox}, L_{eff}, W_{eff} for 200 Monte Carlo iterations, following the 3σ normal distributions as obtained from [4]. The open-loop gain of the SEO is measured for the different values in the parameters mentioned above. It is observed from the histogram plot that around 59% of the performance points are below the nominally synthesized specifications (38dB) when subjected to random variations in device process parameters. The results demonstrated in Fig. 1 highlight the need for variation-aware modeling and variation-tolerant synthesis.

In recent times, several researchers have presented methods to model and optimize analog circuits in the presence of process variation. In [8, 7], the authors employ a low-rank projection scheme to approximate the process sample space. Implicit power iteration is used to compute the dominant Eigen vector for the low-rank formulation. However the trade-off obtained in time complexity and

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.51

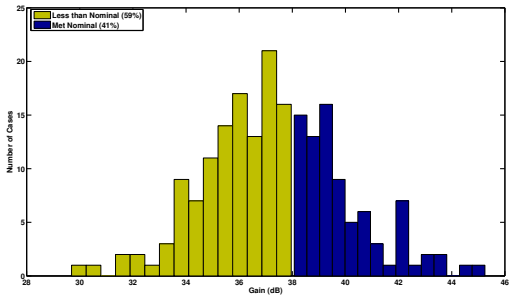


Figure 1. Gain for Single-Ended Opamp

accuracy with the use of rank-one projection may not be achievable with subsequent technology generations. Moreover, power iteration algorithm, though useful in applying to any large sparse matrix, is limited by its ability to find only one dominant Eigen vector and slow convergence if $\frac{\lambda_i}{\lambda_j} \approx 1$ [12].

In this work, we present an alternate technique to model the 3σ bounds in performance variation as a function of the design parameters. Design parameters are controllable by the user during repeated synthesis process for design centering. Our method captures the process variation effect on performance through limited number of Monte Carlo simulations in an internal sampling loop. The macromodels are used to perform variation tolerant circuit synthesis. In the next section, we present the definitions of the key concepts used in our work.

3. Definitions

3.1 Sizing Rules

It is argued that while a circuit may meet all high level performance constraints (gain, bandwidth, etc.), it may possess unwanted behavioral attributes and may be overly sensitive to device size variations [3]. Sizing rules impose matching constraints on sub-circuits such as differential pairs and current mirrors to avoid unwanted search of design parameters leading to electrically incorrect circuits. The sizing rules also help in reducing the number of free design variables, which help reducing the dimensionality of our modeling problem. In this work, we apply the sizing rules during the spice simulations to generate the raw sample data space.

3.2 Nominal Circuit

We define *nominal circuit* as the circuit synthesized with the nominal values of process parameters. Nominal values of the process parameters are the expected values of device components such as V_{th_0}, T_{ox}, L_{eff}, W_{eff} for the target technology. Performance of nominal circuits is expressed as $Perf(nominal)$ in this work .

3.3 Variation Tolerance

As illustrated in Fig. 1, performance of analog circuits deviates significantly from $Perf(nominal)$ under the influence of random process variations. The performance under process variation can vary between $[Perf_l, Perf_u]$ where l and u signify the statistically significant (3σ) lower and upper bounds of the performance swing. The range of the variation ($Perf_r$) is defined as the difference between $Perf_u$ and $Perf_l$ ($Perf_l - Perf_u$). Variation tolerance is characterized by two measures: a) $Perf_l \geq Perf(nominal)$ and b) $Perf_r \leq Perf(spec)_r$ where $Perf(spec)_r$ is the allowed variation in performance based on yield targets.

3.4 Variation Tolerant Circuit Synthesis

Given an unsized circuit topology and a set of performance specifications, variation tolerant circuit synthesis is the process of determining the unknown set of sizes of all devices in the topology such that the variation tolerance measures are satisfied. The synthesis algorithm is guided by a multi-objective cost function which is a weighted sum of the performance costs obtained from the evaluation of the corresponding macromodels. The goodness of candidate design points are judged by the cost estimates they generate. To accept the design solution, we attempt to minimize the cost, subject to the exit criterion for the synthesis process.

3.5 Variation Aware Macromodels

Macromodels are a black-box approximation of an unknown function and may be defined as follows [9]:

$$y = f(x_1, x_2, x_3, ...x_n) \quad (1)$$

$$\hat{y} = \hat{f}(x_1, x_2, x_3, ..., x_n) \quad (2)$$

In the above equations, x_i for $(0 \leq i \leq n)$ represent the independent input variables and y is the corresponding response based on those inputs. When, the input parameters are subjected to the process variation effects, the performance functions no longer remain a single value. If the noisy performances are equally probable to occur, and are continuous in the range, they can be represented by intervals. Therefore, for the real-valued input design parameters, under the influence of process variation, the variation aware macromodels can be defined as follows:

$$y_l, y_u = f(x_1, x_2, x_3, ...x_n) \quad (3)$$

$$\hat{y_l}, \hat{y_u} = \hat{f}(x_1, x_2, x_3, ..., x_n) \quad (4)$$

Here f is the original function with an unknown form and $\hat{f}$ is the approximation of f. In other words, $\hat{f}$ is the macromodel of f. The closer the value of $\hat{f}$ is to f, the more ready they are for use in the synthesis loop.

3.6 Duchon Pseudo-Cubic Splines

Analog performance is non-linear in the presence of process variation. This makes the use of low-order polynomial regression techniques an inadequate choice to model the performance. It has been argued [2] that the use of spline interpolation with an appropriate sampled space would be free of resonance effects due to Runge's Phenomenon which is often observed in high order polynomial regressions.

Duchon pseudo-cubic spline [5] works reliably on multi-variate scattered data. It uses a Radial Basis Function **(RBF)** which is the cube of the Euclidean norm and a polynomial kernel of order=1. Duchon pseudo-cubic spline has the following form:

$$Z_i = \sum_{j=1}^{k} W_j \phi(x_i - x_j) + P^m(x_i) \quad (5)$$

$$\phi = \|x\|_2^3 \quad (6)$$

The height (z_i) of the N-dimensional point to interpolate (x_i) is a weighted (W_j) summation of basis functions (ϕ) applied to the difference between the unknown point and all *'k'* number of sampled nearest neighbor points (x_j) currently defining the spline, plus a polynomial of degree $m = 1$. The RBF is evaluated between the points x_i and x_j. The coefficients together with the

location of the sampled data points completely define the spline interpolate.

Mathematically, the Duchon Spline Center and Range models required for our work are expressed by Equation. (7) and Equation. (8).

$$z_i^c = \sum_{j=1}^{k} W_j^c \phi(x_i^c - x_j^c) + P^m(x_i^c) \quad (7)$$

$$z_i^r = \sum_{j=1}^{k} W_j^r \phi(x_i^r - x_j^r) + P^m(x_i^r) \quad (8)$$

3.7 Target Design Region

Target Design Region **(TDR)** is a sub-space of the input n-dimenstional feasible Euclidean design space $\mathbb{R}^n$ within which designs satisfy the typical performances for correct functional and electrical operation. In a variation aware framework, we define the TDR as the region that contains designs with electrically correct operation and satisfying variation tolerant performance specifications for lower ($Perf_l$) and upper ($Perf_u$) bounds. Identifying the TDR for a circuit, given the feasible design space, can help in developing more accurate macromodels that are meaningfully used to synthesize the circuits in the presence of process variations.

4. Performance Modeling and Synthesis

4.1 Problem Formulation

In this work, we address two problems: a) Accurate variation-aware macromodeling of an analog circuit performance, and b) Fast synthesis in target design region using variation-aware macromodels that produce truly converging solutions. Both the problems are inter-related. Therefore, our method tries to tackle them at the same time. Formally, we define our problem as follows:

Let $\mathbb{R}^n$ be an n-dimensional design space and $\mathbf{X}$ denote the design variable set constrained by the lower and upper bounds $[X_l, X_u]$. Let $Perf(\mathbf{X})_l$ and $Perf(\mathbf{X})_u$ be the measure of the lower and upper bounds of the circuit's performance in the design space as obtained from spice simulation. Our objective is to build a macromodel for the circuit performance such that the macromodel estimates of the performance lower and upper bounds $[\hat{Perf}(\mathbf{X})_l$, $\hat{Perf}(\mathbf{X})_u]$ would be $\approx [Perf(\mathbf{X})_l, Perf(\mathbf{X})_u]$. Given a set of performance specification, denoted by the lower and upper bounds $[Perf(spec)_l, Perf(spec)_u]$, the objective of variation-tolerant synthesis is to find the minimum values of all the free design variables ($\mathbf{X}$), such that the target specifications for the circuit is met. In the following section, we present our methodology addresing the problems mentioned above.

4.2 Modeling and Synthesis

The accuracy driven performance macromodeling and synthesis methodology proposed in this work can be broadly divided into:

- Spline Center and Range Macromodeling
- Constrained Optimization
- Dynamic Reduction of Design Space (**DRDS**)

4.2.1 Spline Center and Range Macromodeling

The two major steps employed in this work for building the performance macromodels are: (a) Raw data generation and (b) Spline interpolation.

To perform raw data generation, we propose a two loop process.

- *outer loop*: Samples the design variables (e.g. W,L of transistors), which are input by a designer. A quasi-random sampling scheme using *Halton Sequence Generator* [6] is chosen to uniformly sample the design space.
- *inner loop*: Samples the device parameters like $T_{ox}, V_{th}, W_{eff}, L_{eff}$. Corresponding to each design sample point, the randomly varying process parameters are sampled through a spice Monte Carlo simulation using pseudo-random sample generator. This is performed using a *Simplified Performance Analyzer (SPA)* that calls the HSpice simulator in the loop.

Having obtained the raw data samples, the data is grouped into intervals based on each outer sample. The interval-valued performance data provides the benefit of having much less number of samples to capture the effect of process variation on performance. We reduce the dimensionality of the modeling problem using the reduced set of design variables obtained after imposing dc sizing rules as described in Section. 3.1

Center and range transformation can faithfully represent the characteristics of an interval data type, and eliminates unnecessary width expansion in classic intervals. Given the interval-valued performance data as an input, the transformation into 'center' and 'range' is done as a secondary step to prepare data for spline interpolation. The transformation is obtained as follows:

$$Input: X = [X_l, X_u] \quad (9)$$

$$Y = [Y_l, Y_u] \quad (10)$$

$$Output: X_c = (X_l + X_u)/2 \quad (11)$$

$$X_r = (X_u - X_l) \quad (12)$$

$$Y_c = (Y_l + Y_u)/2 \quad (13)$$

$$Y_r = (Y_u - Y_l) \quad (14)$$

Model interpolation is used to obtain the coefficients of the selected functional form. In this work, we use the Duchon pseudo-cubic splines modeling given by Equations 7 as the functional form for modeling. Algorithm. 1 presents the pseudo-code used in this work to develop the macromodels for the performance functions in each design space in the TDR graph.

4.2.2 Constrained Optimization

Simulated Annealing (SA) is used in this work to perform a constrained optimization of the circuit in the presence of process variations. In spite of their relatively longer time requirement, SA is capable of finding global minima in presence of several local minima. In our work, the time for optimization is controlled by a relatively smaller search space in the subsequent TDR nodes.

The constrained optimization problem can be expressed as:

$$\begin{gathered} \textit{Minimize} \sum_i x_i \\ such\ that \\ \sum_j \alpha_j * \hat{f}_j(\mathbf{X}) \leq \sum_j \alpha_j * f_j(\mathbf{X}) + \varepsilon \end{gathered}$$

α_j is the weight assigned to each performance function based on user specifications. ε is the allowed error tolerance during synthesis. Since the objective of the synthesis is to generate variation

Algorithm 1: Macromodeling

procedure getModel (Spice file, Config, **[W]** (TDR_node), N_s, N_i)
Input: Spice file, Config,**[W]** , N_s, N_i
Output: Performance Macromodel of the TDR_node
/* *Raw Data Generation* */
for $i \leq N_s$ **do**
  Y_i = SPA(w_i, N_i, Spice file, Config)
endfor
/* *SPA: Simplified Performance Analysis* */
pr_metrics = SPA(sim_data, f_p)
sim_data = HSPICE (spice netlist, $params_j$, monte($j \leftarrow 0$ **to** N_i))
$params_j$ = uniform($w_{ij}, v_{th_j}, t_{ox_j}, l_{eff_j}$)
$X_i = X_{i-1} \cup w_{ij}$
Y_i = $Z_{i-1} \cup$ pr_metrics
$X = X_i, \quad Y = Y_i$
/* *Performance macromodel generation* */
$[Y]$ = Interval_transform (Y)
$[Y_c, Y_r]$ = CR_transform ($[Y]$)
/* *SCRR: Spline based Center and Range Regression* */
β_c = SCRR(X, Y_c)
β_r = SCRR(X, Y_r)
TDR.Macromodel = {β_c, β_r }
$[a] = (a_l, a_u)$

tolerant circuits, we define the cost functions using both the *lower* and *upper* bounds of each performance. A typical performance cost in our context is defined as follows:

$$
\begin{aligned}
cost = & abs(\frac{(perf_u - perf(specs)_u)}{perf(spec)_u}), \\
& for\ perf_l \geq perf(spec)_l, \\
= & (\frac{(perf(spec)_l - perf_l)}{perf(spec)_l}) \\
& + abs(\frac{(perf_u - perf(specs)_u)}{perf(spec)_u}), \\
& for\ perf_l \leq perf(spec)_l.
\end{aligned}
$$

4.2.3 Dynamic Reduction of Design Space

The time required to solve for the co-efficients during spline interpolation grows in a cubic relation with the number of equations. Therefore spline modeling for more than a few thousand points can be computationally expensive. This limits its usage in practical situations. In Fig. 2, we illustrate the relation between Duchon Spline Center and Range **(SCR)** modeling time and the number of samples for the open-loop gain of the single-ended operational amplifier. It can be noted from the figure that the modeling time increases significantly beyond 900 samples. We overcome this problem by splitting the design space into narrow subsets and performing the interpolation on fewer numbers of samples required for desired accuracy over the smaller regions.

The accuracy of performance macromodels are improved dynamically based on the error in estimation of performance for the synthesized solution using our macromodels and spice Monte Carlo simulation. (**DRDS**) is maintained as a directed graph (Target Design Region Graph) whose nodes represent the subsequent target

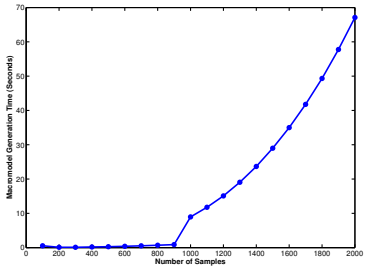


Figure 2. SCR Modeling Time Comparison

design regions (TDR) and the directed edges connect from a parent region to a child sub-region. Therefore, a TDR graph may have just one node (initial design space) or several nodes which are grown as child nodes to a parent node. Fig. 3 shows the typical flow in which DRDS and synthesis occurs in our proposed methodology. The various steps in the DRDS algorithm are explained below.

1. Given an input search space, *Initialize Macromodel* builds a Duchon spline based center and range macromodel of the circuit performance. *Icur* is used to store the current TDR node number. Therefore for the first time building of a macromodel and synthesis, the Icur is initially set to '1'. The initial macromodel is stored in the TDR graph data structure as *TDR(Icur).macromodel*.

2. Given the search space, topology and lower and upper bound performance specifications, the constrained optimizer finds a design **D** that satisfies the given specifications. In this step, the optimization engine uses the macromodels developed for performance evaluation.

3. Having obtained a design **D**, the macromodel accuracy is validated against numerical simulation under the influence of process varying device parameters to avoid a possible false convergence. The error during validation is computed as follows:

$$Error = \frac{|cost(D) - \widehat{cost(D)}|}{cost(D)} * 100 \qquad (15)$$

4. Based on the error in macromodel evaluation, three conditions are checked.
   (a) $Error \leq \varepsilon$? If *yes*, the optimization and DRDS terminates giving the optimized design point **D**. If *no*, the next condition is checked.
   (b) $Number\ of\ samples\ (N_s) <$ Maximum samples ? If *yes*, enhance the number of samples in the space and generate a new macromodel using *getModel*. If *no*, then the next condition is checked.
   (c) $Current\ space(Icur) <$ maximum number of nodes for the TDR? If *yes*, then split the search space. If *no*, then reset the search space to initial design space and re-initialize the macromodel.

5. Split the search space and generate new macromodel using *getModel*. Algorithm. 2 presents the pseudo-code adopted to split the parent search space into the next target design region. The essence of the algorithm is a simple condition check using the current design point and the design space for the TDR node (**[W]**) as follows: $D_i \geq \lfloor (w_{i_l} + w_{i_u})/2 \rfloor$.

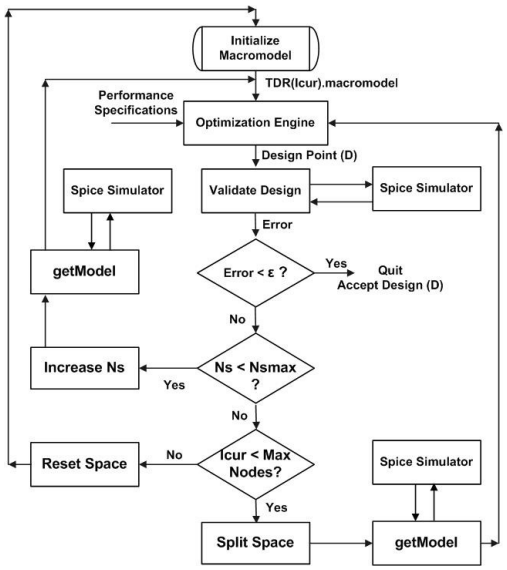


Figure 3. DRDS: Flow Chart

Algorithm 2: DRDS: Splitting Design Space

procedure DRDS (Spice file, Config, TDR(Icur), Icur, $[W_i(Icur)]$)
Input: TDR(Icur) Design Space ($[W_i]$), Best Design Sizes Obtained($[D_i]$)
Output: TDR(Icur).child
/ N = Total number of design variables */*
for *i = 1 to N* **do**
  $w_{i_{mid}} = \lfloor (w_{i_l} + w_{i_u})/2 \rfloor$
  if $D_i \geq w_{i_{mid}}$ **then**
    $w_{i_{new}} = [w_{i_{mid}}, w_{i_u}]$
  else
    $w_{i_{new}} = [w_{i_l}, w_{i_{mid}}]$
  endif
endfor
TDR(Icur).child = $[W_{i_{new}}]$

4.3 Repeated Synthesis Using TDR Graph

The time complexity of the variation-aware macromodeling and repeated synthesis process is considerably reduced by the use of an existing TDR graph. An existing ***Best Cost*** search in the subsequent child nodes of the TDR graph allows the optimization engine to explore the existing solutions in the TDR. This procedure has a three fold advantage: (a) The presence of a better solution in the existing child nodes can guide the synthesis process to the target design region faster. (b) The potential for simulated annealing to converge with a sub-optimal solution can be minimized by choice of an existing better solution. (c) The time due to model generation is overcome through the traversal of the existing TDR graph and using the pre-constructed macromodels of the corresponding nodes.

5. Results

We present the result of the proposed methodology on three circuits in the operational amplifier class: (a) Single Ended Opamp **(SEO)**, (b) Two Stage Opamp **(TSO)** and (c) Differential Opamp **(DOA)**. In our experiments, we consider the width of the transistors as the independent design variables that constitute the search space and hence the TDR nodes. The length of the transistors is kept at their minimum dimensions for simplicity reasons with no loss of generality. Sizing rules constraints, as decribed in Section. 3.1, have been applied to all the three circuits as a part of the spice netlist setup. We consider four process varying parameters $(T_{ox}, V_{th}, W_{eff}, L_{eff})$ for each transistors which are varied randomly following a Gaussian distribution. The circuits are implemented in 65nm predictive technology models [1]. The variations in process parameters are adopted from [4] for 70nm technology. All experiments are run using Matlab, C++ and Hspice for Windows operating system with $Intel^{®}\ CORE^{TM}2$ Quad processor with 8GB RAM.

Table 1. Comparison of different macromodeling techniques

Parameter	Static (5hr)		Adaptive (32hr)		DRDS (6hr)	
	Accuracy		Accuracy		Accuracy	
	r_l^2	r_u^2	r_l^2	r_u^2	r_l^2	r_u^2
BW	0.83	0.87	0.97	0.99	0.96	0.99
GAIN	0.95	0.94	0.95	0.93	0.99	0.99
PM	0.85	0.77	0.94	0.91	0.95	0.98
UGF	0.90	0.86	0.95	0.90	0.99	0.99

5.1 Error Minimization Using DRDS

Fig. 4 plots the error map for the modeling of open-loop gain for the single ended operational amplifier in the five TDR nodes using DRDS. It can be noted that the modeling error is reduced from 30% in the initial design space to 0.82% in node 5 which meets our desired error tolerance. The proposed method is therefore accurate within the final target design region.

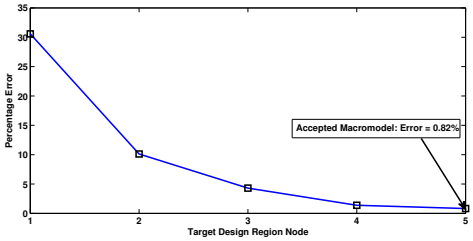


Figure 4. Error Reduction across TDR Nodes

5.2 Accuracy and Time Complexity

To illustrate the accuracy and time complexity advantage, we compare our technique (DRDS Macromodels) with the spline center and range macromodels generated using (a) static samples [11], and (b) adaptive samples on the complete design space [10]. We present the result of this comparison for the single ended operational amplifier circuit for four different AC performance parameters, viz. (a) 3-dB bandwidth (BW), (b) open-loop gain (Gain), (c) phase margin (PM) and (d) unity gain frequency (UGF). For the purpose of accuracy comparison, we generate a random validation set comprising of 100 test cases in the target design region, for which we accept the macromodels.

The accuracy of the macromodels are compared using the statistical correlation coefficient measure (r^2) for the lower and upper bounds of the performance measures. For a better accuracy, the value of r^2 should be close to 100% or 1. Table. 1 summarizes the result of this comparison. It can be observed from the table that DRDS macromodel is the most accurate amongst the three different techniques. It can also be observed that the model generation time for the DRDS macromodel is marginally higher than the static macromodel and is more that 5X efficient than the adaptive macromodel with comparable accuracy. Therefore, DRDS macromodeling has a better practical applicability over static and adaptive sampling based macromodeling techniques.

5.3 Repeated Synthesis

To further illustrate the advantage of the TDR graph during repeated synthesis, we synthesize the three circuits with varying specifications and compare the accuracy of result obtained with corresponding evaluation using HSpice circuit simulator. For convenience of understanding, we group the similar or closer specifications into a single specification category. It can be observed that during repeated synthesis (subsequent rows), the time complexity is reduced by several order of magnitude in most cases, while preserving the same accuracy level. The error in model estimation against HSpice based Monte Carlo simulation is computed using equation. 15. Table. 2 summarizes the measure of macromodeling time and model estimation percentage error for the synthesized design points. It can be observed that the error in all the cases is less than 1% which proves the accuracy of the DRDS macromodels used during synthesis.

Table 2. DRDS Modeling and Synthesis

SEO		TSO		DOA	
Time (secs)	Error	Time (secs)	Error	Time (secs)	Error
3088	0.81	11837	0.31	38456	0.35
21206	0.66	2725	0.71	15598	0.94
39	0.64	2147	0.34	3648	0.93
5367	0.00	14449	0.05	37297	0.94
34757	0.88	8746	0.15	20877	0.90
5868	0.92	2692	0.67	265	0.08
3128	0.88	5883	0.04	8734	0.68
22224	0.00	2315	0.23	1929	0.13
14357	0.00	50	0.13	268	0.79

5.4 Variation Tolerant Synthesis

We perform a Monte Carlo analysis using Hspice simulator with the synthesized sizes of the circuit to illustrate the variation-tolerance on performance. The performance variation data is compared with the corresponding nominal circuit performance. Table 3 summarizes the result obtained for two circuits, SEO and DOA. It is observed from the table that the synthesized circuits meet the bounds of the performance reliably.

6. Conclusion

In this work, we presented a variation-aware macromodeling methodology using target design region graph data structure and dynamic reduction of design region to build fast and accurate performance models for analog blocks. The macromodels are used effectively in the synthesis framework with no false convergence. The key point in the methodology is to find the target design regions in the design space and perform fast synthesis using the accurate macromodels in this space. The target design region graph is grown using dynamic reduction of design space guided by the error in the macromodel evaluation for the synthesized design point. The synthesized circuits achieve a very high performance yield with marginal area penalty. A library of variation tolerant analog blocks can be generated using this method. The next step is to use the block level robust library to build variation tolerant analog systems.

Table 3. Synthesized Circuit Performance

Perf	DOA		SEO	
	Nominal	Variation	Nominal	Variation
BW (MHz)	0.1	[0.1,0.4]	0.24	[0.2,0.3]
GAIN (dB)	40	[39.4, 42]	38.6	[38.2,46.3]
PM (Deg)	86	[86.3,89.3]	58	[58,73.9]
UGF (MHz)	1010	[998,1500]	17	[20,28.8]

7. References

[1] Predictive technology model. Technical report, http://www.eas.asu.edu/ ptm.

[2] B. Fornberg and J. Zuev. The runge phenomenon and spatially variable shape parameters in rbf interpolation. *Comput. Math. Appl.*, 54(3):379–398, 2007.

[3] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich. The sizing rules method for analog integrated circuit design. In *Computer Aided Design, 2001. IEEE/ACM International Conference on*, pages 343–349, 2001.

[4] ITRS. Semiconductor roadmap 2006. Technical report, International Technology Roadmap for Semiconductors.

[5] J. Duchon. *Constructive Theory of Functions of Several Variables, Lecture Notes in Mathematics*. Springer-Verlag, 1977.

[6] J.H.Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Nuremische Mathematik*, 2:84–90, 1960.

[7] X. Li, P. Gopalakrishnan, Y. Xu, and L. T. Pileggi. Robust analog/rf circuit design with projection-based posynomial modeling. In *ICCAD*, pages 855–862, 2004.

[8] X. Li, J. Le, L. T. Pileggi, and A. J. Strojwas. Projection-based performance modeling for inter/intra-die variations. In *ICCAD*, pages 721–727, 2005.

[9] R. Harjani et al. Oasys: A framework for analog circuit synthesis. *IEEE Transaction on Computer Aided Design*, 8(12):1247–1266, December 1989.

[10] Shubhankar Basu, Balaji Kommineni and Ranga Vemuri. Mismatch Aware Analog Performance Macromodeling Using Spline Center and Range Regression on Adaptive Samples. In *VLSID 2008*, pages 287–293, 2008.

[11] Shubhankar Basu, Balaji Kommineni and Ranga Vemuri. Variation Aware Spline Center and Range Modeling for Analog Circuit Performance. In *ISQED 2008*, pages 162–167, 2008.

[12] Zhaojun Bai et al. *Templates for the Solutions of Algebraic Eigen Value Problems: A Practical Guide*. SIAM, Philadelphia, 2000.

A Low Power Architecture to Extend the Tuning Range of a Quadrature Clock

R. Dutta and T. K Bhattacharyya
Electronics and Electrical Communication Engineering Department
IIT Kharagpur
Kharagpur, India
ramen_dutta@yahoo.co.in, tkb@ece.iitkgp.ernet.in

Abstract

A low power architecture to extend the frequency range of quadrature clock is proposed. This architecture is based on a series of dividers. It can enhance the lower frequency limit of a Quadrature Voltage Controlled Oscillator (QVCO) clock to any arbitrarily small frequency. Based on the architecture a design is shown which enhances the low frequency range up to -90% of the center frequency, assuming the QVCO tuning range is +20%. The dividers are made of Dynamic Transmission Gate Logic (DTGL) to reduce power consumption. Simulation result shows that the power consumption of the extender circuit, excluding the QVCO, is 2.1mW from 1.2V supply voltage at 3GHz input frequency in 90nm CMOS technology. The output jitter contribution by this circuit is 2ps and 0.15ps for mismatch and thermal noise respectively. Maximum output frequency achieved is 4.8GHz for differential clock and 2.4GHz for quadrature clock.

***Index Terms*—Tuning range, DTGL, quadrature clock**

1. Introduction

The target of a cognitive radio or Software Defined Radio (SDR) is to access a wide range of frequencies dynamically. Accessing wide frequency range is possible only when synthesizer in the transceiver can generate clock for all those frequencies. Therefore, the Voltage Controlled Oscillator (VCO) used in the frequency synthesizer of the transceiver chip has to cover a large frequency range along with a very good phase noise performance. In modern transceivers, VCO with a LC tank is used because of excellent phase noise though they provide narrow tuning range. To cover a wide tuning range multiple VCO architectures are proposed in [1], need large area and power. In this application, a frequency range of 200MHz to 2.4GHz is required which need three VCOs. The area and power increases hugely with number of VCO. Thus covering wide frequency range by means of extra VCO is not very efficient solution. **To the VCO tuning range can be increased using pseudo-exponential capacitor bank [2]. That can achieve maximum +/- 30% tuning range which is not sufficient for this application.Tuning range extension by a series of divider and multiplexer has also been reported in [1]. This needs two VCO as well to remove the frequency discontinuity between the VCO output and the divider-by-2 output.**

An approach to extend tuning range by a single VCO is reported in [3]. Here, a mixer is used to achieve a 3/2 and ¾ multiplication of the VCO output frequency. This method reduces the area and power requirement but gives rise to strong side bands at half of the VCO frequency. Also a quadrature mixer consumes much more power than a divider designed which can take very low power if designed in digital logic such as True Single Phase Clocking (TSPC) or Dynamic transmission Gate Logic (DTGL). Another scheme was proposed in [4] to extend the tuning range up to 50%, based on phase switching programmable divider architecture [5]. However this method also requires a mixer to produce a 2/3 multiplication of the input frequency with 50% output duty cycle.

In this paper we describe a technique to extend the VCO tuning range which consumes very low extra power, contributes minimum jitter to the output clock and also takes a very minimal extra area. The architecture is fully based on a series of dividers and a multiplexer. It maintains a 50% duty cycle at the output so that the quadrature clock can be generated just by another divider.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.88

After the introduction in Section I, Section II discussed about the proposed architectural and its optimization. Section III briefly describes about the design of different building blocks of this tuning range extension circuit. Simulation results are presented in Section IV and conclusions are drawn in Section V.

2. Architecture

Fig.1 shows the basic architecture of the proposed extender circuit. The output frequency of a VCO is fed to a set of frequency dividers (DIV1 to DIV4). The dividers output is selected by a multiplexer (MUX). Therefore depending on the multiplexer control, any divider output can be passed to the final output. A set of digital bits can be used to control the output frequency by controlling the multiplexer.

As only one divider output is used at a time, other dividers do not need to switch its output. Therefore the digital bits can also be used to switch off all other dividers which are not used at some particular time. This will reduce the power consumption but at the cost of more delay in the signal path due to extra gating. The number of dividers required is dependent on the input clock (VCO output) tuning range and targeted tuning range at the output.

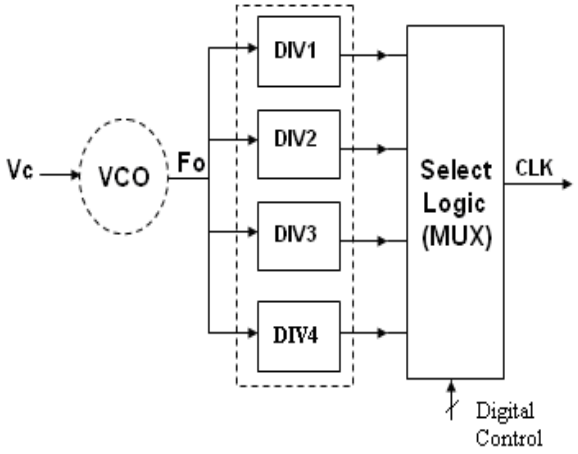


Figure 1: Basic architecture of extending frequency range by dividers

When the QVCO have a tuning range of +/-20%, 6 dividers are required to increase the tuning range up to -90%, as tabulated in Table 1. One of them is a fractional divisor with a division ratio of 3/2 or 1.5. This fractional division can be avoided if the QVCO have tuning range of +/- 33% which is extremely high [4]. On other hand if and the VCO tuning range is less than +/-20% more number of fractional divider will be required and that will make the design complex and prone to more jitter and phase error. An optimal choice is +/-20% which is a reasonable one for a practical VCO. The corresponding architecture is shown in Fig. 2. By this architecture any arbitrarily low frequency range can be obtained just by adding more number of divided by 2 circuits at the output

Table 1: Dividers require to extend the tuning range with Input Tuning Range of +/-20%

DIV ratio	Freq. Low	Freq. high
1	0.80	1.20
3/2	0.53	0.80
2	0.40	0.60
3	0.27	0.40
4	0.200	0.3
6	0.133	0.2
8	0.100	0.15

2.1. Optimization of the architecture

In the architecture shown in Fig. 2 some of the blocks can be reused and thus the architecture can be modified as shown in Fig. 3. A level shifter is placed so that dividers can be designed by digital logic families. After that a 1.5 or 2 frequency divider is used. All other division ratios can be made by adding divide by 2 block after the 1.5/2 divider block as shown. One dedicated divide by 2 is used to generate quadrature phases at the output. Quadrature phase is required in several applications such as in transceivers it is required to cancel the image frequency component. If quadrature clock is generated, the output frequency range obtained will be changed from $1.2f_{in}$-$0.1f_{in}$ to $0.6f_{in}$-$0.05f_{in}$ (Fig. 3). In other words, to meet the same output frequency, double input frequency is required if quadrature output is needed.

3. Design of the frequency range extender

The divider circuit is designed in Dynamic Transmission Gate Logic (DTGL) which can achieve higher frequency than other digital logic families such as TSPC and consumes low power than Current Mode Logic (CML) dividers. Divide by 2 circuit by DTGL logic is shown in Fig. 4 and fig. 5 for 2 phase and 4 phase output respectively. In these dividers in clock to output delay is just transmission gate delay. Therefore DTGL dividers not only can operate at higher frequencies but also contributes very less jitter. This is

because the output jitter is proportional to the square of the delay [7]. It consumes less area and the simplicity in the circuit reduces the design time considerably. However, due to its dynamic operation the dividers can produce a minimum output frequency of 2MHz.

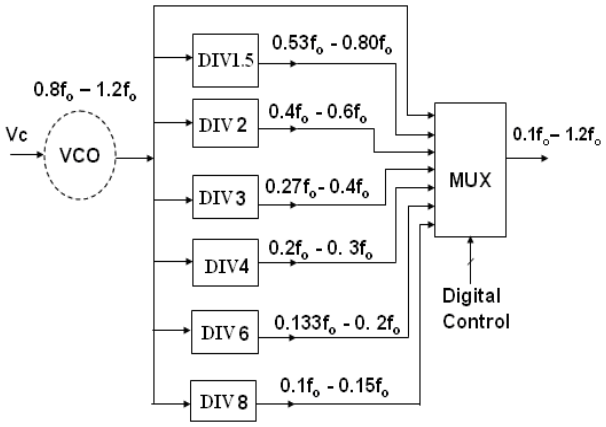


Figure 2: Architecture for extending VCO frequency from +/- 20% to 90%

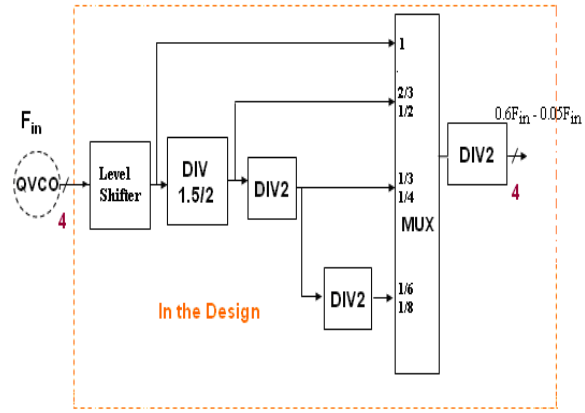


Figure 3: Modified architecture for frequency extension

The MUX shown in the Fig. 3 is made of transmission gate. Transmission gate multiplexer don't need any extra power though they can achieve good speed.

A level shifter is required just after the QVCO because the VCO gives low voltage swing output. Fig. 6 shows the level shifter used in this design. The feedback resistance in the first inverter biases the inverter at its trip point which ensures level shifting to rail-to-rail output from any dc level. The capacitor value is chosen such that the cut off frequency of the filter is around 10MHz which is 10 times less than the lowest frequency of operation to assure proper operation.

The other block is the 1.5/2 divider which is described in the next sub-section in detail. The SET or RESET flip-flop used in the control logic is made of static CMOS logic because it has to have the capability to operate in arbitrary low frequency.

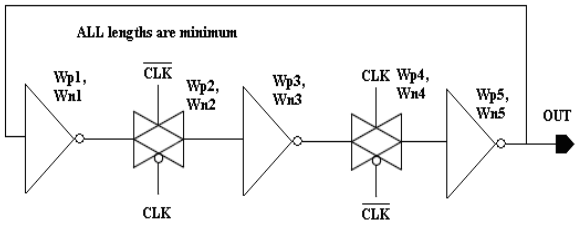


Figure 4: Frequency divider with DTGL

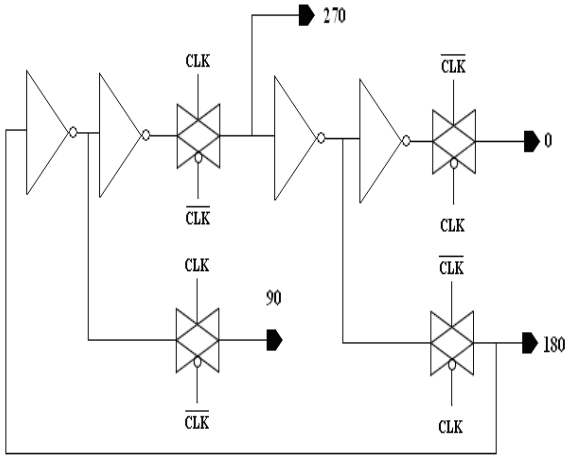


Figure 5: Divide by 2 by DTGL with four phase output

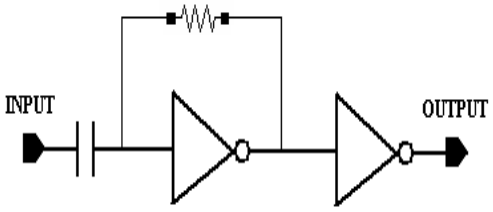


Figure 6: Level shifter

3.1. Divider-1.5 with 50% duty cycle

General 1.5 division circuits give non- 50% duty cycle. Phase selection method shown in [5] can be used to divide a four phase clock by 1.5 times [4]. It gives duty cycle of 33.3% or 66.7%. To achieve 50% duty cycle the clock feedback in the phase selection (clock interpolation) method is modified in this design. Here,

the MUX output itself clocks a state machine which selects one of the 4 input phases. The concept of the 1.5 division is shown in Fig. 7. The control is designed such that the phases are selected as follows 0°-270°-180°-90°. The waveform in Fig. 8 shows the input clocks, the multiplexer output (MUX_OP) and the final divider output (DIV2_OP). The control logic is triggered at the negative edge of the MUX_OP signal.

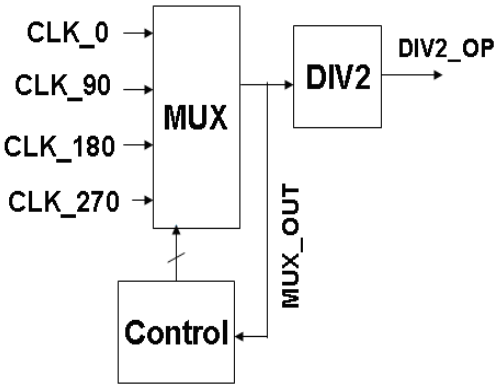


Figure 7: Concept of the 1.5 division

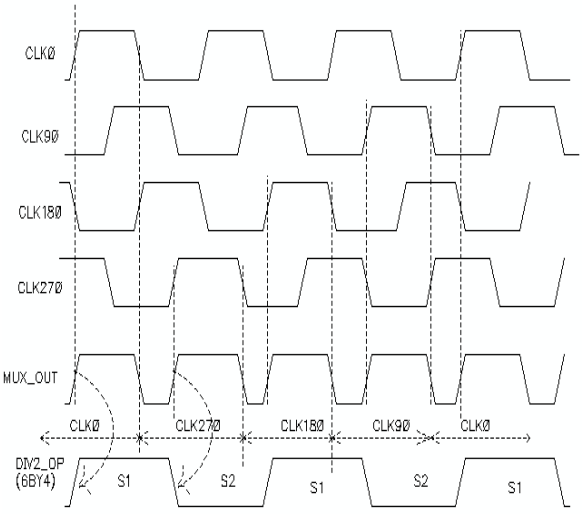


Figure 8: Four phase input and 1.5 division by phase interpolation

When the control bits select CLK_0, MUX_OUT will follow the CLK_0 signal. As soon as CLK_0 negative edge comes, the control is triggered and it selects the CLK_270 input. Therefore, the MUX_OUT will follow CLK_270 and rises after 90° from the previous fall. Again after the next falling edge, CLK_180 is selected. This will make MUX_OUT to rise 90° after its previous falling edge. Like this way the MUX_OUT gets a full cycle after 270° of the input clock, which makes MUX_OUT signal frequency 4/3 of the input frequency. The duty cycle of this signal is 66.67%. After a divider a perfect 50% duty cycle is obtained and the frequency is changed to 2/3 of the input frequency which is nothing but a 1.5 division.

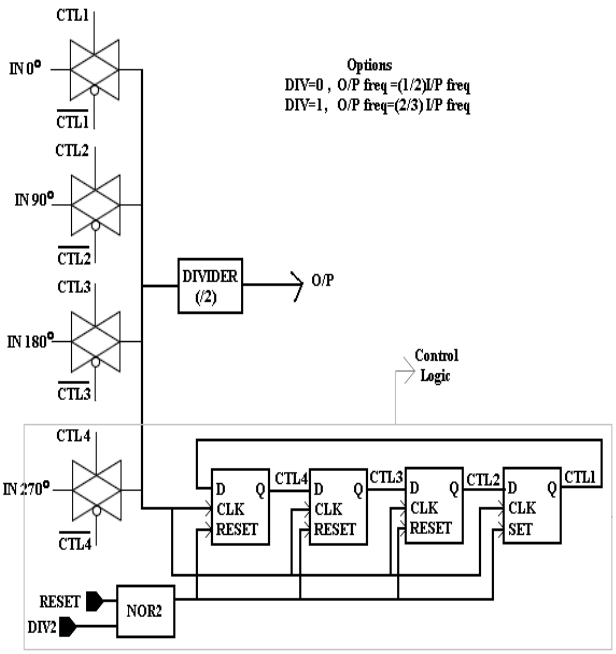


Figure 9: Schematic diagram of 1.5/2 divider with 50% duty cycle output

3.2. Divider 1.5 or 2 by external control

The usage of the 1.5/2 divider block has been shown in Fig. 3. It can be done by minimum modification over the 1.5 divider. The 1.5/2 divider circuit is shown in Fig. 9. Here the multiplexer is designed by transmission gates. Static CMOS flipflop is used in the controller.

If DIV2 input of the Fig. 9 is HIGH, it's SET or RESETS the flip-flops and only one phase permanently selected by the phase interpolating multiplexer. Therefore the input clock will be passed through a divide-by-2 circuit and gets 2 division.

The DTGL dividers in 1.5/2 block required both clock and complementary clock. The complementary clock can be generated either by inverter or by repeating the same circuit with the input clock inverted. The first method is power optimized but adds jitter and skew between clock and complementary clock. Second method can improve the frequency of operation at the cost of power. In this design the second method is adopted to meet the frequency of

operation. This part is not shown in the figure to reduce complexity.

3.3. Integration and overall circuit diagram

Fig. 10 shows the circuit diagram of the full tuning range extender circuit. The divider 1.5/2 is followed by a set of dividers and their output is selected by a multiplexer. The multiplexer output goes to a divider with 4 phase output. The complementary clock required for the final divider can also be realized by an inverter or a separate multiplexer having the complementary output of the first three dividers (not shown in the figure). Adding an inverter will add jitter and phase mismatch in the final output. Therefore the second option is chosen in this design which will improve the jitter performance at the cost of power consumption.

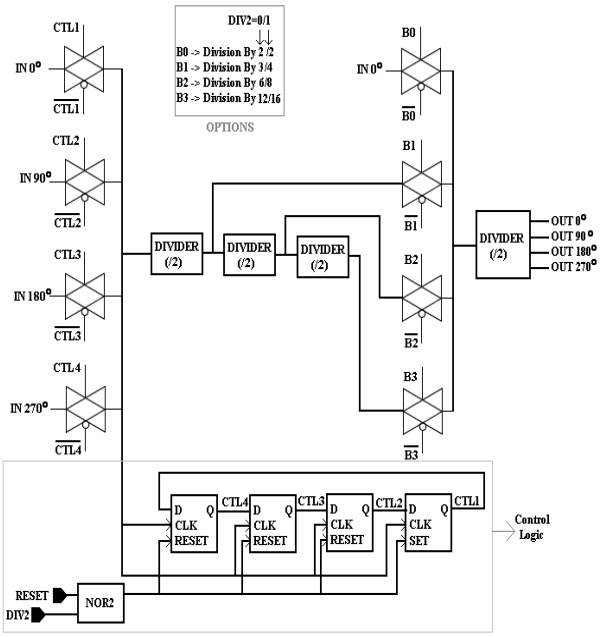


Figure 10: Full schematic of the tuning range extension circuit

4. Simulation result

Simulation on the full frequency extension circuit (as in Fig. 10) is done with 90nm CMOS technology. The highest input frequency at which the circuit is functional is 4GHz. Thus the output frequency range achieved is from 400MHz to 4.8GHz for differential clock with 50% duty cycle and from 200MHz to 2.4GHz for quadrature output. Maximum input frequency is limited by the first 1.5 divider circuit. Maximum power consumption is obtained when the full extender works as a divider-by-6 circuit. In that mode, power consumption of the extension circuit is 2.1mW from 1.2 power supply voltage with a 4GHz input. For the above case jitter value is found out by simulation. The mismatch jitter contribution is 2ps. The output jitter due to noise sources such as thermal noise, flicker noise etc is 0.15ps.

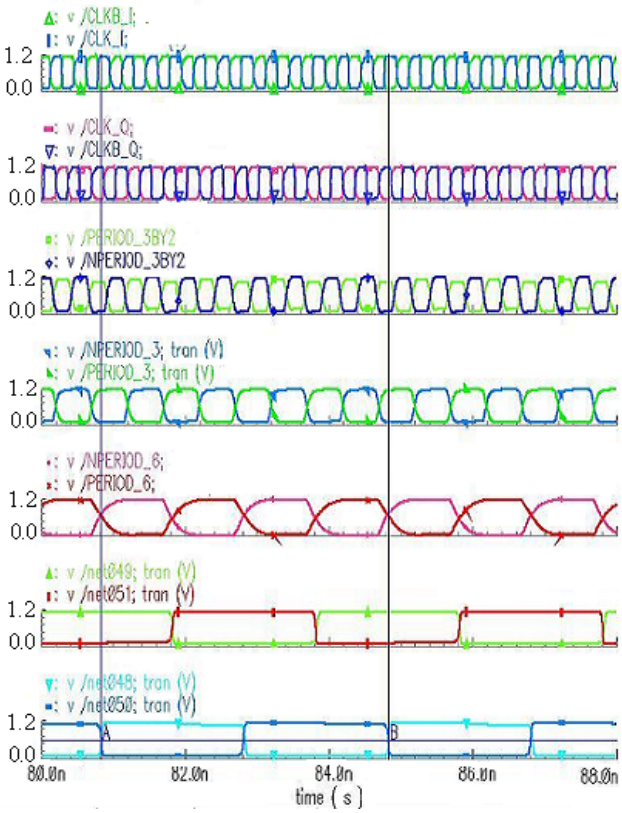


Figure 10: Waveform of 4-phase input clock, 1.5-division, 3-division and 6-division (quadrature) outputs [I and Q of the Quadrature clocks are shown separately]

For other division modes, both the power consumption and the output jitter gets reduced which is verified by simulation. As the no of divider decreases in the path the jitter contribution will be less. If quadrature clock is not required, the power consumption can be further reduced by 0.2mW to 0.6mW depending on the output frequency.

5. Summary and conclusion

In summary, an architecture which can extend the MPCG frequency tuning range is explored. By using a divider chain, wide tuning range is achieved with 4 phase input clock. This block can be incorporated after

a QVCO with +/- 20 % tuning range. With a 4GHz QVCO the extender circuit can produce 400MHz to 4.8GHz with differential 50% duty cycle output and 200MHz to 2.4GHz with Quadrature output consuming power as low as 2.1mW from 1.2V power supply in 90nm technology. This architecture is highly suitable to extend the VCO tuning range for wide frequency range applications.

6. References

[1] Kim, Jae Y., Chih-Wei Yao, Willson, Alan N, "A programmable 25 MHz to 6 GHz rational-K/L frequency synthesizer with digital Kvco compensation," *IEEE Int. Symp. on Circuit & Systems*, pp. 2629-2632, May. 2008.

[2] Jongsik Kim, Jaewook Shin, Seungsoo Kim, and Hyunchol Shin, "A wide-band CMOS LC VCO with linearized coarse tuning characteristics*", IEEE transaction on circuit and systems—II: express brief*, vol. 55, pp. 399-403, May 2008.

[3] Adil Koukab, Yu Lei and Michel J. Declercq, "A GSM-GPRS/UMTS FDD-TDD/WLAN 802.11a-b-g multi-standard carrier generation system," *IEEE J. Solid-State Circuits*, vol. 41, pp. 1513-1521, Jul. 2006.

[4] Davide Guermandi, Paola Tortori, Eleonora Franchi, Antonio Gnudi, "A 0.75 to 2.2GHz continuously-tunable quadrature VCO," *Int. Solid-State Circuit Conference*, pp. 536-537, Feb. 2005.

[5] J. Craninckx and M. Steyaert "A 1.8-GHz low-phase-noise voltage controlled oscillator with prescaler," *IEEE J. Solid-State Circuits*, vol. 30, pp. 1474, Dec. 1995.

[6] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolic, Digital Integrated Circuits: A Design Perspective, 2nd Edition, *Prentice Hall, ISBN*: 0-13-090996-3, 2003.

[7] X. Gao, E. Klumperink and B. Nauta, "Advantages of shift registers over DLLs for flexible low jitter multiphase clock generation", *IEEE Trans. Circuits Syst. II*, vol. 55, pp. 244 -248, Mar. 2008.

2009 22nd International Conference on VLSI Design

Fuzzy Logic Based Guidance to Graph Grammar Framework for Automated Analog Circuit Design

Angan Das and Ranga Vemuri
Department of Electrical and Computer Engineering, University of Cincinnati, Cincinnati, OH 45221-0030, USA.
Email: {dasan, ranga}@ececs.uc.edu

***Abstract*— This paper introduces a fuzzy logic based guidance architecture to a graph grammar framework for automated design of analog circuits. The grammar generates circuit topologies through a derivation tree. To boost this tree based synthesis mechanism, smaller building blocks in the form of subtrees have been used for the purpose. These blocks have been automatically generated and their appropriateness for the design is updated runtime through the fuzzy system. Fuzzy logic helps to provide a smooth gradation for the relative merit of each block with respect to the design synthesized. The tool has been used to design an operational amplifier and a voltage controlled oscillator.**

I. INTRODUCTION

Modern day SoCs house analog, digital and RF sections on a single chip. But unlike tools in the digital domain that have achieved noteworthy success, the development curve of automated analog synthesis tools has always been slow and inadequate, both in the industry as well as in academia [1]. Analog synthesis comprises of two steps — Topology formation and subsequent Sizing of the topology. Topology formation can again be achieved through two different ways — Topology selection and Topology generation.

Owing to heavy designer dependency and huge set-up effort, heuristic-based selection approaches [2], [3] gradually evaded with the advent of generation techniques. Circuits were actually generated with the aid of evolutionary algorithms like Genetic Algorithms (GA) [4] and Genetic Programming (GP) [5]. But they involved heavy computational burden. This shortcoming was alleviated in [6], where design-specific building blocks were used to construct a circuit. Unfortunately, this approach requires a designer certified block library for each design class. To meet this deficiency, adaptively built blocks were introduced in our earlier work [7]. But there the blocks were judged using numerical approaches based on a single weighted cost function. [8] is one of our prior works on passive circuit synthesis.

In a pursuit to overcome all of the above drawbacks, we introduce a graph grammar based framework for the generation of analog circuit topologies. A grammar can be defined as a set of rules that can generate a construct from a list of terminals [9]. In case of graphs, these rules are used to construct a complete graph from a variety of nodes and allowable interconnections. In this work, we construct an analog topology graph using components like PMOS and NMOS.

In the process, to boost the synthesis run, we also construct a certain number of circuits in each set from meaningful building blocks that have been dynamically obtained. The merit or appropriateness of these blocks is quantified through a guidance architecture based on fuzzy logic. Fuzzy rules help to represent, manipulate, and implement a human's heuristic knowledge on how to control a system. Hence fuzzy logic systems act as artificial decision makers and thereby have numerous engineering applications [10], [11]. In the analog topology synthesis domain, FASY [12] is an example where fuzzy rules are used for topology selection among a fixed set of alternatives.

Finally, it is to be noted that most analog design problems are multi-objective optimization problems that often involve conflicting objectives. In this regard, we adopt the Non-dominated Sorting Genetic Algorithm (NSGA-II) [13]. Using the above techniques, we have evolved an operational amplifier design and a voltage controlled oscillator (vco) design.

This work was supported by National Science Foundation under award numbers CCF-0429717 and CNS-0421092.

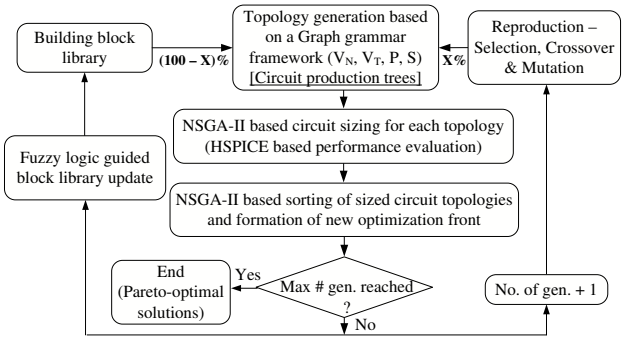


Fig. 1. Synthesis flow

The rest of the paper is organized as follows. Section II describes the synthesis methodology. Section III introduces the graph grammar based topology generation technique. In Section IV, we describe the solution reproduction mechanisms. Section V deals with the fuzzy logic guidance architecture to quantify the merit of building blocks. Section VI discusses the circuit sizer. In Section VII, we synthesize an opamp and a vco design. Section VIII concludes the work.

II. SYNTHESIS METHODOLOGY

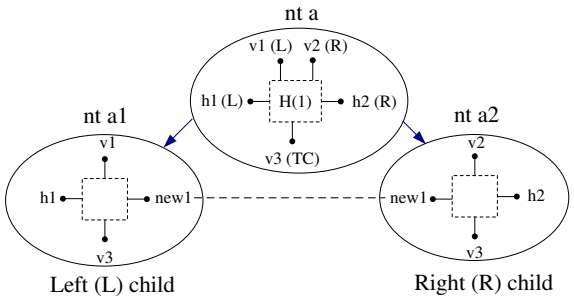
Fig. 1 outlines our approach. Similar to any evolutionary algorithm, we maintain a collection of solutions or chromosomes in the form of a generation. Parent generations breed to produce offspring generations. The procedure continues for the allowable number of generations and finally a Pareto-optimal set containing the best solutions is obtained.

Circuit topologies are synthesized using library components through the graph grammar technique. These circuits are eventually sized whereby device dimensions and voltage and current values are assigned. Here, the multi-objective sizer is based on NSGA-II. NSGA-II produces a set of Pareto-optimal solutions, where no two solutions, belonging to the same optimal front, are inferior compared to each other [13]. Subsequently, a wide range of sized topologies is obtained. This whole set of sized circuits is then sorted with NSGA-II again. A new solution front is produced and the process continues.

The blocks used for circuit formation are housed in a library. Initially the library contains basic components only. But after each generation, the library is updated whereby new blocks are added and existing blocks are ranked. In this way, it gradually includes bigger and functionally more useful blocks as the synthesis progresses. The ranking is done depending on fuzzy rules. Compared to crisp numerical techniques, fuzzy logic provides a smoother gradation for the merit of each block with respect to the design synthesized.

Also, from the second generation onwards, children circuits are produced in two ways — one fraction through the crossover and

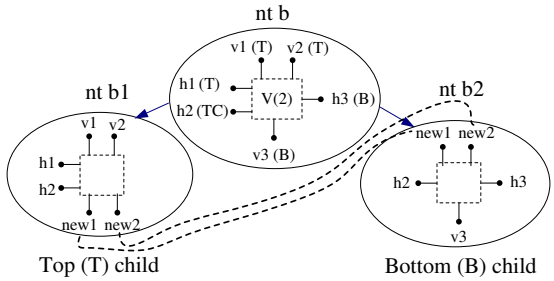
1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.79



Structural disassociation notations for parent node (nta):
H(1) – Horizontal division with 1 new interconnection (viz. new1)
Result: new1 is hor-out for left child (nta1) and hor-in for right child (nta2)

Terminal disassociation notations for parent node (nta):
L – Assigned to left child, R – Assigned to right child,
TC – Assigned to both (T-Connection).

(a)



Structural disassociation notations for parent node (ntb):
V(2) – Vertical division with 2 new interconnections (viz. new1 and new2)
Result: new1 and new2 are ver-out for top child (ntb1) and ver-in for bottom child (ntb2)

Terminal disassociation notations for parent node (ntb):
T – Assigned to top child (must for ver-in of ntb), B – Assigned to bottom child (must for ver-out of ntb), TC – Assigned to both (T-Connection).

(b)

Fig. 2. Production rules (example shown): Structure and terminal disassociation rules — (a) Horizontal disassociation and (b) Vertical disassociation

mutation reproduction mechanisms of parent circuits [14], while the rest are produced afresh from the better ranked library blocks.

III. GRAPH GRAMMAR: CIRCUIT FORMATION

A graph grammar (G) is represented by the 4-tuple:

$$G = (V_N, V_T, P, S)$$

where V_N denotes the set of non-terminal symbols (*nt-symbols*) or variables, V_T the finite set of terminal symbols (*t-symbols*), P denotes the finite set of production rules, and $S(\in V_N)$ is the start symbol. Production rules transform an nt-symbol into other nt-symbols or t-symbols. Each production rule therefore derives a successor graph (connected left and right graphs) out of a predecessor graph.

In this work, analog circuit topologies are treated as connected planar graphs. This graph is generated by production rules encoded in form of a derivation tree. The tree starts with a single root node (start symbol) and gradually branches off through the allowable rules. Intermediate nodes represent nt-symbols, while leaf nodes stand for t-symbols. Fig. 6(a) shows an example topology producing derivation tree, and Fig. 6(b) the corresponding circuit topology.

A. Start Symbol (S)

The start symbol (S) or root node of the derivation tree is the top-level blackbox representation of the design. The terminals of the blackbox are classified into four terminal sets, viz. horizontal-in (hor-in), horizontal-out (hor-out), vertical-in (ver-in) and vertical-out (ver-out). With the above consideration, node $nt0$ in Fig. 6(a) shows the start symbol of a 2-input (V_{IN+} and V_{IN-}), 1-output (V_{OUT}) design. V_{IN+} and V_{IN-} belong to hor-in, V_{OUT} to hor-out, Vdd power rail belongs to ver-in, and Gnd belongs to ver-out.

B. Non-terminal symbols or nt-symbols ($\mathbf{V_N}$)

All tree nodes barring the leaf nodes are composed of nt-symbols (S is a part of V_N). These are blackbox structures with the same set of directional terminals as described above. They represent substructures that are obtained through the step-by-step decomposition of the main top-level structure, governed by production rules. Node ntb in Fig. 2(b) is an example representation of an nt-symbol tree node.

C. Terminal symbols or t-symbols ($\mathbf{V_T}$)

nt-symbols may be replaced by t-symbols. The t-symbols form the leaf nodes of the tree, and thereby gives the SPICE netlist. Hence these symbols comprise of all the actual circuit elements like PMOS, NMOS, resistors, capacitors, etc. required for the design.

Terminal renaming: To avoid pathological structures, two terminals belonging to different sets, within the same symbol, are not interconnected during evolution. Hence, to evolve all kinds of topologies, certain t-symbols have the same physical node signified as separate terminals. For e.g., in Fig. 6(a), terminals $n2$ and $n4$ of t-symbol $t0$ are both the same drain node of transistor $M0$. In such cases, these nodes are later renamed (the same) during netlist generation, from a knowledge of the respective t-symbol.

D. Production rules (P)

Production rules form the core of any grammar. In this work, each nt-symbol may either produce two new nt-symbols (children nodes) or may be replaced by a t-symbol. nt-symbols at a lower tree depth have higher probability for the former and vice-versa. Thereby, all the production rules may be divided into two broad classes —

(1) nt-symbol → nt-symbol: When an nt-symbol gives rise to two new nt-symbols, the underlying rules are again composed of two sets:

(a) Structure disassociation rules: The parent structure may disassociate horizontally with the left (L) and right (R) child physically placed adjacently, or vertically with the top (T) child placed above the bottom (B) child. Also, the two children produced may be interconnected through one or more new terminals. Terminal-set wise, they will be 'in' for one and 'out' for the other. Figs. 2(a) and (b) demonstrate horizontal and vertical disassociation respectively. As shown in Fig. 2(a), 'new1' is the new horizontal connection.

(b) Terminal disassociation rules: The directional terminals of the parent structure are distributed among its children substructures. In case of a horizontal break-up, the horizontal terminals are assigned either to the left (L) or to the right (R) child. Vertical terminals can be assigned either to L, or to R, or to both of them (TC).

For vertical disassociation, all ver-in terminals of the parent are assigned to top (T) child, and all ver-out terminals to bottom (B) child. Horizontal terminals may be assigned to either or both of them. For e.g., in Fig. 2(b), terminal $h1$ is assigned to T, $h3$ to B, and $h2$ is assigned to both of them (TC).

(2) nt-symbol → t-symbol: In this case, there are no disassociation rules. Instead, there is an actual terminal-terminal mapping between those of the nt-symbol and the t-symbol. This helps in generating the SPICE netlist. As is evident in Fig. 6(a), terminals $n6, n7$ and $n8$ of node $nt0121211$ corresponds to the source, gate and drain nodes of the PMOS in t-symbol $t3$. $t3$ represents the transistor $M3$ in the final circuit of Fig. 6(b).

IV. REPRODUCTION MECHANISMS

Topology generation trees are built in two ways — one set from the updated block library, and the remaining ones through the reproduction mechanisms viz. crossover and mutation of parent trees.

A. Trees from updated block library (better blocks)

Here, during tree formation, the production rule for nt-symbol → t-symbol is applied with a higher probability. This is because after the first generation, the t-symbols are not restricted to simple design elements only. They also comprise of bigger subcircuits in the form of subtrees (Section V-A). Better ranked subtrees are preferentially selected over the poorer ones and used to form a tree.

B. Crossover

Crossover swaps subtrees (subcircuits) between two parents to produce two offsprings. Two parents are selected through the crowded comparison operator [13]. Subsequently, we decide on a range for the number of design elements to be contained from each parent. Accordingly nodes are individually selected from both the parent trees, that match each other in terminal properties, and satisfy the size constraints. This kind of crossover — a) Prevents the production of pathological or structurally incorrect circuits, b) Keeps the size of the circuit within bounds as decided by the user.

C. Mutation

To prevent premature convergence, a certain number of randomly chosen children circuits undergo mutation, through subtree replacement. A randomly selected portion of the circuit tree is replaced with a library subtree. Also, alike crossover, we ensure the structural integrity of the generated circuits and keep a tag on the circuit size.

V. FUZZY LOGIC GUIDED BLOCK LIBRARY UPDATE

The building block library is continually updated to include bigger and functionally more useful blocks. This is a dynamic procedure and is explained as follows.

A. Subtrees from circuit trees: New t-symbols (blocks)

After a circuit topology derivation tree is formed and the circuit performance is evaluated, subtrees are extracted bottom-up out of the main circuit tree. These subtrees (subcircuits) are then treated as new t-symbols, that are later used in circuit formation. It is so because they can replace any nt-symbol in a circuit tree, when there is a match between the terminal properties of that nt-symbol and the root node of this new t-symbol. Within the t-symbol library, a hierarchy is maintained based on number of terminal types and number of design elements (PMOS, NMOS, etc.) contained within the block.

B. Library building block parameters

Each library building block has two parameters [7]:

1) *No. of occurrences (N_{block}):* This count signifies the number of times the block has been used for circuit formation.
2) *Fitness (F_{block}):* This denotes the merit of the block. In other words, it is the level of appropriateness with which the block should be used for the design under consideration.

C. Library update: Existing blocks and new blocks

The library update algorithm is given in Algorithm 1. The update procedure followed for the different block parameters is as follows:

- *N_{block}:* The occurrence count is incremented every time the block is used. If the block is new, it is included into the library.
- *F_{block}:* For each performance measure, the new average fitness of an existing block comes from its old fitness and its newly calculated merit - $\hat{F_{\text{subtree}}}$. This merit is obtained through a fuzzy logic based architecture explained below. For a new block, the fitness is given by a scaled value of the merit (η - scaling factor).

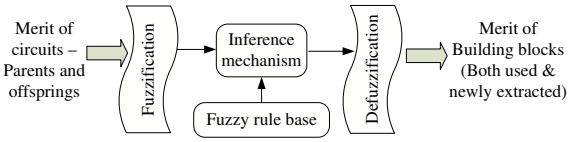


Fig. 3. Fuzzy logic based guidance architecture

At the end of each generation, the blocks are sorted and ranked based on their fitness for each performance, as per NSGA-II.

D. Fuzzy Logic Based Guidance Architecture

1) Fuzzy system — An overview: Fig. 3 shows that a fuzzy system has a fuzzification interface that converts the crisp input values to fuzzy linguistic values; a set of fuzzy rules in the form of a rule base; an inference mechanism that decides which rules are on at what time; and a defuzzification interface that converts the conclusions reached by the system into real or crisp values.

Fuzzy rules are in the form of *if-then rules*. The *if* part signifies the premise or antecedent, while the *then* stands for the consequent. For e.g., a rule for a two input (x and y) one output (z) fuzzy system with possible linguistic values signified by low, medium and high, can be:

Rule → If x is high and y is medium, then z is large.

The rule signifies the way z should be concluded, based on the states of x and y. The values of x and y determines - (a) whether they are small/medium/large; (b) the degree to which they are small/medium/large, which in turn determines the strength or premise of the corresponding rule. Each rule has a certain premise $\in [0, 1]$. Similarly, other rules may be formulated with different combinations of low, medium and high for variables x and y.

Now, at any given instant, a crisp value of an input variable may represent one or more linguistic values for that variable, though with varying degrees of truthfulness. Each such derived linguistic term triggers one or more rules, which are thereby turned *on*. Hence, at any time, the output is a result of all such *on* rules.

Algorithm 1: Building block library update after each generation

Input: Gen-(n) block library, gen-(n) parents and offsprings
Output: Updated block library: For gen-(n+1) formation
procedure Block library - update and extension
forall *parent-offspring pairs* **do**
  forall *performance objectives* **do**
    f1 = fuzzy_membership_circuit_quality();
    f2 = fuzzy_membership_block_impact();
    `merit` = defuzzify (rules(f1, f2), centroid method);
  end
end
forall *offspring circuits (trees)* **do**
  Extract subtrees hierarchically out of main circuit tree;
  $\hat{F_{\text{subtree}}}$ = appropriate `merit` ();
  forall *subtrees (≡building blocks)* **do**
    if *block is not found in library (i.e. new block)* **then**
      $N_{\text{block}} = 1$; // Introduce block into the library
      $F_{\text{block}} = \hat{F_{\text{subtree}}} * \eta *$ generation_no;
    else
      $N_{\text{block}} = N_{\text{block}} + 1$;
      $F_{\text{block}} = \frac{F_{\text{block}} * N_{\text{block}} + \hat{F_{\text{subtree}}}}{N_{\text{block}} + 1}$;
    end
  end
end
Rank all blocks (multiobjective sorting) as per NSGA-II;

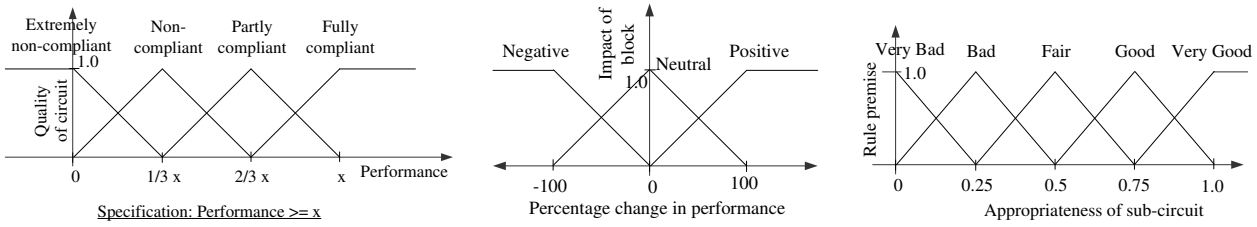


Fig. 4. Fuzzy membership functions: (a) Input-1: Quality of circuit (shown for an e.g. objective $\geq$ x), (b) Input-2: Impact of block (shown for an increasing objective), and (c) Output: Appropriateness/merit of subcircuit

2) Application to library update — Suitability of subcircuits: Related to the inherent nature of analog circuits, the performance of a circuit is actually contributed by the appropriateness or merit of the participating blocks [15]. But in this CAD framework, owing to the absence of different design heuristics and equations, we need some further assumptions. It is assumed that during crossover, the difference in performance (or fitness) between the child and the parent owes equally to that of the incoming and outgoing blocks (swapped blocks) [7]. A raise in fitness from parent to child suggests that the incoming block was good i.e. it had a positive impact, and the outgoing block was equally bad (negative impact).

3) Fuzzification interface: Following the above assumption, the two inputs to the fuzzy system are — (a) Absolute performance of the circuit produced, and (b) Impact of the subcircuit block, measured by the change in performance from parent to offspring. These measures are then transferred to linguistic values or fuzzy sets through the fuzzification interface. Fuzzy sets $\in$ [0, 1] gives the membership grade or truthfulness for each of the linguistic terms. In this work, triangular membership functions have been used. The linguistic terms or fuzzy sets for the two inputs are:

- *Performance of circuit:* The crisp numerical input of the circuit performance is categorized into four ranges. Accordingly, the performance is symbolized with four linguistic terms, viz. Extremely non-compliant (ENC), Non-compliant (NC), Partly compliant (PC), and Fully compliant (FC). To cite an example, Fig. 4(a) shows an example for a $\geq$ objective, where the horizontal axis stands for the crisp input value and the vertical axis represents the membership for the corresponding linguistic value. The demarcations for the four categories are 0, $(1/3)x$, $(2/3)x$, and x. For e.g., if performance = $(1/6)$x, the circuit quality is ENC with 0.5 membership grade, and NC with the same (0.5) grade.
- *Impact of subcircuit block:* The impact of a participating subcircuit block may be positive, negative or neutral. These linguistic values are decided based on the % change in performance the block introduces, related to the design. Similar to the previous input, there are three ranges for categorizing the impact. They are — a 100% change in performance in the favorable direction of the objective, a 100% change in the opposite direction, and no change in performance (or 0% change). Fig. 4(b) shows the membership function for an example increasing objective. As shown, a 1.0 membership grade for positive impact occurs for a $\geq 100\%$ increase, while that for negative impact occurs for a $\geq 100\%$ decrease, with intermediate values lying in between. Similar theory applies to neutral impact.

E. Rule base

The rule base used in this fuzzy framework is given in Fig. 5. Rules are formulated for all combinations of the circuit performance and subcircuit impact inputs. The inferred output relates to the appropriateness of the subcircuit involved. For e.g., *if* the quality of the circuit produced is *fully compliant* with the specifications, *and* the impact of the participating block is *positive*, *then* the subcircuits contained within the block are considered *very good* for the design. The rules show that the output follows a smooth gradation, with linguistic values ranging from very bad to very good with intermediate values denoting bad, fair and good. Finally, it is to be noted that for blocks used in circuits constructed without crossover, or for newly extracted blocks, the contribution of the subcircuit impact towards a rule is assumed neutral.

F. Inference mechanism

The inference section of a fuzzy system is the primary decision-making section. It collects the recommendations from each rule to determine which rules are *on*, and accordingly gives linguistic conclusions (VG, G, F, B, and VB) on the consequent. For each and-based rule that is *on*, the degree of membership of the two inputs collectively gives the premise or degree to which the relevant rule is *on*. In this respect, we adopt the minimum operation in this work [11]. Thereby, the minimum among the two input memberships is used to determine the truthfulness of the corresponding rule.

G. Defuzzification interface

A defuzzification interface converts the linguistic-natured conclusions into real or crisp values. The conclusions of all the *on* rules are combined to give the most certain output value. Similar to the inputs, we use triangular membership functions for the output as shown in Fig. 4(c). But in this case, the horizontal axis gives the crisp output, while the vertical input axis signifies the degree of the linguistic value of the output, which is in turn dependent on the rule premise. To calculate the crisp output, here, we adopt the widely popular center of gravity or centroid method [11]. For a fuzzy system with N rules, and b_i denoting the center of the membership function of the consequent of rule-$i (i \in N)$, along with w_i denoting the area under the output membership, the output is given by —

$$\text{Output}_{\text{crisp value}} = \frac{\sum_{i=0}^{N} b_i * w_i}{\sum_{i=0}^{N} w_i}$$

Impact of block / Quality of circuit	Negative	Neutral	Positive
Extremely non-compliant	VB	VB	B
Non-compliant	VB	B	F
Partly compliant	B	F	G
Fully Compliant	F	G	VG

Notations for the appropriateness of sub-circuits:
VB → Very Bad, B → Bad, F → Fair,
G → Good, VG → Very Good

Fig. 5. Fuzzy rule base (*AND* rules)

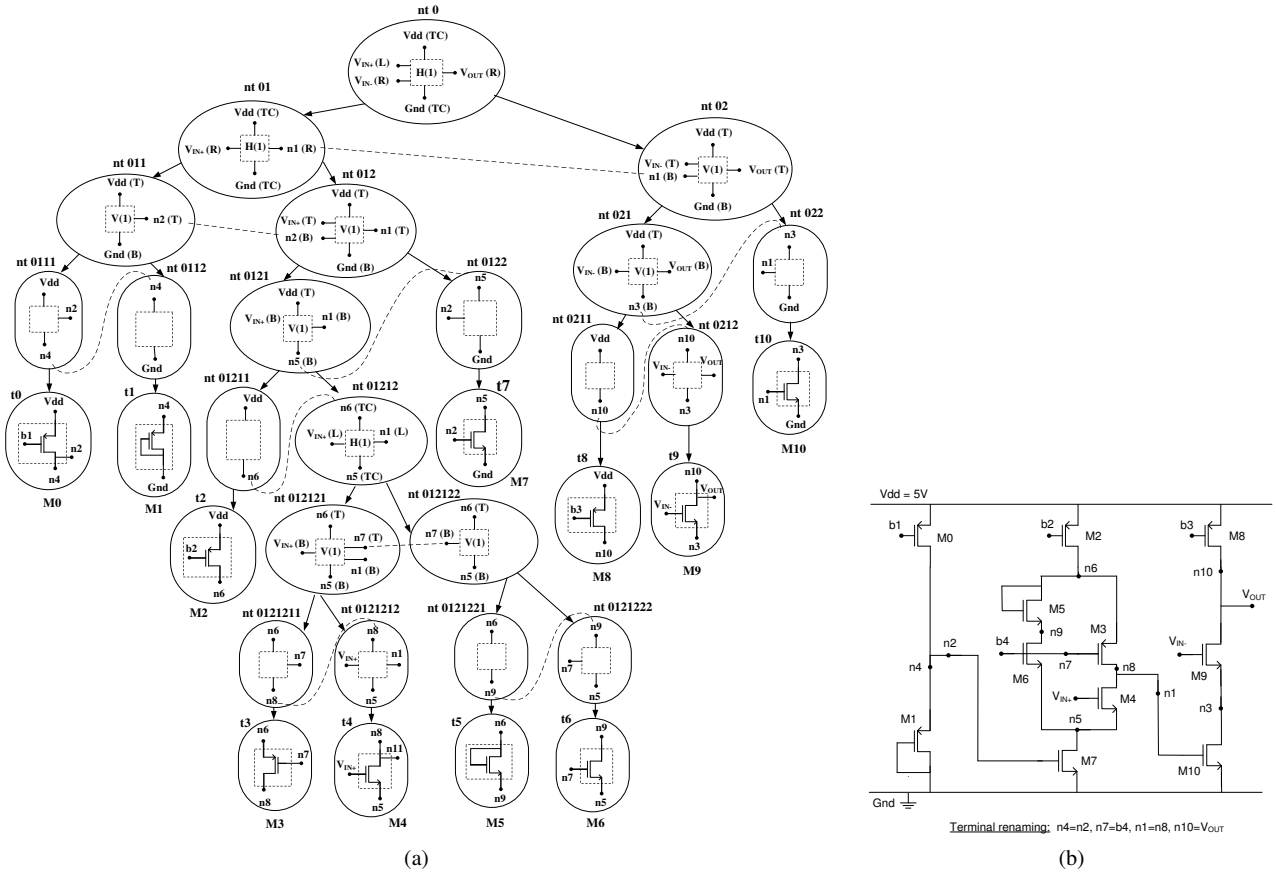


Fig. 6. 2-input 1-output opamp design — (a) Derivation tree for topology generation (b) Corresponding circuit topology

VI. MULTI-OBJECTIVE SIZER

Drawing a conclusion on the quality of an analog circuit topology is possible only after the topology has been sized properly [1]. A circuit sizer assigns dimensions and values to all the active and passive devices in the circuit like PMOS, NMOS, voltage source, resistors, and so on. Following in line with some efficient sizers developed lately [16], we have implemented a sizer based on the NSGA-II algorithm. In this regard, we have used the crowded comparison operator for selection, and the SBX method for crossover [13].

Inputs to the sizer comprise of the range and granularity of design variables like transistor dimensions, voltage source values, etc. Same gate-node transistors are assumed to be matched and their widths are in integral multiples of each other. For performance evaluation, despite some computational effort, HSPICE is used because it provides accuracy, requires no set-up effort (unlike symbolic methods), and helps to integrate the tool into any existing industry framework.

VII. EXPERIMENTS AND RESULTS

Two different designs are chosen as synthesis benchmarks. One is an operational amplifier, while the other is a voltage controlled oscillator. The specifications are given in Table I. These specifications have been decided after carefully considering several designs [15]. The various algorithm run parameters are set after tuning the algorithm over several runs. For topology synthesis, 30% of the circuits are formed using library blocks and the rest 70% through crossover and mutation. For the sizer, each topology is sized using 50 generations with 48 chromosomes per generation. The technology file used is that of 0.18μ. The range of variables are (in the format min:granularity:max) W_tran $\rightarrow$ 0.36:0.18:80μ and V_bias $\rightarrow$ 0:0.1:5.0V, the symbols carrying their usual meanings. All transistor lengths are fixed at 0.36μ for ease in optimization.

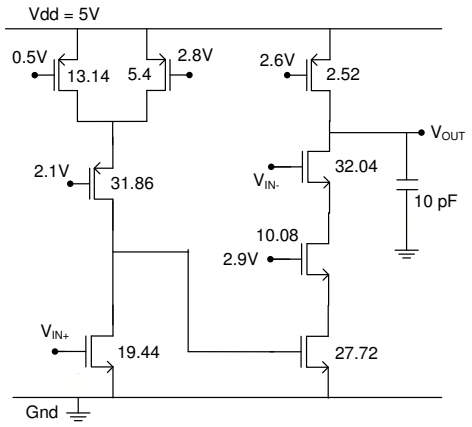
A. Operational Amplifier Synthesis

The operational amplifier or opamp is chosen owing to its versatility and wide usage, either by itself or as a part of a bigger analog module. Such modules include comparators, filters, pre-amplification stages, signal converters and so on [15]. The 2-input 1-output opamp design is given (user-provided) a load capacitance (C_L) of 10pF.

A total of 100 chromosomes or topologies are built per generation and the synthesis was run for 100 generations. The results are given in Table I. 18 designs are produced that conform to the specifications and occupy the Pareto-optimal front. Fig. 6(b) and Fig. 7 shows two designs. For the opamp in Fig. 6(b), the parameters are — (width in μ) M0 = 8.46, M1 = 18.72, M2 = 3.78, M3 = 9.36, M4 = 16.92, M5 = 16.02, M6 = 9.36, M7 = 5.4, M8 = 1.26, M9 = 34.92, and M10 = 16.2; (voltage in volts) b1 = 1.2, b2 = 1.5, b3 = 2.5, and b4 = 0.1. The corresponding performances are — Gain = 42.9 dB,

TABLE I
DESIGN SPECIFICATIONS AND RESULTS

SPECIFICATIONS / OBJECTIVES		
Design-I: Operational Amplifier — Specifications: DC Gain $\geq$ 40 dB, 3dB frequency $\geq$ 50 MHz, UGF (Bandwidth) $\geq$ 5 GHz		
SPECIFICATIONS / OBJECTIVES		
Design-II: Voltage controlled oscillator — Specifications: Center oscillation frequency $\geq$ 2 GHz, Tuning range $\geq$ 0.1 GHz		
RESULTS	**Design-I**	**Design-II**
No. of Pareto-optimal designs	18	8



Gain = 47.18 dB, UGF = 7.52 GHz, 3dB frequency = 327 MHz

Fig. 7. An optimal opamp design obtained (All transistor widths are in μ)

UGF = 5.04 GHz, 3dB freq = 174 MHz. After the synthesis run, the block library was observed. Fig. 8 shows some of the highly ranked library blocks. They include designer identifiable blocks like differential pairs, as well as non-bookish MOS structures.

To cite a comparison with previous works, [5] used $640,000$ chromosomes per generation and generated a single solution for a 5dB amplifier design after 45 generations. Hence our approach converges faster and produces more designs compared to [5].

B. Voltage Controlled Oscillator (VCO) Synthesis

We choose the voltage controlled oscillator (vco) as our next synthesis benchmark owing to its numerous applications ranging from phase locked loops to function generators, frequency synthesizers and so on [15]. The oscillation frequency of a vco is varied through a control voltage (V_c). The design parameters of interest related to a vco are its center oscillation frequency and the tuning range. Accordingly, the specifications are provided in Table I.

The synthesis framework uses 100 chromosomes and runs for 50 generations. The peformance metrics are obtained using HSPICE Fourier analysis. As shown in Table I, the tool produced 8 compliant designs that occupy the Pareto-optimal front. The design shown in Fig. 9 is an example of an optimal design obtained. It has an oscillation frequency of 2.86 GHz with $V_c = 2.5$ V. Through variation of V_c, the tuning range achieved for the design is 0.23 GHz. This is better than [6], which synthesized a single similar design taking 40 generations, each consisting of 600 chromosomes.

With regard to the block library, Fig. 10 shows some of the better blocks obtained. They comprise of of a set of same-biased transistors and diode-connected loads among others.

VIII. CONCLUSION

We have presented a fuzzy logic based guidance architecture to the graph grammar topology generation framework for automated analog

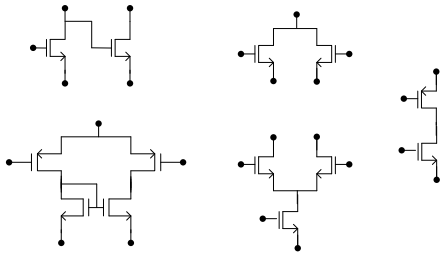


Fig. 8. Some of the better library blocks obtained in opamp synthesis

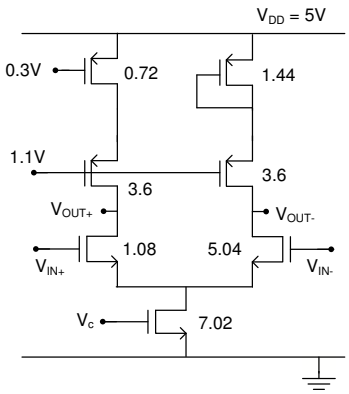


Fig. 9. VCO circuit generated (All transistor widths in μ, L = 0.36μ)

circuit design. Circuit topologies are generated based on appropriate production rules represented though derivation trees. In the process, a dynamically generated building block library has been used for the purpose. The library is updated through the fuzzy architecture. The fuzzy system includes a fuzzification interface, the fuzzy rule base, an inference mechanism and a defuzzification interface. Results show that our tool has been successful in synthesizing an operational amplifier design and a voltage controlled oscillator design better compared to previous works.

REFERENCES

[1] R. A. Rutenbar, G. G. E. Gielen, and B. A. Antao, *Computer-Aided Design of Analog Intg. Cir. and Sys.* Wiley-IEEE Press, May 2002.

[2] R. Harjani, R. A. Rutenbar, and L. R. Carley, "OASYS: A framework for analog circuit synthesis," *IEEE Trans. on CAD*, Dec. 1989.

[3] W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS Opamp synthesis by means of a genetic algorithm," in *Proc. of DAC*, Jun. 1995.

[4] J. D. Lohn and S. P. Colombano, "A circuit representation technique for automated circuit design," *IEEE Trans. on Evol. Comp.*, 1999.

[5] J. R. Koza, F. H. Bennett III, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Trans. on Evolutionary Comp.*, vol. 1(2), 1997.

[6] T. R. Dastidar, P. P. Chakrabarti, and P. Ray, "A synthesis system for analog circuits based on evolutionary search and topological reuse," *IEEE Transactions on Evolutionary Computation*, Apr. 2005.

[7] A. Das and R. Vemuri, "Topology synthesis of analog circuits based on adaptively generated building blocks," in *Proc. of ACM/IEEE Design Automation Conference (DAC)*, Jun. 2008.

[8] ——, "An automated passive analog circuit synthesis framework using genetic algorithms," in *Proc. of IEEE ISVLSI*, March 2007, pp. 145–152.

[9] C. Zhao, J. Kong, J. Dong, and K. Zhang, "Pattern-based design evolution using graph transformation," *Journal of Visual Languages and Computing*, vol. 18, no. 4, pp. 378–398, Aug. 2007.

[10] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8(3), 1965.

[11] K. M. Passino and S. Yurkovich, *Fuzzy Control*. Addison-Wesley.

[12] A. Torralba, J. Chavez, and L. G. Franquelo, "FASY: A fuzzy-logic based tool for analog synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15(7), 1996.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[14] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. Addison-Wesley Prof., 1989.

[15] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw-Hill.

[16] T. McConaghy, P. Palmers, G. Gielen, and M. Steyaert, "Simultaneous multi-topology multi-objective sizing across thousands of analog circuit topologies," in *Proc. of Design Automation Conference*, 2007.

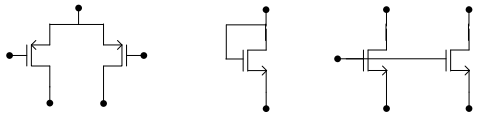


Fig. 10. VCO synthesis library blocks

Session 7B

Reliability and Design Space Exploration

RADJAM: A Novel Approach for Reduction of Soft Errors in Logic Circuits

Koustav Bhattacharya and Nagarajan Ranganathan
Department of Computer Science and Engineering
University of South Florida
Tampa, FL 33620
{kbhattac,ranganat}@cse.usf.edu

Abstract

The task of achieving reliability against transient faults poses a significant challenge due to technology scaling trends. Several optimization techniques have been proposed in the literature for preventing soft errors in logic circuits. However, most approaches for avoiding soft errors in logic circuits have significant overheads in terms of delay, area or power. In this work, we propose a circuit level technique called RADJAM (RADiation JAMmer) to prevent soft errors, occuring due to radiation strikes, in logic cells [17]. The RADJAM circuit when inserted at the output of a logic can reduce the generation of transient glitches significantly. Further, we propose an algorithm to insert RADJAM cells on selective nodes in a logic circuit. The algorithm uses signal logic probabilities and circuit slack for insertion of RADJAM cells on circuit nodes, thus improving the reliability of the logic circuit with minimal impact on the overall circuit delay. The proposed algorithm has been implemented and validated on the ISCAS85 benchmarks. Experimental results indicate that RADJAM optimized logic circuits can reduce soft error rates by around 39% with marginal delay, area and power overheads.

1 Introduction

The trends in technology scaling have made nanometer designs highly susceptible to transient faults. Transient faults occur due to several reasons, such as soft errors, power supply and interconnect noise, and electromagnetic interference. Soft errors occur when the energetic neutrons coming from space or the alpha particles arising out of packaging materials hit the transistors. A soft error may manifest itself as a bit flip in a latch or memory element. Additionally, soft errors can occur in any internal node of a combinational logic and subsequently propagate to and be captured in a latch. Although, soft errors have been a greater concern for memory elements, technology trends like smaller feature sizes, lower voltage levels, higher operating frequency and reduced logic depth, are projected to increase the soft-error rate (SER) in combinational logic beyond that of unprotected memory elements [13, 3]. In a recent study [9], the SER of logic circuits were quantified in technology nodes from 600nm to 50nm and it was projected that by 2011, the SER in logic circuits will increase by nine orders of magnitude and will essentially be comparable to unprotected memory.

Several approaches have been proposed in the literature to protect logic circuits against soft errors. In [12], time redundancy is exploited to detect and recover from soft-errors. In [6], concurrent error detection circuits are added to nodes in logic circuits which have high soft error susceptibility. However, such approaches for soft error *detection and correction* by using spatial and temporal redundancy typically incur huge area, power and delay overheads. Some of works reported in literature have proposed methods to *prevent* the generation of transient faults by sizing the individual gates of a logic circuit. In [11], assymetric logical masking probability of nodes in a logic circuit is exploited to selectively resize gates. In [1], an optimization framework based on geometric programming is used for simultaneous dual-VDD assignment and sizing. In [8], the authors have proposed a technique to reduce SER in logic circuits by simultaneous sizing and flip-flop selection. Although the achieved reduction in SER rate is quite high, the area, power and delay overheads for such reliability-centric sizing technique is quite high. Relatively lower overhead SER reduction can be achieved by circuit level optimization techniques by selectively hardening circuit nodes against charges deposited by radiation strikes [13]. In [15], gates are locally duplicated and the duplicated gate outputs are connected by a voltage clamper circuit. This prevents the output node of the gate and its duplicate not to deviate in voltage due to a radiation strike. In [14], the logic gates that are affected by radiation strikes are isolated by using complimentary pass gates. The complimentary pass gates act as a low pass filter and filter out transient voltage pulses due to a radiation strike. In [16],

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.76

a class of soft error masking circuits is proposed using the schmitt trigger circuit. However, most of these works still achieve reduction in SER at the cost of high overheads in terms of delay, area or power.

In this work, we have developed a transistor level circuit called RADJAM which significantly reduces the generation of random glitches due to radiation strikes. Based on this, we have developed an algorithm for selective insertion of RADJAM cells on critical circuit nodes. Experimental results indicate that our RADJAM based methodology can reduce the SER in logic circuits by about 39% with overheads of only 4%, 7% and 6% in delay, area and power respectively. The rest of the paper is organized as follows. In section 2, we discuss the preliminaries of soft errors in logic circuits. In Section 3, we present RADJAM, a transistor level circuit that can reduce propagation of transients due to radiation. Section 4 describes an algorithm to selectively insert RADJAM cells on circuit nodes for low overheads in delay, area and power. Section 5, describes our experimental setup and illustrates the results. Finally, Section 6 concludes the paper.

2 Soft Errors in Logic Circuits

The occurences of random radiation induced energetic neutron strikes are generally distributed fairly uniformly in space and time. The probability of a particle strike in a circuit node is thus roughly proportional to its active area. The charge deposition at a particular circuit node is traditionally modeled by a double exponential current pulse $I_{in}(t)$ [4], which can be represented as,

$$I_{in}(t) = \frac{Q}{\tau_\alpha - \tau_\beta}(e^{-\frac{t}{\tau_\alpha}} - e^{-\frac{t}{\tau_\beta}}) \quad (1)$$

where Q is the charge deposited as a result of a particle strike, τ_α is the collection time-constant of the junction, and τ_β is the ion-track establishment time constant. τ_α and τ_β are generally defined by process parameters. A threshold critical charge, Q_{crit}, marks on the onset of the double exponential current pulse behavior described above. Though the characteristics of a transient pulse at a node depends on the energy distribution of the incident particle, the drive strength of the gate, and the critical charge, various masking factors determine whether the transient pulse can actually propagate to the primary outputs/latches/flip-flops and result in a soft error. The three primary factors that can potentially mask radiation induced transients are as follows,

- **Logical masking** occurs when there is no sensitized path from the gate node where the transient pulse occurs to any of the primary outputs. The transient pulse is filtered when it arrives to an input of a gate whose any of the other inputs are at a controlling logic value.
- **Electrical masking** occurs due to electrical attenuation of the transient pulse in a sensitized path, from its occurence at a particular gate node to any of the primary outputs. Thus, the extent of electrical masking depends on the electrical property of the gates in the sensitized path.
- **Timing-window masking** occurs when the transient pulse does arrive at the primary outputs with sufficient strength to cause a soft error but is sufficiently separated in time from the arrival of the clock edge. As the latch only samples its input on the clock edge, and as the transient pulse is momentary it does not effectively lead to a soft error.

These masking effects thus makes various internal circuit nodes to be quite different in their susceptiblity to soft errors [11]. We show in Section 4, how this asymmetric distribution of masking probability can be used to optimize only selective nodes of a logic circuit. Soft error rates also depend on environmental factors like altitude, however, we do not model this in our formulation.

3 Radiation Jammer Circuit

In this section, we describe a circuit level technique for countering transient faults in a standard cell based design flow. The technique focuses on transistor level feedback to counter transient glitches due to radiation strikes occurring on the active area of the cells. The feedback circuitry which is attached to standard cell outputs, has been named "RADJAM" (RADiation JAMmer).

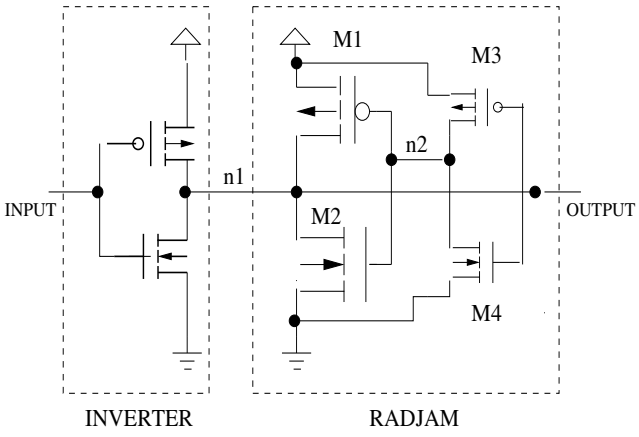


Figure 1. Schematic of a inverter cell protected with RADJAM

The transistor level schematic of a basic inverter protected against transient faults with a RADJAM cell is shown in Figure 2. As shown in the figure, the RADJAM circuit

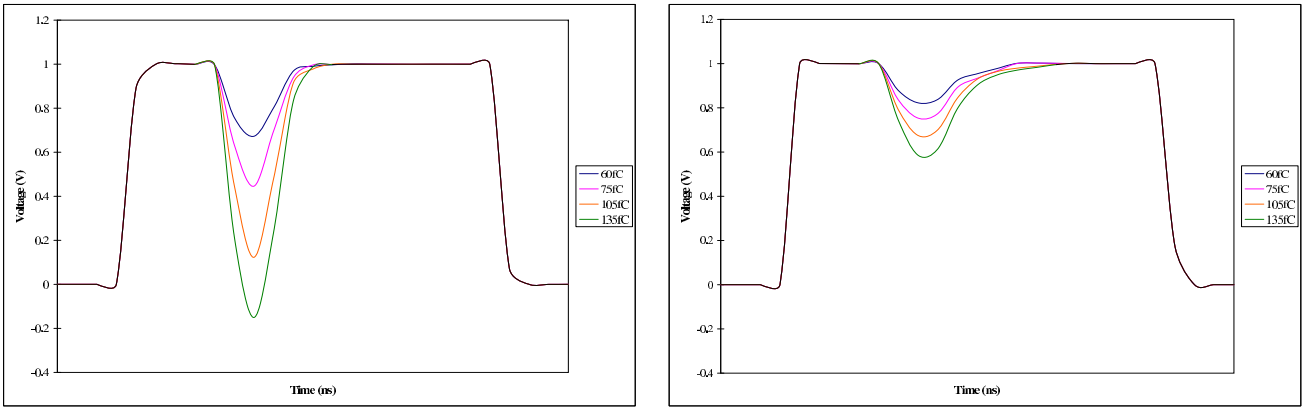


Figure 2. (A) Transient pulses on inverter cell due to radiation strikes of varying strength, (B) Corresponding results on an inverter cell with RADJAM

consists of 4 transistors M1-M4 which is arranged in the configuration of a cross-coupled inverter. The charge stored on gate capacitance of M1 and M2 connected to the node $n2$ represents the previous stable logic state of the cell. When a transient glitch appears on node $n1$, M3 and M4 tries to switch logic states. But M1 and M2 are still controlled by the huge charge stored due to the previous stable state. This in turn, thus helps M1 and M2 fight the transient glitch on node n1 by using the stable charge stored in the intermediate node. Thus, when a transient glitch occurs on the output of the inverter cell due to a radiation strike, the RADJAM cell prevents the transient glitch from affecting the output. The RADJAM circuit improves the effect of electrical masking by adding a small keeper in the intermediate node and hence adds memory to a memory-less node. In the RADJAM circuit, M1 and M2 are sized relatively high so that they have a larger stored charge from the previous stable state due to their high gate capacitances, while M3 and M4 are sized relatively low so that they have a small delay impact during a regular logic transitions. The exact sizes of the transistors M1-M4 are based on the characterization of each standard cell and are decided based on the size configuration that achieves the best reduction in transient pulses with the least delay penalty. Efficient layouts of each of the RADJAM protected standard cells were created by hand using the euler path method to provide as much single strip layout as possible.

We experimented with the RADJAM framework using the NSCU Process Design Kit and the OSU standard cell library using the TSMC 180nm technology. A subset of the library cells was selected and the extracted netlist from the layout were then simulated in SPICE with parasitics for the original standard cells and the standard cells with RADJAM. We modelled radiation strikes of deposited charges in the range of [60fC, 135fC] with current sources as defined in equation 1 with a τ_α of 10ps and τ_β of 5ps. The range of deposited charges were considered based on typical radiation flux at the sea-level [4]. We found that RADJAM is quite effective in reducing transient pulses due to radiation strikes on standard cells. Figure 3(A) illustrates the transient glitches generated due to radiation strikes of varying strength on an inverter standard cell. Figure 3(B) shows the corresponding results for inverter standard cell with RADJAM circuitry, which shows significant reduction in transient pulse in standard cells with RADJAM. It should be noted that RADJAM only filters out glitches, which occur due to finite excess charge generation, using the charge stored due to the previous stable state. Ordinary, logic transitions have a steady source of charge which eventually flips the output state after the stored charge, due to the previous stable state, has been removed.

4 Selective Insertion Algorithm

RADJAM through effective in reducing transients due to radiation strikes has a high impact on delay and area on the standard cell. Thus, protecting all standard cells in a circuit with RADJAM may nullify the SER savings due to significant overheads in delay, area and power. In this section, we therefore propose an algorithm for selective insertion of RADJAM cells on circuit nodes to provide a low overhead technique for reduction in circuit SER.

We capture the asymmetric soft error susceptibility by computing a *cumulative probability for observability* (CPO). The CPO of each net is computed using a backward traversal of the structural netlist from the primary output towards the primary inputs. The CPO of the primary outputs are initially set to 1. The CPO of each input of a gate is calculated recursively by multiplying the CPO of its output net

Table 1. Experimental Results for RADJAM based SER optimization

Benchmark	% Reduction in SER	% Delay Overhead	% Area Overhead	% Power Overhead
c17	49.24	1.43	5.96	4.27
c432	36.74	6.96	5.77	6.17
c499	38.53	3.55	8.10	7.30
c880	35.28	6.40	9.90	8.31
c1355	35.84	3.83	6.82	6.29
c1908	41.69	1.33	7.44	5.84
c2670	39.19	8.81	6.09	4.97
c3540	38.67	11.1	8.10	7.27
c5315	36.02	5.03	10.9	11.2
c6288	34.56	4.46	7.30	6.71
c7552	39.88	2.44	6.36	5.99
AVG	38.69	4.45	7.52	6.76

with the product of the probabilities of all *other* inputs to the gate being at its enabling value. The enabling value of a gate input is the logic value that makes switching at other inputs of a gate to be observable at the gate output. The CPO of the stem of a fanout node is computed by considering the maximum CPO of all its branches. Note that, as the CPO values are computed using logic probabilities and CPO values progressively decrease at lower logic depths, the CPO accurately captures both electrical and logical masking effects in a circuit at the logic level.

A combinational circuit without feedback can be modelled as a directed acyclic graph (DAG). The DAG can be made polar by assigning a dummy source node connected to all primary inputs and a sink node connected to all primary outputs. The earliest arrival time (EAT) of each each net can now be computed by traversing the DAG in the topologically sorted order from the source and assigning the EAT of a gate output as the maximum of the EATs of its inputs plus the delay of the gate. Similarly, the latest arrival time (LAT) of each net can be computed by traversing the DAG in the topologically sorted order from the sink and assigning the LAT of a gate input as the minimum of the LATs of its outputs minus the delay of the gate. The difference of the LAT and the EAT provides the slack for each net.

The *probability for RADJAM insertion* (PSI) of each net is now computed by taking the product of the slack and the CPO for each net. A higher value of PSI of a gate output indicates that the corresponding net has a high slack and is highly susceptible for soft errors upsets at the registers/primary outputs due to radiation strikes on the active area of the gate. We select a set of $M\%$ of the gate nodes, G_M, by sorting the various gate output nets based on its PSI values. We experimented with different values of M and we found that setting the value of M to be of 20% best optimizes reduction in SER with minimal delay, power and area overhead. RADJAM cells are then inserted at the output of these G_M gate nodes. This ensures that RADJAM cells protect only nodes on the non-critical path, but those which are highly susceptible to generation and propagation of soft errors.

5 Experimental Results

The proposed algorithm was implemented on 1.5Ghz UltraSparc processor with 4GB of memory and running SunOS 5.8. The results were validated using the ISCAS'85 benchmark circuits. We have used the NSCU Process Design Kit and a subset of cells of the OSU standard cell library based on the TSMC 180nm technology [18]. Synopsys Design Compiler was used to do the initial technology mapping and for computing the enabling probability of the nets. Many soft error estimation tools have been reported in literature [10, 2, 7]. The SEAT-LA tool [7] models the entire spectrum of neutron strikes (from charge values in the [10fC,150fC] range) and is quite close in accuracy to actual SPICE simulations. We extended this tool for our SER estimation based on Monte Carlo Simulations. Our simulation flow is illustrated in Figure 4.

In Figure 3, we illustrate the results of SER reduction on ISCAS85 benchmarks for our SER reduction algorithm for the original structural netlist and the structural netlist obtained by selective RADJAM insertion. We performed Monte Carlo simulation runs until convergence was achieved, and found that on average the SER was reduced by around 39% over the unoptimized structural netlist. We also computed the delay, area and power overhead of the RADJAM based SER optimization technique, as the percentage increase in the metrics in the the RADJAM optimized circuit compared to the unoptimized circuit. As shown in Table 3, on average the delay, area and power

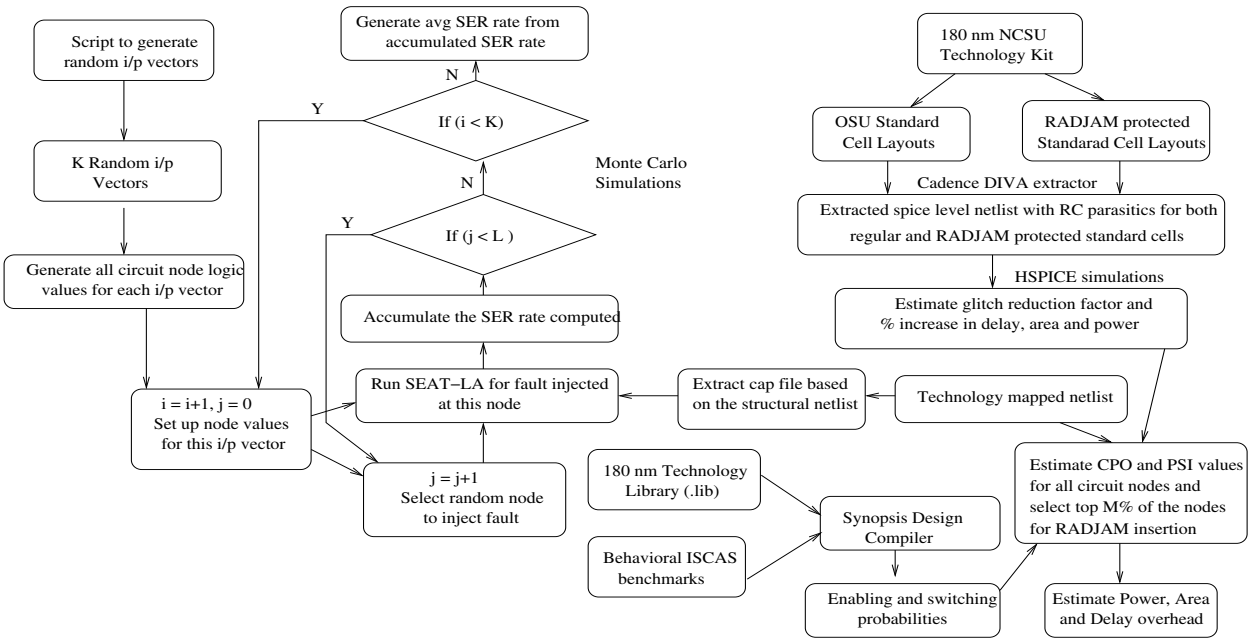


Figure 4. RADJAM based SER Reduction in Logic Circuits: Simulation Flow

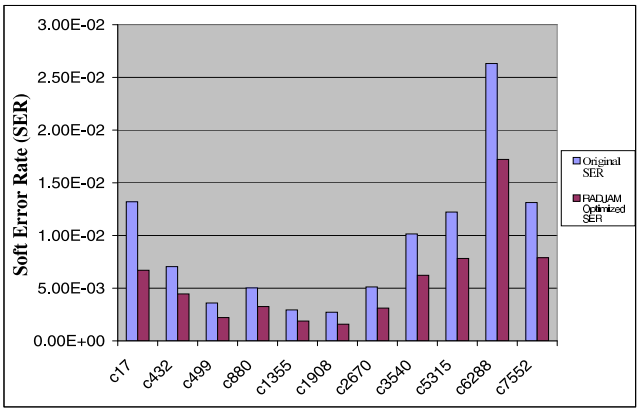


Figure 3. SER for ISCAS85 benchmarks

overhead for RADJAM based SER optimization was around 4%, 7% and 6% respectively. Thus, selective insertion of RADJAM cells as has been described in this work can achieve significant reduction in SER at quite low area, power and delay overheads.

We have also compared the overheads of our scheme with several recent works found in literature for SER reduction in logic circuits. The works report various SER reduction values and the area, power and delay overheads associated with the approach. We have fixed a common SER reduction rate for all approaches and compared the percentage increase in delay, area and power for each approach. We have used the data as reported in the results

Table 2. Comparison with related works

Scheme	% Delay Cost	% Area Cost	% Power Cost
Gate Sizing [11]	6.20	47.40	44.67
Shadow Gates [15]	5.12	56.84	41.45
Pass Gates [14]	15.00	15.00	22.00
Sizing/rad-hard FF [8]	5.00	40.00	37.83
Our approach	4.45	7.52	6.76

of the corresponding research papers and interpolated any missing data according to our simulation setup. The results of these comparisons have been summarized in Table 2. Selectively sizing gates of a circuit can achieve high reduction in SER [11]. However, in order to balance the load due to resizing a gate at the higher logic depths, the entire path must be resized. This results in huge area and power overheads. As shown in the table, the problem still persists when gate sizing is combined with radiation hardened flip-flop assignment [8]. Shadow Gates with diode clamper provides a clever way for hardening circuit nodes [15]. However, the duplication of entire cells lead to high area overheads. Moreover, due to process variations the duplicate gate may not have the exact delay as original gate and hence may affect the performance of the hardened standard cell. Also, the critical depth based selective insertion method that has been used in this work, is based only logic depth and hence

does not take into account any logical masking effects. On the other hand, our selective insertion models both logical and electrical masking effects and the the slack available at a node and hence incurs marginal delay and area overhead for similar reduction in SER. Complimentary pass gates can act as a low pass filter for glitches induced by radiation strikes [14]. However, the method cannot reduce the magnitude of transient pulses with moderate/high magnitudes. Hence, large sized pass gates or a chain of pass gates need to be used which, as shown in the table, makes the approach expensive in terms of area, power and delay for realistic radiation flux. As RADJAM is quite effective in filtering glitches of large amplitudes, our approach on the other hand, incurs much lower overheads for the same SER reduction.

6 Conclusions

We have developed a transistor level circuit which can significantly reduce soft error transients in logic circuits. The proposed RADJAM circuit could be expensive if applied blindly to all nodes on the circuit. Towards this, an intelligent algorithm is proposed which inserts RADJAM cells selectively on soft error vulnerable nodes in the non-critical paths of a circuit.

Acknowledgment

We would like to thank Dr. Vijaykrishnan Narayanan of Penn State University for generously allowing us to use the SEAT-LA tool for our study.

References

[1] M. Choudhury, Q. Zhou, and K. Mohanram. Design optimization for single-event upset robustness using simultaneous dual-vdd and sizing techniques. *Proc. of ICCAD*, pages 204–209, 2006.

[2] Y. Dhillon, A. Diril, and A. Chatterjee. Soft-error tolerance analysis and optimization of nanometer circuits. *Proc. of DATE*, pages 288–293, 2005.

[3] T. Karnik, B. Bloechel, K. Soumyanath, V. De, and S. Borkar. Scaling trends of cosmic ray induced soft errors in static latches beyond 0.18u. *Proc. of Symp. on VLSI Circuits*, pages 61–62, 2001.

[4] G. Messenger. Collection of charge on junction nodes from ion tracks. *Trans. of Nuclear Science*, 29(6):2024–2031, 1982.

[5] S. Mitra, T. Karnik, N. Seifert, and M. Zhang. Logic soft errors in sub-65nm technologies design and CAD challenges. *Proc. of DAC*, pages 2–4, 2005.

[6] K. Mohanram and N. Touba. Cost-effective approach for reducing soft error failure rate in logic circuits. *Proc. of ITC*, pages 893–901, 2003.

[7] R. Rajaraman, J. Kim, N. Vijaykrishnan, Y. Xie, and M. Irwin. SEAT-LA: A soft error analysis tool for combinational logic. *Proc. of VLSID*, pages 499–502, 2006.

[8] R. Rao, D. Blaauw, and D. Sylvester. Soft error reduction in combinational logic using gate resizing and flipflop selection. *Proc. of ICCAD*, pages 502–509, 2006.

[9] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi. Modeling the effect of technology trends on the soft error rate of combinational logic. *Proc. of DSN*, pages 389–398, 2002.

[10] B. Zhang, W. Wang, and M. Orshansky. FASER: Fast analysis of soft error susceptibility for cell-based designs. *Time*, 1(66):2–10, 2003.

[11] Q. Zhou and K. Mohanram. Gate sizing to radiation harden combinational logic. *Trans. on CAD*, 25(1):155–166, 2006.

[12] M. Nicolaidis. Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies. *Trans. on VTS*, 99:86–94, 1999.

[13] T. Karnik, S. Vangal, V. Veeramachaneni, P. Hazucha, V. Erraguntla and S. Borkar. Selective node engineering for chip-level soft error rate improvement. *Proc. of Symp. On VLSI Circuits*, pages 204–205, 2002.

[14] J. Kumar and MB. Tahoori. Use of pass transistor logic to minimize the impact of soft errors in combinational circuits. *Proc. of Workshop on SELSE*, 2005.

[15] R. Garg, N. Jayakumar, S.P. Khatri and G. Choi. A design approach for radiation-hard digital electronics. *Proc. of the DAC*, pages 773–778, 2006.

[16] Y. Sasaki, K. Namba and H. Ito. Soft Error Masking Circuit and Latch Using Schmitt Trigger Circuit. *Proc. of the Symp. on DFT*, 327–335, 2006.

[17] N. Ranganathan and K. Bhattacharya. Methodology and Apparatus for Reduction of Soft Errors in Logic Circuits. *US Patent*, Provisional Patent Application filed on June 13, 2008.

[18] J. Stine, J. Grad, I. Castellanos, J. Blank, V. Dave, M. Prakash, N. Iliev, and N. Jachimiec. A Framework for High-Level Synthesis of System-on-Chip Designs. *Proc. of Microelectronic Systems Education*, 11–12, 2005.

Soft Error Rates with Inertial and Logical Masking*

Fan Wang
Juniper Networks, Inc.
Sunnyvale, CA 94089, USA
fanw@juniper.net

Vishwani D. Agrawal
Auburn University, Dept. of ECE
Auburn, AL 36849, USA
vagrawal@eng.auburn.edu

Abstract

We analyze the neutron induced soft error rate (SER). An induced error pulse is modeled by two parameters, probability of occurrence and probability density function of the pulse width. We calculate failures in time (FIT) rates for ISCAS85 benchmark circuits. A comparison with measured SER for SRAMs shows better relevance of our work over other published work. Our CPU times are reasonable; benchmark circuit C1908 with 880 gates requires only 1.14 seconds. Further, we study the influence of circuit topology on SER. We find that for some circuits with many levels of logic there exists a critical single event transient (SET) width. For smaller induced pulse width the SER depends not on the size of the circuit but only on the gates near the output, and only those need to be protected. For an inverter chain in TMSC035 technology, the critical width is between 25ps and 50ps. For a shallow circuit, e.g., a ripple-carry adder, the critical SET width may not exist.

1 Introduction

Continuous downscaling of CMOS technologies has resulted in clock frequencies in the multiple gigahertz range, supply voltage below one volt and load capacitances of circuit nodes dropping to femtofarads. As a result, soft error rates in logic and processor circuits are increasing. In addition, if other circuit noises such as interconnect coupling and ground bounce are also considered soft errors, the logic FIT (*failure in time*, 1 FIT = 1 failure in 10^9 hours) rate is expected to increase faster and become comparable to the FIT rate of memories [9]. The SER due to high-energy neutrons in SRAM cells, latches, and logic circuits for feature sizes from 600nm to 50nm have been reported [25]. According to that study, the SER of logic circuits is expected to increase nine orders of magnitude from 1992 to 2011, becoming comparable to the failure rate of unprotected on-chip memories.

Well-known noise sources include noisy power supply, lightning, electrostatic discharge (ESD), ground bounce, and interconnect coupling capacitances. With advances in the design and manufacturing technology, such non-environmental conditions may not remain the dominant influence on the sub-micron semiconductor reliability. Errors caused by cosmic rays and alpha particles will become the prevalent reliability issue in electronic systems. A detailed discussion on the source of alpha particles and neutrons and their effects on electronics can be found in a recent tutorial paper [30].

In Section 2, we summarize the previous work on soft error rate estimation and in Section 3, we will review a novel environment dependent soft error model, which is based on both error occurrence rate represented as a probability, and the single event transient (SET) pulse density represented as a probability density function [28, 29, 31]. We favor this model because it includes both inertial and logic masking effects inherent in digital circuits. In Section 4, we compare analysis results with relevant published work. We discuss various key factors that may influence logic SER. Some of those factors are barely considered in existing logic SER estimation work. In Section 5, we study the influence of circuit topology on soft error rate.

2 Previous Work

Soft-error studies have traditionally been experimental where one uses an accelerated life environment for a VLSI device [12]. Typically, a neutron beam accelerator may be used. An alternative is a real life environment where a tester evaluates the failure rate for hundreds of chips at nominal conditions. Though field testing is very expensive and takes up to a year to obtain reliable results, it is important to validate the accelerated testing assumptions. The long delay in getting the SER results is often unacceptable for a contemporary chip market. Alternative is either a costly test of more chips with bigger tester or deviation from the nominal conditions to more sensitive ones [35]. For example, the test facilities in the Jungfraujoch lab in Switzerland located at 11,000 feet can accelerate ground-level test times by a factor of 11. In this lab, *iRoC Technologies* obtained a statistically significant number of soft errors on several different devices over a period of 4 to 6 months [16].

The JEDEC (*Joint Electron Device Engineering Council*) standard includes JESD89, JESD89-A [10] and JESD89-2. In JESD89 [10], the standard specifications cover soft errors due to alpha particles and atmospheric neutrons. These standards specify that the SER data obtained from alpha accelerated SER tests should be extrapolated to an alpha flux of 0.001 particles/hr-cm^2. For example, the neutron accelerated SER (*ASER*) test results have been extrapolated to the typical neutron flux observed at New York City. For energy in the range of 10–10000 MeV, the neutron flux is 3.9×10^{-3} N/cm^2-s and when the energy range

*This research is supported in part by the National Science Foundation Grant CNS-0708962.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.77

is from 1 to 10 MeV, the neutron flux is 4.0×10^{-3} N/cm^2-s [8, 10]. Primarily, the procedures apply to memory devices like DRAMs and SRAMs, but with some adjustments can be used for logic devices [10].

The existing computer programs to model the single event effects (*SEE*) on electronics include SEMM, developed by IBM [26]; CRIME, supported by U.S. Air Force and Office of Naval Research grant [4] and CREME96 from Naval Research Laboratory [27].

Unlike memories, soft errors in logic circuits may be filtered out by the circuit itself and may not effect the circuit performance. This is known as logic masking, electrical masking or temporal masking [18]. Also, the complex topology of a logic circuit is different from memory's regular structure.

Asadi *et al.* [2] present a soft error rate estimation technique based on error probability propagation. Rejimon and Bhanja [24] gave a single event fault model based on probabilistic Bayesian networks, which captures spatial dependencies. These approaches do not take the circuit electrical masking factor and the characteristic of transient pulses like pulse widths into account. An improvement was provided by Zhao *et al.* [33], who proposed a constraint-aware robustness insertion methodology to protect the sequential elements in digital circuits against various noise effects. However, the authors did not include the environmental factors like the error frequency. Besides, their propagation method required tabulating all the pulse width and height data for each logic gate. It would thus take enormous memory for large logic circuits. Other notable logic circuit SER estimation work includes SEAT-LA [21], SERA [32] and an algorithm by Rao *et al.* [23].

Mohanram and Touba [17] gave a cost effective approach to selectively protect high susceptibility nodes in logic circuits. A recent paper [15] proposed symbolic approaches using binary decision diagrams, algebraic decision diagrams and a probabilistic model for sequential SER analysis. Rewriting, with optimization for area and power consumption, can also be used for reducing the soft error rate [1].

3 Environment-Based Model

Different from memories, in a logic circuit, a single event effect exists as a single event transient (SET) pulse. An SET has its unique characteristics like polarity, waveform, amplitude and duration, and these characteristics depend on particle impact location, particle energy, device technology, device supply voltage and output load. A single event upset does not occur unless the SET can survive the circuit masking effects and is captured by a clock edge of some sequential element [16].

Environmental neutrons come from cascaded interactions when galactic cosmic rays traverse through earth's atmosphere. These neutrons reach the ground with finite probabilities [19]. The intensity of cosmic-ray induced neutrons flux in the atmosphere varies with altitude, location in the geomagnetic field, and solar magnetic activity. The flux rate data is available from the reported measurement records over decades [14]. Not every particle hit on the sensitive silicon area can induce an error. An SEU occurs with certain probability for each high-energy particle hit. Such probability can be obtained from existing computer programs, for example, IBM's SEMM (Soft Error Monte-Carlo Modeling) program [26].

We consider all energy components in the proposed soft error model. We average the error probability over different energies and assign each circuit node with a unique error occurrence probability.

The particle energy distribution at any specific geographic locations for any specific technology can be obtained from experimentally measured results. For example, the cosmic particle strikes were simulated using a heavy ion beam at the Twin Tandem Van de Graaff accelerator of the Brookhaven National Laboratory. Those results suggest that in the natural environment of space the probability distribution of high-energy particles falls rapidly with increasing energy. For 0.5μ and 0.35μ CMOS technology processes at the ground level, the largest population has an linear energy transfer (LET) of $20MeV\text{-}cm^2/mg$ or less and the particles with LET greater than $30MeV\text{-}cm^2/mg$ are exceedingly rare [7]. LET determines the ionization energy and hence the charge collection as the particle traverses through the material of a switching device.

The transient current pulse created by a striking particle with given LET has been represented by a double exponential expression [13]:

$$\begin{cases} I(t) = \frac{Q_{coll}}{\tau_\alpha - \tau_\beta}(e^{-\frac{t}{\tau_\alpha}} - e^{-\frac{t}{\tau_\beta}}) & \text{(a)} \\ Q_{coll} = 10.8 \times L \times LET & \text{(b)} \end{cases} \quad (1)$$

where Q_{coll} is the collected charge in the sensitive region, τ_α is the collection time constant, which is a process-dependent property of the junction, and τ_β is the ion-track establishment time constant, which is relatively independent of the technology. In bulk silicon, the typical charge collection depth (L) is 2μ. For every 1 $MeV\text{-}cm^2/mg$, and an ionizing particle deposits about $10.8fC$ (femtocoulomb) charge along each micron on its track. Typical values are approximately $1.64 \times 10^{-10} sec$ for τ_α and $5 \times 10^{-11} sec$ for τ_β from measurements [3, 33].

By charging and discharging the circuit node capacitance, the single event transient current pulse is converted into a transient voltage pulse in Figure 1. Figure 2 gives a neutron-induced soft error model for logic circuits. Because the probability per hit is related to the neutron flux which is location dependent, we can easily get the circuit SER in units of *FIT* for different locations if the corresponding neutron flux data is available.

In summary, this probabilistic soft error model is based on two considerations: (1) the SEU occurrence rate, represented as probability and (2) once an SEU occurs, it exists in the logic circuit as SETs with a random pulse width characterized by a probability density function [28, 31].

4 Simulation Results

We compare new simulation results with the relevant published work and discuss various key factors

Table 1. SEU error rate (SER) analysis of ISCAS85 benchmark circuits.

Circuit	# PI	# PO	# Gate	Our approach [28, 31]		Rao et al. [23]		Rajaraman et al. [21]	
				CPU s	SER (FIT)	CPU s	SER (FIT)	CPU min.	Error Prob.
c432	36	7	160	0.04	1.18×10^{3}	<0.01	1.75×10^{-5}	108	0.0725
c499	41	32	202	0.14	1.41×10^{3}	0.01	6.26×10^{-5}	216	0.0041
c880	60	26	383	0.08	3.86×10^{3}	0.01	6.07×10^{-5}	102	0.0188
c1908	33	25	880	1.14	1.63×10^{4}	0.01	7.50×10^{-5}	1073	0.0011
Computing platform				Sun Fire 280 R		Pentium 2.4GHz		Sun Fire v210	
Circuit technology				TSMC035		Std. 0.13μm		70nm BPTM*	
Altitude				Ground		Ground		N/A	

Table 2. Comparison with measured data.

(Measured Data) (Altitude Unknown)		(Estimated Logic Circuit SER) (Ground Level)	
Devices	SER (FIT/Mbit)	Our Work (FIT)	Rao et al. [23] (FIT)
0.13μm SRAMs [6]	10,000 to 100,000	1,000 to 20,000	1×10^{-5} to 8×10^{-5}
SRAMs, 0.25μm and below [11]	10,000 to 100,000		
1 GBit memory in 0.25μm [20]	4,200		

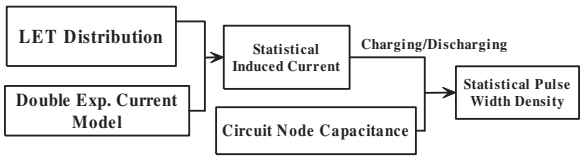


Figure 1. Transforming statistical neutron energy spectrum to SET width statistics.

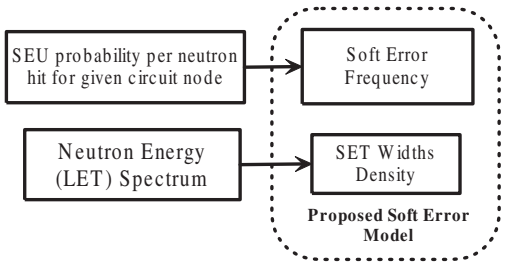


Figure 2. Proposed probabilistic neutron induced soft error model for logic.

that may influence logic SER.

For a detailed algorithm to propagate soft errors through elementary logic gates and the algorithm to calculate circuit SER, one may refer to recent publications [28, 31]. We simulated ISCAS85 benchmark circuits by a simulator developed in C programming language. We assume that all circuits are working at the ground level and the probability of SEU per particle hit is 10^{-4}. We have neglected the polarity of SETs and the temporal masking factor. At ground level we use the neutron energy statistics obtained from [7] assuming the SET width density per circuit node follows a *normal* distribution with mean $\mu = 150$ and standard deviation $\sigma = 50$. These assumptions are justified for relatively small values of particle flux and small chip area. From [34], the total neutron flux at sea level is $56.5m^{-2}s^{-1}$. For a CMOS circuit in TSMC035 technology, we assume the sensitive region to be $10\mu m^2$ for each circuit node. For a circuit with n primary outputs and m nodes, the circuit SER is $\Sigma_{i=0}^{n}(\Sigma_{j=0}^{m} SER_{i\_caused\_by\_j})$ which is different from [31], in which we calculated the SER per gate per output $(\frac{1}{n}\Sigma_{i=0}^{n}(\frac{1}{m}\Sigma_{j=0}^{m} SER_{i\_caused\_by\_j}))$. The unit for SER is FIT.

In Table 1, we compare these results for several benchmark circuits with available results; not all benchmark circuit SER results have been published. Our results have several orders of magnitude difference from the results of Rao et al [23] and the cause of this huge discrepancy will be discussed in the following section. The term BPTM marked with asterisk (*) stands for Berkeley Predictive Technology Model. The run times of our approach appear acceptable. For example, for C1908 with 880 gates, the simulation run time is only 1.14 second.

Field test data for logic circuits is largely unavailable but the actual neutron experiments on a test chip in the future will help validate our analysis. However, the measured SER data for memories, both SRAM and DRAM, is available. Table 2 shows our results, estimated logic SER from Rao et al. [23], and the reported SRAM SER measurement data [6, 11, 20]. Clearly, our results show better relevancy with the measured SRAM SER. In Table 3, we compare the proposed approach with previous relevant works on logic soft error rate estimation [2, 21, 23, 24, 32]. We observe that none of the existing logic SER estimation work has considered the re-convergent fanout, which may have a significant influence on the analysis. We will further discuss these factors in the next section.

From the specification or experimental setup comparison presented in Table 3 we find that to accurately calculate logic SER, factors that influence logic SER estimation should be comprehensively considered. However, no analysis has considered all of them. Consider the following:

(1) The physics of the SEU phenomena seems involved. For example, the analysis of the funneling and the angle of incidence are not considered. We take the energy of neutrons to be the main source that induces the SEU. In reality, it is the physics of interaction be-

Table 3. Comparison of SEU error rate (SER) estimation methods.

Authors and Reference	Factors considered							
	LET Spectrum	Re-conv. Fanout	Sensitive Regions	SEU prob.	Vectors Applied	Location Altitude	Circuit Tech.	SET Degradation
Our work	yes	no	yes	yes	no	yes	yes	yes
Rao et al. [23]	yes	no	no	no	yes	yes	yes	yes
Rajaraman et al. [21]	no	no	no	no	yes	no	no	yes
Asadi-Tahoori [2]	no	no	no	yes	no	no	no	no
Zhang-Shanbhag [32]	yes	no	yes	yes	yes	yes	yes	yes
Rejimon-Bhanja [24]	no	no	no	yes	yes	no	no	no

tween neutrons and silicon that produces the SEU. Simpler modeling and assumptions may influence the SER estimation accuracy.

(2) The sensitive region of a transistor is defined as the channel region of an off nMOS transistor or the drain region of an off pMOS transistor. For a CMOS circuit, the "on" or "off" status of transistors is determined from inputs. We statically assume that each circuit node's sensitive region is $10\mu m^2$. This may bias the SER result. Although we have considered the sensitive node areas, the strikes on pMOS or nMOS nodes also influence the polarity of the SET. Thus, the dynamic state of the circuit is important.

(3) Compared to the earth's surface, the size of the sensitive region of a single transistor or a circuit board is trivially small and continues to reduce with the technology trend. At the surface of the earth we take the probability of a particle strike to a sensitive node simply by taking the ratio of the number of particle strikes per μm^2-s to strikes per m^2-s. Because 1 m^2 equals 10^{12} μm^2, most probably there will be no strike on the sensitive regions though such low probability events cannot be neglected. Once the SEU occurs, the SER may easily be several orders of magnitude higher compared to the case of no strike at all. For example, 1 SEU in 6 months (4320 hours) would be measured as 231,480 FIT. On the other hand, a 0 SEU in those 6 months will measure as 0 FIT.

(4) For logic circuits fan-out details should be considered. Our analysis only considers the worst case error rate for re-convergent fan-outs. For example, if a re-convergent fanout has two paths, and one passes through more gates than the other, our program only takes the path that has fewer gates because it is likely to give a higher SER. Timing and logic simulation of all paths would be needed for better accuracy [5]. Two situations can arise as shown in Figure 3:

(a) When SET goes through a high fan-out node, the large load capacitance can eliminate the SET through node inertia.

(b) Or if the SET is not canceled by the fan-out node, it goes through multiple fan-out paths. If all paths have equal length, the SET might cancel itself at a re-converging point depending on path inversions. However, in general, one SET on the affected node can cause several propagating SETs to further increase the SER of the circuit.

Path delays may also influence logic SER.

(5) It is highly recommended to have more field tests for logic circuits. Also, we suggest that the SER results from field tests for the same circuit, even in the same working environment, may be widely different at different times. Still, with field test data, the logic circuit SER results can be validated. A comparison with measurement may be the only way to determine which factors can be really neglected and which assumptions and approximations are justified.

(6) None of the SER estimation approaches consider process variation effects, which may also be a factor in the vulnerability to transient errors. It is reported that, intra-die process variation of threshold voltage may result in SER variation of 41% in a small circuit [22].

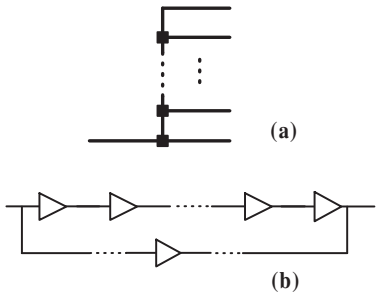


Figure 3. Circuit fanout stem and reconvergence of paths with different lengths.

5 Circuit Topology and SER

The circuit topology is an important factor that influences the logic SER. Deep and shallow circuits, having many or few levels, respectively, may have quite different masking effects on SER. Besides, the operating environment, technology, node capacitances and supply voltage are the other factors that should be considered. Once these factors and their weights are taken into account, we will have information to decide whether and how SEU protection methods should be applied. A good analysis will give circuit engineers opportunities to choose the most economic protection technique instead of the costly traditional methods like triple modular redundancy (TMR).

We analyzed examples of the two types of circuits mentioned above, an inverter chain as a deep circuit and a ripple carry adder as a shallow circuit. To apply the SER estimation method of our recent paper [28, 31], we assumed the working environment is the ground-level. The simulation results are shown in Figure 4. Figures 4 a(1) and b(1) show the circuit schematics. In the inverter chain each gate has

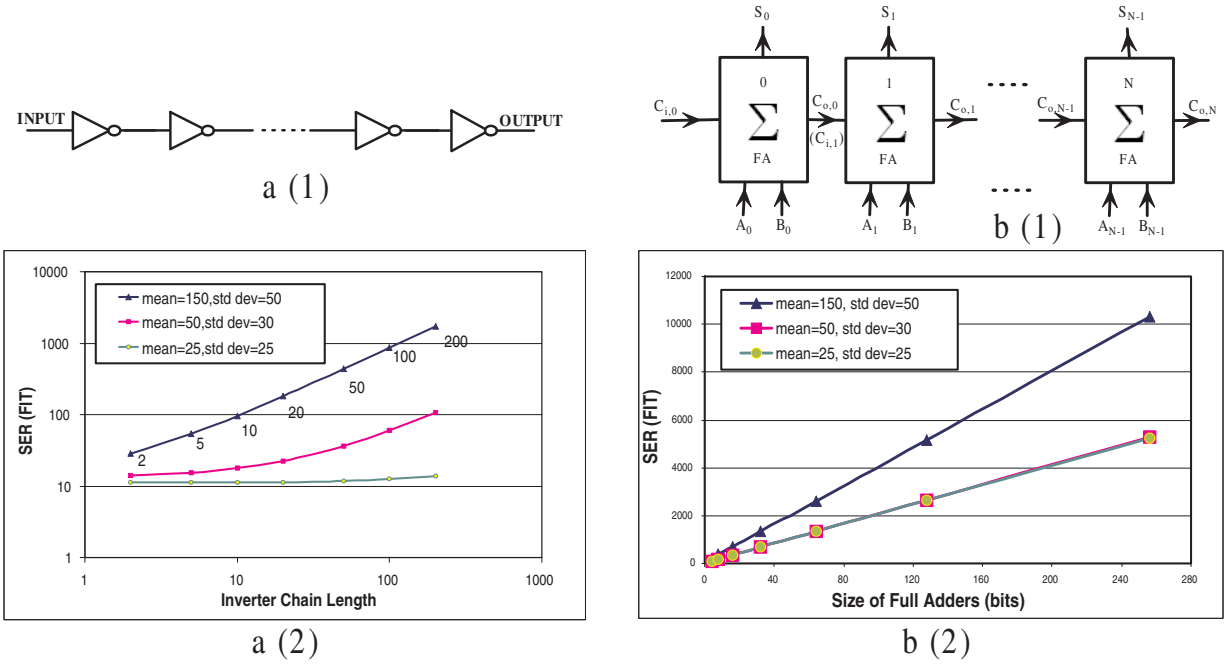


Figure 4. SER for circuits with different topologies: an inverter chain structure, a(1), and its SER simulation, a(2); a ripple-carry adder, b(1), and its SER simulation, b(2).

only one input and one output. Thus, there can be only electrical masking and no logic masking. The ripple carry adder has a parallel topology. With the exception of the rippling carry signal, all other inputs (A_n and B_n) propagate through the same number of gates to arrive at output (Sum_n). Figure 4 a(2) shows the estimation results for different lengths of inverter chains, with three different induced SET width distributions. For the two wider pulse widths (in picoseconds) whose normal distribution mean and standard deviation are (150, 50) and (50, 30), respectively, the SER increases almost linearly with the chain length. In the absence of logic masking, longer the chain, larger is the probability the circuit can be affected by the radiation source. For these two cases the pulse widths are large enough compared to the inertial delay thus survive the electrical masking. When the neutron flux density (mean, standard deviation) is (25, 25), the circuit SER remains fixed even with the increasing chain length. This is because the majority of induced SET pulse widths are small enough so the electrical masking is able to filter out these pulses. We can, therefore, determine a critical pulse width for the inverter chain as being somewhere between 25ps and 50ps. Majority of pulses of smaller widths are filtered out and the SER depends only on a few gates near the output of the chain. Only the gates near the primary output need to be protected against upset.

In Figure 4 b(2) the simulation results for ripple-carry adders are presented. For SET pulse width distributions (mean, standard deviation) of (50, 30) and (25, 25), the two SER curves are almost identical. Even with these SET pulse widths, the SER always increases with the increasing circuit size. This is caused by the parallel topology of the RC adder. Increasing number of gates will increase the cosmic ray hit probability thus causing the circuit SER to increase linearly with the circuit size. For ripple-carry adder, there is no critical SET pulse width below which the transients would be localized.

6 Conclusion

We have estimated the logic circuit SER based on an environment-dependent soft error model characterized by error occurrence rate and SET pulse width density. Results show better relevancy over other published work. Field test or accelerated test data on logic devices would be needed to further validate the accuracy of the our analysis. Our soft error rate estimation method requires logic signal probabilities. For any given set of input vectors or signal statistics, these may be obtained either from logic simulation or from static analysis. From our discussion, the logic SER may be highly sensitive to factors like sensitive region calibration, process variation and circuit characterization, making soft error estimation for logic circuits a complex problem. The influence of circuit topology on logic circuits is studied. We proposed a critical SET width such that for smaller pulse widths the SER does not increase with the increasing circuit size. However, for a shallow and wide circuit like a ripple-carry adder, the critical pulse width does not exist. More comprehensive studies should provide better insights for soft error protection schemes in the future.

References

[1] S. Almukhaizim, Y. Makris, Y. S. Yang, and A. Veneris, "Seamless Integration of SER in

Rewiring-Based Design Space Exploration," in *Proc. International Test Conference*, 2006, pp. 1–9.

[2] G. Asadi and M. B. Tahoori, "An Accurate SER Estimation Method Based on Propagation Probability," *Proc. Design Automation and Test in Europe Conf*, pp. 306–307, 2005.

[3] V. Carreno, G. Choi, and R. K. Iyer, "Analog-Digital Simulation of Transient-Induced Logic Errors and Upset Susceptibility of an Advanced Control System," in *NASA Technical Memo 4241*, 1990.

[4] D. L. Chenette, J. Chen, E. Clayton, T. G. Guzik, J. P. Wefel, M. Garcia-Munoz, C. Lopate, K. R. Pyle, K. P. Ray, E. G. Mullen, and D. A. Hardy, "The CRRES/SPACERAD Heavy Ion Model of the Environment (CHIME) for Cosmic Ray and Solar Particle Effects on Electronic and Biological Systems in Space," *IEEE Trans. on Nuclear Science*, vol. 41, no. 6, pp. 2332–2339, 1994.

[5] A. Dharchoudhury, S. M. Kang, H. Cha, and J. H. Patel, "Fast Timing Simulation of Transient Faults in Digital Circuits," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 1994, pp. 719–722.

[6] Graham, "Soft errors a problem as SRAM geometries shrink," http://www.ebnews.com/story/OEG20020128S0079, ebn, 28 Jan 2002.

[7] K. J. Hass and J. W. Ambles, "Single Event Transients in Deep Submicron CMOS," in *Proc. 42nd Midwest Symposium on Circuits and Systems*, 1999.

[8] T. Heijmen and A. Nieuwland, "Soft-Error-Rate Testing of Deep-Submicron Integrated Circuits," in *Proc. Eleventh IEEE European Test Symposium*, 2006, pp. 247–252.

[9] B. Ingols and A. Rambaud, "iRoC Releases Robust SPARC Test Report," *http://www.us.design-reuse.com/news/news65.html*, 2002.

[10] JEDEC, "Measurements and Reporting of Alpha Particles and Terrestrial Comic Ray-Induced Soft Errors in Semiconductor Devices," *JESD89, August*, 2001.

[11] W. Leung, F.-C. Hsu, and M. E. Jones, "The ideal soc memory: 1t-sramtm," in *Proc. 13th Annual IEEE International ASIC/SOC Conference*, 2000, pp. 32–36.

[12] T. C. May, D. L. Crook, D. W. Gralian, R. A. Reininger, and R. C. Smith, "Soft Error Testing," in *Proc. International Test Conference*, 1980, pp. 137–150.

[13] G. C. Messenger, "Collection of Charge on Junction Nodes from Ion Tracks," *IEEE Trans. on Nuclear Science*, vol. 29, no. 6, pp. 2024–2031, 1982.

[14] G. C. Messenger and M. Ash, *Single Event Phenomena*. Chapman & Hall, 1997.

[15] N. Miskov-Zivanov and D. Marculescu, "Circuit Relaiability Analysis Using Symbolic Techniques," *IEEE Trans. on CAD*, vol. 25, no. 12, pp. 2638–2649, Dec. 2006.

[16] S. S. Mitra, N. Kee, and S. Kim, "Robust System Design with Built-In Soft-Error Resilience," *IEEE Design & Test Computers*, vol. 38, no. 2, pp. 43–52, 2005.

[17] K. Mohanram and N. A. Touba, "Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits," in *Proc. International Test Conference*, 2003, pp. 893–901.

[18] H. T. Nguyen and Y. Yagil, "A Systematic Approach to SER Estimation and Solutions," in *Proc. 41st Annual IEEE International Reliability Physics Symposium*, 2003, pp. 60–70.

[19] E. Normand, "Single-Event Effects in Avionics," *IEEE Transactions on Nuclear Science*, vol. 43, no. 2, pp. 461–474, 1996.

[20] OnlineResource, "Soft Errors in Electronic Memory - A White Paper," Technical report, Tezzaron Semiconductor, 2004.

[21] R. Rajaraman, J. S. Kim, N. Vijaykrishnan, Y. Xie, and M. J. Irwin, "SEAT-LA: A Soft Error Analysis Tool for Combinational Logic," in *Proc. 19th International Conference VLSI Design*, 2006, pp. 499–502.

[22] K. Ramakrishnan, R. Rajaraman, S. Suresh, N. Vijaykrishnan, Y. Xie, and M. J. Irwin, "Variation Impact on SER of Combinational Circuits," in *Proc. 8th International Symposium on Quality Electronic Design*, 2007, pp. 911–916.

[23] R. R. Rao, K. Chopra, D. Blaauw, and D. Sylvester, "An Efficient Static Algorithm for Computing the Soft Error Rates of Combinational Circuits," in *Proc. Conference on Design, Automation and Test in Europe*, 2006, pp. 164–169.

[24] T. Rejimon and S. Bhanja, "An Accurate Probabilistic Model for Error Detection," in *Proc. 18th International Conference on VLSI Design*, 2005, pp. 717–722.

[25] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic," in *Proc. International Conference on Dependable Systems and Networks*, 2002, pp. 389–398.

[26] G. R. Srinivasan, "Modelling the Cosmic Ray-Induced Soft-Error Rate in Integrated Circuits: An Overview," *Microelectronics Reliability*, vol. 37, no. 4, pp. 691–691, 1997.

[27] A. J. Tylka, J. H. Adams Jr, P. R. Boberg, B. Brownstein, W. F. Dietrich, E. O. Flueckiger, E. L. Petersen, M. A. Shea, D. F. Smart, and E. C. Smith, "CREME96: A Revision of the Cosmic Ray Effects on Micro-Electronics Code," *IEEE Transactions on Nuclear Science*, vol. 44, no. 6, p. 2150, 1997.

[28] F. Wang, "Soft Error Rate Determination for Nanometer CMOS VLSI Circuits," Master's thesis, Auburn University, Dept. of ECE, May 2008.

[29] F. Wang and V. D. Agrawal, "Probabilistic Soft Error Rate Estimation from Statistical SEU Parameters," in *Proc. 17th IEEE North Atlantic Test Workshop*, May 2008, pp. 86–92.

[30] F. Wang and V. D. Agrawal, "Single Event Upset: An Embedded Tutorial," in *Proc. 21th International Conference on VLSI Design*, 2008, pp. 429–434.

[31] F. Wang and V. D. Agrawal, "Soft Error Rate Determination for Nanometer CMOS VLSI Logic," in *Proc. 40th Southeastern Symposium on System Theory*, Mar. 2008, pp. 324–328.

[32] M. Zhang and N. R. Shanbhag, "A Soft Error Rate Analysis (SERA) Methodology," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2004, pp. 111–118.

[33] C. Zhao and S. Dey, "Evaluating and Improving Transient Error Tolerance of CMOS Digital VLSI Circuits," in *Proc. International Test Conference*, 2006, pp. 1–10.

[34] J. F. Ziegler, "Terrestrial Cosmic Rays," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 19–39, 1996.

[35] J. F. Ziegler and H. P. Muhfeld, "Accelerated testing for cosmic soft-error rate," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 51–63, 1996.

Accelerating System-Level Design Tasks using Commodity Graphics Hardware: A Case Study

Unmesh D. Bordoloi Samarjit Chakraborty
Department of Computer Science, National University of Singapore
E-mail: {unmeshdu, samarjit}@comp.nus.edu.sg

***Abstract*— Many system-level design tasks (e.g. timing analysis, hardware/software partitioning and design space exploration) involve computational kernels that are intractable (usually NP-hard). As a result, they involve high running times even for mid-sized problems. In this paper we explore the possibility of using commodity graphics processing units (GPUs) to accelerate such tasks that commonly arise in the electronic design automation (EDA) domain. We demonstrate this idea via a detailed case study on a general hardware/software design space exploration problem and propose a GPU-based engine for it. Not only does this problem commonly arise in the embedded systems domain, its computational kernel turns out to be a general combinatorial optimization problem (viz. the knapsack problem) which lies at the heart of several EDA applications. Our experimental results show that our GPU-based implementation offers very attractive speedups for this computational kernel (up to 100×), and speedups of up to 17× for the full problem. In contrast to ASIC/FPGA-based accelerators – since even low-end desktop and notebook computers are today equipped with GPUs – our solution involves no extra hardware cost. Although recent research has shown the benefits of using GPUs for a variety of non-graphics applications (e.g. in databases and bioinformatics), hardly any work has been done on harnessing the parallelism of GPUs to accelerate problems from the EDA domain. We hope that our results and the generality of the problem we address will motivate researchers from this community to explore the possibility of using GPUs for a wider variety of problems from the EDA domain.**

I. Introduction

Many system-level design tasks arising within electronic design automation (EDA), such as hardware/software co-design and schedulability analysis, involve one or more computationally expensive cores. In a typical iterative design flow, system designers repeatedly invoke design tools which run these computational cores as a part of their backend. Hence, the running times of these tools which often are in the tune of several hours critically impact their usability. In this paper we explore the possibility of using commodity graphics processing units (GPUs) to accelerate these computational kernels, and thereby improve the running time and usability of the design tools that use them.

There are two main reasons behind exploiting GPUs for such non-graphics related applications (in contrast to using, say, ASIC/FPGA-based accelerators): (i) modern GPUs are extremely powerful (high-end GPUs such as the nVIDIA GeForce 8800 GTX have a FLOPS rating of around 330 GigaFLOPS, whereas high-end general-purpose processors are only capable of around 25 GigaFLOPS) (ii) GPUs are now commodity items as their costs have dramatically reduced over the last few years. Hence, the attractive price-performance ratios of GPUs gives us an enormous opportunity to change the way design automation tools perform, with almost no additional cost. In fact, recent years have seen the increasing use of graphics processing units (GPUs) for different general-purpose computing tasks. These span across numerical algorithms [15], computational geometry [1], database processing [2], image processing [18], and bioinformatics [17]. On the other hand, in spite of a wide variety computationally expensive problems from the EDA domain that need to be regularly solved by design tools running on desktops and laptops equipped with high-end GPUs, the use of GPUs for accelerating such problems has not been sufficiently explored so far. In fact, apart from recent interests [9], [5] in this direction, there are no other known efforts.

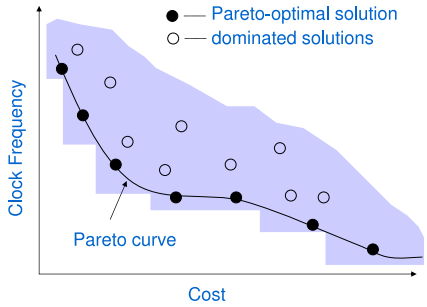


Fig. 1. Pareto-optimal solutions.

In this paper, we use a common design space exploration problem to establish the utility of the GPUs in accelerating system-level design tasks. We have chosen this problem mostly because of its generality. System designers today have a wide range of implementation possibilities ranging from fully-programmable processors, fixed-function components or hardware accelerators, to partially-programmable engines. Thus, they are faced with a large array of implementation possibilities each with different trade-offs involving costs, packaging constraints, and performance metrics (e.g. power consumption, delay and throughput). As a result, identifying only *one* implementation that meets the specified design constraints is no longer useful, but rather it is important to identify *all* implementations that expose the different possible performance tradeoffs (see Figure 1). Unfortunately, this *multi-objective* design space exploration often tends to be very tedious because of two reasons. First, a large number of designs need to be evaluated in order to cover the entire design space. Second, evaluating even a point in the design space may be computationally very expensive. Traditionally, researchers have been using different techniques to get around the high running times associated with such problems. The most notable amongst these are genetic algorithms [10] and approximation algorithms. However, these algorithms do not yield *exact* solutions and neither do they offer any performance guarantees.

Our contributions: From a computer architecture standpoint, GPUs naturally support what are referred to as *streaming algorithms* [22]. The main contribution of this paper is a reformulation of a standard design space exploration problem related to hardware/software partitioning [12] as a streaming algorithm that can be efficiently implemented on a GPU. Our results in this paper show that using GPUs may result in more than 100× speedup of the core of the design space exploration algorithm. These speedups will certainly improve the usability of a tool for such design space exploration, especially when used in an interactive fashion (i.e. where the designer

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.35

repeatedly makes small changes to the problem and invokes the tool until a satisfactory solution is obtained).

Our contributions are also significant because the core problem we solve is a general knapsack problem (viz. the *multiple-choice* knapsack problem). Given the generality of this combinatorial optimization problem – which arises in a variety of EDA tasks – we believe that our results might initiate an interest to explore the use of GPUs for accelerating other problems within the EDA domain as well.

Related work: Over the last two decades, numerous approaches have been proposed to accelerate computationally expensive algorithms arising in the EDA domain. Many of these approaches are similar to our work in the sense that they also exploit some form of parallelism in the application. The most notable approaches have used multiprocessors and reconfigurable hardware like FPGAs. However, none of them have explored the possibility of employing GPUs, which in contrast to FPGAs involve no extra hardware cost since most computing platforms today are equipped with GPUs. Further, high-level APIs and programming languages such OpenGL [19] and CUDA [4] have greatly simplified the task of programming GPUs.

Results reported in [21], [23] represent early efforts towards using multiprocessors to reduce computation time of EDA algorithms like VLSI routing. In [7] parallel algorithms for design space exploration to be run on a multiprocessor system have been described. More recently, [13] has proposed techniques for reducing simulation time by building simulation models for execution on multiprocessor systems. Further, the use of reconfigurable computing to accelerate problems from the EDA domain has been proposed in [20]. All of these proposals are for accelerating the Boolean SAT problem which also lies at the core of several EDA applications. Other efforts in this direction include hardware-based acceleration for fast simulation [11], [14]. Towards this, hardware acceleration is used to offload compute-intensive tasks from the software simulator.

In contrast to the above threads of work, the main advantage of our approach stems from the low cost associated with GPU-based acceleration since all desktop and notebook computers are now invariably equipped with GPUs. Hence, no extra hardware investment is necessary and EDA design tools can seamlessly incorporate our technique in a manner that is completely transparent to the end-user or the design engineer using the tool. Further, as mentioned above, with the development of high-level APIs and programming languages for graphics programming, it is now easy to exploit GPUs to accelerate the back-end of any EDA design tool with relatively low additional programming effort and graphics-specific knowledge.

Problem overview and our scheme: Assume a real-time application where two tasks T_1 and T_2 are required to run concurrently and have predefined deadline constraints. Both T_1 and T_2 can be partially implemented in hardware, with their remaining parts implemented as software running on the same programmable processor P. Such a scheme is in line with coarse-grained FPGA architectures (e.g. Virtex-II PRO from Xilinx), which consist of one or more programmable processors embedded within the FPGA's logic fabric. It is also in line with customizable processors where the system designer may choose to implement frequently occurring computation patterns in hardware. The portions (or fractions) of T_1 and T_2 to be implemented in hardware constitute the different implementation options. The two objectives to be optimized are the total hardware cost and the minimum clock frequency of P (which, for example, might influence its power consumption). Clearly, there can be different implementation options which satisfy T_1 and T_2's deadline constraints. If larger fractions of T_1 and T_2 are implemented in hardware, then the hardware cost increases and the required clock frequency of P decreases, and vice versa.

For any schedulable implementation, if (c, f) denotes the corresponding hardware *cost* and clock *frequency*, then a designer will be interested in identifying all possible tuples $(c_1, f_1), \ldots, (c_n, f_n)$ which capture the different performance tradeoffs. In the multicriteria optimization parlance, the set $\{(c_1, f_1), \ldots, (c_n, f_n)\}$ is referred to as the *Pareto curve* and each point (c_i, f_i) in this set is called a *Pareto-optimal solution* [6] (see Figure 1). Each (c_i, f_i) in this set has the property that there does not exist any schedulable implementation of T_1 and T_2 with a performance vector (c, f) such that $c \leq c_i$ and $f \leq f_i$, with at least one of the inequalities being strict. Further, let $\mathcal{S}$ be the set of performance vectors corresponding to all schedulable implementations. Let $\mathcal{P}$ be the set of performance vectors $\{(c_1, f_1), \ldots, (c_n, f_n)\}$ corresponding to all the Pareto-optimal solutions. Then for any $(c, f) \in \mathcal{S} - \mathcal{P}$ there exists a $(c_i, f_i) \in \mathcal{P}$ such that $c_i \leq c$ and $f_i \leq f$, with at least one of these inequalities being strict (i.e. the set $\mathcal{P}$ contains *all* performance tradeoffs). The vectors $(c, f) \in \mathcal{S} - \mathcal{P}$ are referred to as *dominated solutions*, since they are "dominated" by one or more Pareto-optimal solutions as shown in Figure 1.

In this paper we present a GPU based engine – called *GPU-Pareto* – for high speed computation of the Pareto curve $\mathcal{P} = \{(c_1, f_1), \ldots, (c_n, f_n)\}$. Our algorithm consists of the following two parts:

- The first part involves running a pseudo-polynomial time dynamic programming algorithm to find all the design points. This algorithm is the computationally expensive core of the design space exploration and in this paper we re-formulate this algorithm as a streaming algorithm to accelerate it on GPUs.
- The second part involves retaining the non-dominated solutions from the set of solutions found in the previous step. Since this part is not amenable to GPU-based acceleration, we run it on the CPU for optimized performance.

In essence, GPUPareto involves both GPU and the CPU to achieve optimal performance improvement.

Organization of the paper: The rest of this paper is organized as follows. In the next section we introduce our task model and some necessary notations. Section III introduces the formal problem statement and a pseudo-polynomial time algorithm to solve it. This is followed by a discussion on GPU architectures in Section IV. In Section V we present our GPU-based design space exploration technique, followed by the experimental results in Section VI.

II. Task Model

In this paper, we use the sporadic task model [3] in a preemptive uniprocessor environment to illustrate our GPU-based design space exploration scheme. Thus, we are interested in the schedulability analysis of a task set $\tau = \{T_1, T_2, \ldots, T_m\}$ consisting of m hard real-time tasks. Any task T_i can get triggered independently of other tasks in τ. Each task T_i generates a sequence of jobs, where each job is characterized by the following parameters:

- *Release Time*: the release time of two successive jobs of the task T_i is separated by a minimum time interval of P_i time units.
- *Deadline*: each job generated by T_i must complete by D_i time units since its release time.
- *Workload*: the worst case execution requirement of any job generated by T_i is denoted by E_i.

Throughout this paper, we assume the underlying scheduling policy to be the earliest deadline first (EDF). Our algorithm can be suitably

Tasks in the Task Set	Workload	Cost
# choices for task $T_1 = 3$ $E_1 = 12,\ P_1 = 40$	10	15
	8	45
	4	90
# choices for task $T_2 = 2$ $E_2 = 6,\ P_2 = 16$	5	24
	2	42
# choices for task $T_3 = 3$ $E_3 = 11,\ P_3 = 25$	8	11
	6	26
	5	82

TABLE I
TASK SET PARAMETERS.

modified to handle other scheduling policies as well. Assuming that for all tasks T_i, $D_i \geq P_i$, the schedulability of the task set τ can be given by the following well-known condition.

Theorem 1: A set of sporadic tasks τ is schedulable under EDF if and only if

$$(U = \sum_{i=1}^{m} \frac{E_i}{P_i}) \leq 1$$

where U is the processor utilization due to τ [3], [16].

III. THE PROBLEM STATEMENT

In this section, we formally state the multi-objective problem along with the help of an illustrative example. We then discuss the intractability of this problem even for the simple sporadic task model described in Section II. Finally, we derive a pseudo-polynomial time algorithm for solving it.

Recall that we are given a processor P, and a specified number of subtasks of each task T_i which can be implemented in hardware. For simplicity of exposition, we will henceforth assume that the processor P's clock frequency is constant and all the execution times of the tasks are specified with respect to this clock frequency. Our objective will be to minimize P's utilization (by mapping certain subtasks onto hardware) and at the same time also minimize the total hardware cost. In other words, our goal is to compute the *cost-utilization* Pareto curve $\{(c_1, u_1), \ldots, (c_n, u_n)\}$ for a pre-specified clock frequency of P. It is straightforward to see that such a Pareto curve can be easily transformed into a *cost-frequency* Pareto curve with P's utilization being ≤ 1 for the different frequency values.

For each task T_i, let there be n_i hardware implementation choices. Each of these n_i choices is associated with a certain hardware cost. Choosing the jth implementation choice for the task T_i lowers its execution requirement on P from E_i to $e_{i,j}$. Equivalently, the amount by which the execution requirement of T_i gets lowered on P is $\delta_{i,j} = E_i - e_{i,j}$. Hence, for each task T_i we have a set of choices $S_i = \{(\delta_{i,1}, c_{i,1}), \ldots, (\delta_{i,n_1}, c_{i,n_1})\}$, where $c_{i,j}$ is the hardware cost associated with the jth implementation choice. In this setup, the objective is to minimize the utilization

$$U(S) = \sum_{i=1}^{m} \frac{E_i - x_{i,j}\delta_{i,j}}{P_i}$$

and the cost $C(S) = \sum_{i=1}^{m} c_{i,j}x_{i,j}$, where S is the chosen implementation among the various available options.

We now illustrate this problem with the help of an example. A task set τ has three tasks $\{T_1, T_2, T_3\}$ with $\{E_1 = 12, P_1 = 40\}$, $\{E_2 = 6, P_2 = 16\}$, and $\{E_3 = 11, P_3 = 25\}$. The different possible hardware implementation choices for each task in this set is shown in Table I. Each row of this table shows the new execution requirement of a task on P after a part of this task is implemented in hardware, and the associated hardware cost. Note that as the execution requirement or workload of a task decreases, its associated hardware cost increases.

Algorithm 1 Minimum-cost schedulability analysis

Input: The task set τ, and a set S_i for each task T_i.
1: $U_{0,0} \leftarrow \sum_{i=1}^{m} E_i/P_i$
2: **for** $j \leftarrow 1$ to mC **do**
3: $\quad U_{0,j} \leftarrow \infty$
4: **end for**
5: **for** $i \leftarrow 1$ to m **do**
6: $\quad$ **for** $j \leftarrow 0$ to mC **do**
7: $\quad\quad$ For each pair $(\delta_{i,k}, c_{i,k})$ that belongs to the set S_i
8: $\quad\quad U_{i,j} \leftarrow \min\{U_{i-1,j}, U_{i-1,j-c_{i,k}} - \delta_{i,k}/P_i\}$
9: $\quad$ **end for**
10: **end for**

Following the notation we introduced above, for T_1 we have $e_{1,1} = 10$, $e_{1,2} = 8$ and $e_{1,3} = 4$. The corresponding hardware costs are $c_{1,1} = 15$, $c_{1,2} = 45$ and $c_{1,3} = 90$. Hence, the implementation choices for T_1 are given by the set $S_1 = \{(2, 15), (4, 45), (8, 90)\}$. The choices for T_2 and T_3 can be similarly computed from this table. Note that while T_1 and T_3 have three choices each, T_2 has only two choices. Thus, $n_1 = n_3 = 3$ and $n_2 = 2$. The goal is to identify the *cost-utilization* Pareto curve, which in this case includes pairs like $\{(11, .995), (214, .425)\}$ etc.

A. A Pseudo-polynomial Time Algorithm

Unfortunately, computing the exact *cost-utilization* Pareto curve is computationally intractable. This can be easily verified from the following two facts. First, the Pareto curve would typically contain an exponential number of points (which obviously cannot be computed in polynomial time). Second, computing any one point on the Pareto curve is NP-hard. This can be easily shown by a polynomial-time transformation of the knapsack problem [8].

Now, we present our algorithm to compute the Pareto curve. It consists of two parts. First, a dynamic programming algorithm (Algorithm 1) computes the minimum utilization that might be achieved for each possible cost. This algorithm runs in pseudo-polynomial time, and hence, turns out to be the compute-expensive kernel of our scheme. In Section V, we reformulate this algorithm to derive an accelerated GPU-based scheme. The second part involves finding out all undominated solutions (*cost-utilization* Pareto curve) from the entire solution set found by the dynamic programming algorithm. This is a straightforward implementation, and is not discussed in this paper because of space constraints.

Overview of Algorithm 1: Let $U_{i,j}$ be the minimum utilization that might be achieved by considering only a subset of tasks from $\{1, 2, \ldots, i\}$ when the cost is exactly j. If no such subset exists we set $U_{i,j} = \infty$. Let the maximum cost be C i.e. $C = max_{(i=1,2,\ldots,n;j=1,2,\ldots,n_i)}c_{i,j}$. Clearly, mC is an upper bound on the total cost that might be incurred. All other notations used are as introduced in Section II and Section III. Lines 1 to 4 of Algorithm 1 initialize $U_{0,0}$ to $\sum_{i=1}^{m} E_i/P_i$, and $U_{0,j}$ to ∞ for $j = \{1, 2, \ldots, mC\}$. The values $U_{i,j}$ for $i = 1$ to $i = m$ are computed using the iterative procedure in lines 5 to 10. Thus, any non-infinity value $U_{n,j}$ for $j = \{1, 2, \ldots, mC\}$ implies a feasible design choice of the task set with utilization $U_{n,j}$ and cost j. It can be easily verified that the running time of Algorithm 1 is $O(nmC)$, where $n = \sum_{i=1}^{m} n_i$, and its space complexity is $O(m^2C)$.

IV. GPU ARCHITECTURES

Before introducing our GPU-based engine, we give a brief overview of the GPU architecture in this section: we highlight

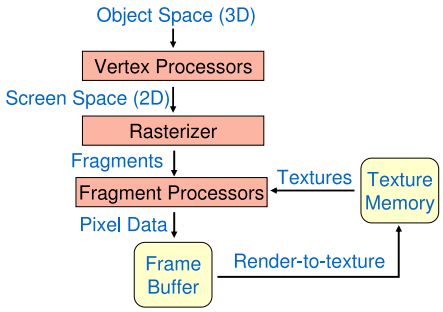


Fig. 2. The GPU graphics pipeline.

the GPU pipeline, the features that make GPUs attractive *stream processors* and the challenges in programming the GPUs.

The GPU Pipeline: All of today's commodity GPUs structure their graphics computation in a fixed order of processing stages called the graphics pipeline. Figure 2 shows the pipeline stages in a modern GPU. The input to the pipeline is a list of geometry, expressed as vertices in object (3D) co-ordinates and the output is an image in a framebuffer. A framebuffer is the portion of the graphics card memory that holds the information necessary to display a screen image. The first stage of the pipeline (on vertex processors) performs geometric transformations on each vertex and transforms each vertex from object space (3D) into screen space (2D) and assembles the vertices into triangles. Thus, the output of the first stage or geometry stage is triangles in screen space. The next stage, or rasterization, determines the screen positions covered by each triangle. The result of the rasterization stage is a data stream of elements or fragments for each pixel location covered by a triangle. Each incoming data element has a set of texture co-ordinates that reference a texture memory (see Figure 2). The third stage or the fragment stage consists of multiple fragment processors. They generate the addresses into the texture memory referred by the fragments and fetch their associated texture values. This data is used by a user-defined program executing on the processors to compute the fragment color (i.e. the color for each pixel). The output is finally written to the frame-buffer memory.

In this work, we will concentrate only on the fragment processors. In fact, a vast majority of general-purpose GPU applications use only fragment programs for their computation. This is because (i) they are the last in the graphics pipeline and their output may be read out directly, (ii) they are highly parallel (they are more in number than vertex processors), and (iii) they have a better memory-read performance compared to the vertex processors.

GPUs as Streaming Processors: In order to meet the ever increasing performance requirements set by the gaming industry, modern GPUs use two types of parallelism. First, multiple processors work on the vertex and fragment processing stage, i.e. they operate on different vertices and fragments in parallel. For example, a typical graphics card such as the nVidia GeForce 7900 GT has 8 vertex processors and 24 fragment processors. Second, each fragment processor can perform four concurrent vector operations such as instructions on the texture coordinates or on the color components of the incoming data stream.

Such explicit parallelism make GPUs an excellent platform for *stream processing* applications. Streaming processors read an input *stream* (which is a collection of records requiring similar computation), and apply the *kernel* (or operations to be performed on each element) to the stream before writing the results into an output stream. Since there are no dependencies between the various elements of the stream, they provide immense data parallelism for the multiple processors running the kernels. Another feature of stream processing applications is that several kernels often operate successively on the streams, and the output stream of the leading kernel is the input stream for the following kernel (see Figure 3).

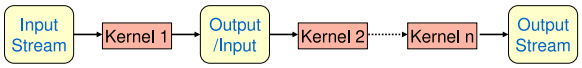


Fig. 3. Streaming model that applies kernels to an input stream and writes to an output stream.

To realize a streaming application on GPUs, kernels are compiled to fragment processors, textures are considered as input streams and render buffers as output streams. Realizing several successive kernels of a streaming application on a GPU requires several passes on the fragment processing pipeline. Towards this, the output of the fragment processor is stored as a texture and is then used during subsequent passes by successive kernel programs running on the fragment processor. This feedback loop is realized by using the output buffer of a completed pass as input texture for the following one (known as *render-to-texture* (RTT); see Figure 2).

Challenges in programming GPUs: Programming a GPU is not as straightforward as implementing an application on the CPU. This is because a GPU follows a highly parallel stream processing computational paradigm as described above. Since the kernels on all the fragment processors run in parallel, they can not have arbitrary data dependency on each other. Hence, the challenge is to correctly identify the data parallel segments so that dependency constraints are not violated. Hence, given any application, it must be first appropriately recast as an streaming application for an efficient implementation on the GPU.

V. The Design of GPUPareto

Before discussing the details of our GPUPareto engine, we outline the overall scheme to reformulate any algorithm as a stream processing application to run it on a GPU (see Figure 4). The first and most important step is to identify the data parallel kernels. Next, we need to compile these kernels to the fragment processors and properly set up the GPU data structures. Finally, depending on the application, we determine the number of iterations required on the fragment processors, and on completion, download the output to the CPU.

Following the above discussion, we first need to appropriately identify the data parallel computation in our dynamic programming (DP) algorithm (Algorithm 1) for it to be mapped onto the GPU. This crucial is because in Algorithm 1 the computation of the recurrence relation (line 7 to 8) involves non-trivial data dependencies. To resolve this, we constructed a data dependency graph; Figure 5 shows the dependency for the $U_{i,j}$th cell. Recall that $U_{i,j}$ be the minimum utilization that might be achieved by considering only a subset of tasks from $\{1, 2, \ldots, i\}$ when the cost is exactly j. The computation of $U_{i,j}$ depends on the $U_{i-1,j}$th cell and on the values of $U_{i-1,j-c_{i,k}}$ where $k = 1, 2, \ldots$, i.e the $U_{i-1,j-c_{i,1}}$th cell, the $U_{i-1,j-c_{i,2}}$th cell, and so on. Recall that $c_{i,j}$ is the hardware cost associated with the jth implementation choice of task T_i. Hence, Figure 5 shows that the computation of $U_{i,j}$ depends *only* on the previously computed cells i.e. cells in the $i-1$th row and *not* on cells in the ith row.

The observation here is that computation of U_{i,j_1} in the ith iteration is independent of U_{i,j_2}, where j_1 and j_2 may be any values between 1 and mC. In other words, given any iteration i computing any two

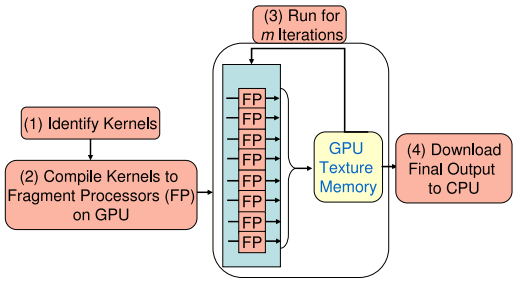


Fig. 4. Overall scheme to design and implement a GPU-based algorithm.

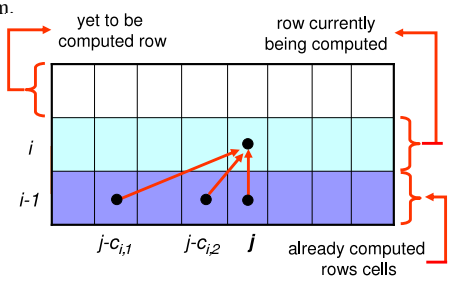


Fig. 5. Data dependency graph for Algorithm 1.

values of $U_{i,j}$ for different j are independent of each other. Hence, in Algorithm 1, the computation of the cells in the inner loop of the dynamic programming algorithm (i.e. lines 6 to 9) can be done independently of each other. Therefore, the basic idea is to compute the DP-based matrix in a row-by-row fashion.

Now we are ready to describe our formulation of this DP algorithm as a streaming application — the cells in a previously computed row are the *streams*, and the arithmetic operations specified by the recurrence relations in lines 7 and 8 of Algorithm 1 are implemented as *kernels*. Each row (streams) of the DP-based matrix is stored as a texture in the texture memory of the GPU and the recurrence relations (kernels) are compiled to the fragment processors (as explained in Section IV). A complete row of the matrix is computed in parallel by the fragment processors in the GPU. Note that since we have correctly mapped the data parallel sections to the fragment processors, there are no incorrect data fetches and we can achieve correct results. The newly-computed row is then stored in the texture memory. Finally, the subsequent kernel (i.e the next iteration of the DP) reads this computed row form the texture memory and this process is repeated for m passes, where m is the number of tasks in a task set τ as explained in Section II. Of course, at the start of our streaming application, we have to set the initial values of the cells in the first row in the texture memory according to the initialization in lines 1 to 4 of Algorithm 1. Algorithm 2 shows the pseudo-code of the recursive algorithm for a kernel. U_{prev} is the old value of a cell in the texture memory and U_{new} is the new value computed by the kernel. $f()$ is a function which returns the column value of the cell being computed by this kernel i.e. the corresponding value j (see Line 3). Thus, $U_{tmp} - \delta_{i,k}/P_i$ (line 4) corresponds to the recursive equation in line 8 of Algorithm 1.

A. Data Structures

In this section we discuss the data structures created on the GPU memory for handling the *streams* in our GPU-based computation. We need to store two rows (which stores $U_{i,j}$ values) each of size $m \times C$, where one previously computed row is read while the other row is computed by the DP algorithm. Following the memory organization supported by GPU architectures, we used two texture buffers to store

Algorithm 2 The Streaming Formulation of the DP

Input: The task set τ, and a set S_i for each task T_i.

1: **for** $i \leftarrow 1$ to m **do**
2: For each pair $(\delta_{i,k}, c_{i,k})$ that belongs to the set S_i
3: $tmp = f() - c_{i,k}$
4: $U_{new} \leftarrow \min\{U_{prev}, U_{tmp} - \delta_{i,k}/P_i\}$
5: **end for**

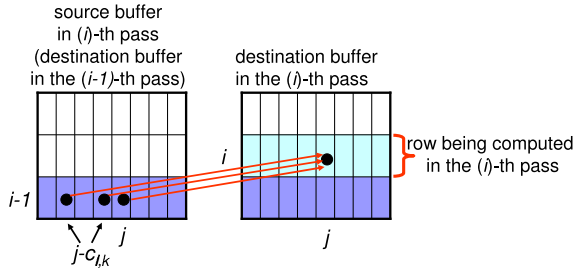


Fig. 6. Data buffers in the GPU memory during the i-th pass through the rendering pipeline.

the cells of the row – one of which serves as the source buffer (containing the previously computed row of the matrix) and the other serves as the destination buffer (containing the row being computed in a certain pass). During each pass through the GPU pipeline, the destination buffer of the previous pass serves as the current source buffer and their roles are interchanged from one pass to the next. Corresponding to the dependency relation shown in Figure 5, in Figure 6 we illustrate the use of the source and destination buffers during the i-th pass.

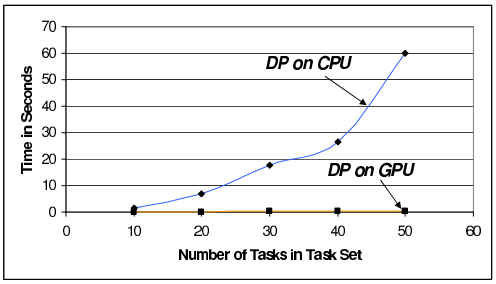
The above matrix computation procedure was implemented using OpenGL's [19] *Render-to-Texture* support. The two texture objects are attached to the frame buffer object bound for rendering, with one texture for writing and the other one for reading. These are swapped during each new pass as explained above. In each pass, the previous render target buffer binds as texture for reading and the previous buffer for reading becomes the render target.

In this section we discussed the GPU-based dynamic programming algorithm, which is the first part our design space exploration scheme. The second part involves retaining the undominated solutions (*cost-utilization* Pareto curve) from the entire solution set found in the previous step. Since, this is not compute-intensive step and is not amenable to GPU-based acceleration, we have implemented it on the CPU. This algorithm is straightforward and is not elaborated here due to space constraints.

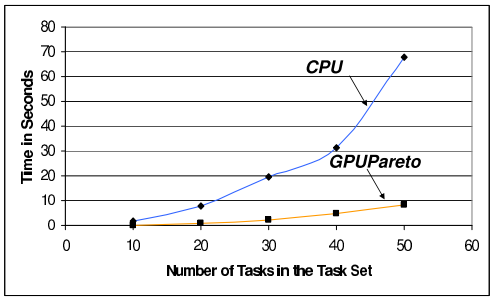
VI. Experimental Results

In this section we report some of the experimental results that were obtained by running our GPUPareto engine on a set of synthetic task sets. We have compared these results with those obtained by running a pure CPU-based implementation.

For our experiments we randomly generated tasks with execution requirements between 200 and 600 time units; the periods were between 600 and 20,000 time units. The number of hardware implementation choices associated with any task was varied between 1 and 10, i.e. $1 \leq n_i \leq 10$. For each choice, the maximum value associated with any $\delta_{i,j}$ was set to E_i. The parameter C, which is the maximum cost associated with any implementation choice was set to 16384 for our experiments. This number was chosen because graphics processors lack integer arithmetic. Using floating point values might lead to wrong address calculations (e.g. see line 3 in Algorithm 2) due to improper rounding-off. Thus, if C is not a power of 2 for a given task set, one needs to choose the next higher power of 2 as an upper bound. To show that this is not a restriction of our scheme,



(a) Running time of DP.



(b) Total running time.

Fig. 7. Running times for a purely CPU-based implementation versus a GPU-based implementation (GPUPareto).

we choose 16384 (2^{14}) and show that even for such large values our DP algorithm runs within a fraction of a second. Furthermore, with $C = 16384$, there are upto 16384 design points, and typically around 6100 points on the pareto curve for task set with around 50 tasks. Such large design space instances are clearly very suitable to test the applicability of the GPUPareto scheme.

All the CPU times reported below were measured on a machine with Windows XP, running on a 3.0 GHz CPU with 1 GB RAM. Our machine had a PCI express board equipped with an nVIDIA GeForce 8800 GTX GPU with 768 MB RAM, where we conducted our GPU experiments. All the implementations were done in C++. For the GPU implementation, we used OpenGL with Cg as the shader language (for programming the fragment processors).

Figure 7(a) shows the time taken to run the DP algorithm on the CPU versus the time taken on the GPU, when the number of tasks in the task set is progressively increased from 10 to 50. Our implementation on the GPU clearly achieves very attractive (upto $100\times$) speedups. Figure 7(b) shows the overall running time involved in computing the exact Pareto curve on the CPU versus the time taken by the GPUPareto engine. In order to accurately report the total processing time on the GPU, we take into account the sum of data structure uploading time to GPU memory, the computation time on the GPU and the downloading time from GPU memory. The total time taken by the GPUPareto engine adds this sum with the time taken to run the second part of our algorithm, which is run on the CPU. Recall, that to compute the exact Pareto curve, we need to run (i) the Algorithm 1 (GPU implementation) and (ii) then retain all the undominated solutions (CPU implementation).

Compared to a purely CPU-based implementation, the GPU-based analysis results in significant speedups, with the analysis times reducing from more than a minute to less than 9 seconds. Such speedups allow a designer to get almost instantaneous feedback during an interactive design session with an automated tool, thereby improving design productivity. Further, this comes at no additional cost, assuming that the desktop/notebook computer running the design tool already has a commodity GPU.

VII. Concluding Remarks

Using a specific case study, in this paper we showed that modern commodity graphics hardware may be exploited to accelerate computationally expensive kernels in system-level design tasks. In particular, we presented *GPUPareto*, an engine to solve a standard multiobjective design space exploration problem arising in the context of hardware/software partitioning. Our contribution might also be valuable in light of the fact that the core problem solved is a variant of a classic optimization problem, viz. the knapsack problem. This NP-hard problem is at the heart of numerous problems arising in the context of EDA and other areas of computer science and engineering. We believe that the generality of this problem might motivate other researchers within the EDA community to explore the possibility of exploiting GPUs for a variety of other system-level design problems as well.

References

[1] P. K. Agarwal, S. Krishnan, N. H. Mustafa, and S. Venkatasubramanian. Streaming geometric optimization using graphics hardware. In *European Symposium on Algorithms (ESA)*, 2003.

[2] A. Ailamaki, N. K. Govindaraju, S. Harizopoulos, and D. Manocha. Query co-processing on commodity processors. In *VLDB*, 2006.

[3] S. Baruah, A.K. Mok, and L.E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *RTSS*, 1990.

[4] nVIDIA CUDA Zone, http://www.nvidia.com/object/cuda_home.html.

[5] Programming massively parallel processors: the nVIDIA experience. Tutorial, DAC, 2008.

[6] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[7] R. Dutta, J. Roy, and R. Vemuri. Distributed design-space exploration for high-level synthesis systems. In *DAC*, 1992.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[9] Kanupriya Gulati and Sunil P. Khatri. Towards acceleration of fault simulation using graphics processing units. In *DAC*, 2008.

[10] A. Hamann and R. Ernst. Efficient priority optimization in complex distributed embedded systems through search space adaptation. In *GECCO*, 2007.

[11] R. Henftling, A. Zinn, M. Bauer, M. Zambaldi, and W. Ecker. Re-use-centric architecture for a fully accelerated testbench environment. In *In DAC*, 2003.

[12] H. P. Huynh and T. Mitra. Instruction-set customization for real-time embedded systems. In *DATE*, 2007.

[13] D. Kim, S. Ha, and R.Gupta. Parallel co-simulation using virtual synchronization with redundant host execution. In *DATE*, 2006.

[14] Y. Kim, W. Yang, Y.-S. Kwon, and C.-M. Kyung. Communication-efficient hardware acceleration for fast functional simulation. In *DAC*, 2004.

[15] J. Krüger and R. Westermann. Linear algebra operators for GPU implementation of numerical algorithms. *ACM Transactions on Graphics*, 22(3):908–916, 2003.

[16] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[17] W. Liu, B. Schmidt, G. Voss, and W. Müller-Wittig. GPU-ClustalW: Using graphics hardware to accelerate multiple sequence alignment. In *HiPC*, 2006.

[18] K. Mueller and F. Xu. Ultra-fast 3d filtered backprojection on commodity graphics hardware. In *IEEE ISBI*, 2004.

[19] R. J. Rost. *OpenGL Shading Language*. Addison-Wesley, 2006.

[20] I. Skliarova and A. B. Ferrari. A software/reconfigurable hardware SAT solver. *IEEE Transactions on VLSI Systems*, 12(4):408–419, 2004.

[21] L. Soulé and T. Blank. Parallel logic simulation on general purpose machines. In *DAC*, 1988.

[22] S. Venkatasubramanian. The graphics card as a stream computer. In *SIGMOD-DIMACS Workshop on Management and Processing of Data Streams*, 2003.

[23] M. R. Zargham. Parallel channel routing. In *DAC*, 1988.

Session 7C

BIST, Error Modeling

Built in Self Test Based Design of Wave-Pipelined Circuits in ASICs

V. Vireen, N. Venugopalachary, G. Seetharaman, and B. Venkataramani
National Institute of Technology, Trichy, India.
bvenki@nitt.edu.

Abstract

Wave-pipelining enables digital systems to be operated at higher frequencies by properly selecting the clock periods and clock skews so as to latch the output of combinational logic circuits at stable periods. In the literature, only trial and error and manual procedures are adopted for these selections. The major contribution of this paper is the proposal for automating the above procedure for the ASIC implementation of wave-pipelined circuits using built in self test approach. For the purpose of verification, a Coordinate rotation digital computer and filters using the distributed arithmetic algorithm are implemented. To test the efficacy, these circuits are implemented by adopting three schemes: wave-pipelining, pipelining and non-pipelining.

From the implementation results, it is observed that the wave-pipelined circuits are 21-29 % faster compared to non-pipelined circuits. The pipelined circuits are 22-48 % faster compared to wave-pipelined circuits but at the cost of about 18-28 % increase in area.

Index terms- ASIC, Wave-pipelining, BIST, FSM, DAA, PRSG

1. Introduction

The design of high speed arithmetic circuits is a key element to build high-performance computing systems. Techniques such as pipelining, wave-pipelining and asynchronous pipelining have been proposed for increasing the speed of arithmetic logic units.

In conventional pipelining, a combinational logic circuit is partitioned into a number of parts or stages and registers are introduced between adjacent stages. All the registers are fed with a common reference clock. Even though, this technique improves the throughput of a logic circuit, it has a number of disadvantages such as increase in latency, increase in area and clock distribution complexity.

Wave-pipelining is one of the alternatives to pipelining. It provides a method for significantly reducing clock loads and the associated area and latency while retaining the external functionality and timing of a synchronous circuit.

The idea of wave-pipelining was originally introduced by Cotton [1], who named it as *maximum rate pipelining*. Cotton observed that the rate at which logic can propagate through the circuit depends not on the longest path delay but on the difference between the longest and the shortest path delays. Wave-pipelining has been employed for implementing a number of systems on both ASICs and FPGAs [2], [3]. The concept of wave-pipelining has been described in a number of previous works [4], [5]. In [6], an automation technique for tuning the clock period is proposed. This technique uses the system clock: First, it applies the system clock to the circuit; if the circuit is not working with system clock then it doubles the period and applies it to the circuit. This doubling process is continued until system works properly. However, this approach cannot ensure that the circuit works at the highest possible frequency. This may be either because the circuit operating frequency is greater than system clock frequency or because the highest operating frequency is not exactly equal to clock frequency / 2^n where n is an integer.

An automation technique which uses on chip clock and skew generators and Built In Self Test (BIST) approach is proposed in [7] for adjusting the clock period and clock skew of an FPGA based wave-pipelined circuit. In this paper, the extension of this technique for wave-pipelined circuits implemented on ASICs is considered. A Coordinate rotation digital computer (CORDIC) unit and filters using distributed arithmetic algorithm are used for the study and implementation.

The organization of the rest of the paper is as follows: In section 2, an overview of wave-pipelining and the challenges involved in the design of wave-pipelined circuits are described. In section 3, self tuned wave-pipelined circuit is presented. An overview of the CORDIC unit and DAA based FIR filters are given section 4 and 5 respectively. In section 6, the implementation results are presented. Section 7, summarizes the conclusions.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.46

2. Overview of wave-pipelining

Figure 1 shows a typical combinational logic circuit along with the input and output registers [8]. Figure 2 depicts the flow of data through the above circuit. The skew between the clocks at the input and output registers is denoted as Δ. At the beginning of each clock cycle, data is fed into the combinational logic block through the input register. A number of paths may exist between the inputs and output of a logic block. A change in the input causes the output to change after a delay of D_{min}, D_{max} through the shortest and longest path respectively. The shaded regions bounded by (D_{min} and D_{max}) depict the periods where the logic levels of the logic block vary with time. The non shaded areas depict the stable duration of the logic block.

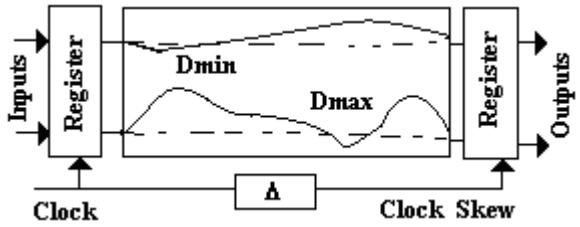


Figure 1. Combinational logic circuit with input and output registers.

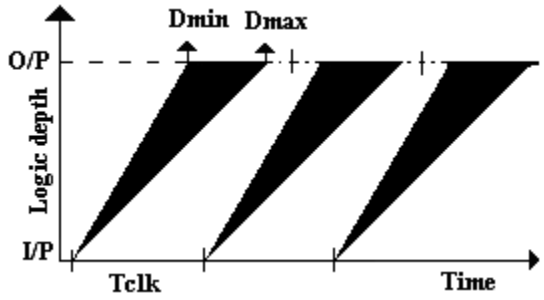


Figure 2. Temporal /spatial diagram of data flow through the combinational logic circuit.

In the conventional system, the output register is clocked in the non shaded region and the minimum clock period, T_{clk} is chosen to be greater than D_{max}. In the Wave-Pipelined (WP) system, the clock period is chosen to be (D_{max} - D_{min}) + clocking overheads such as set up time, hold time etc. Partitioning overhead is avoided since the wave-pipelined circuit is not partitioned into stages separated by synchronizers (registers). Hence, it may result in lower power dissipation and clock routing complexity compared to that of pipelined circuit. However, the maximization of the operating speed of the wave-pipelined circuit requires the adjustment of the clock period, clock skew (δ) and equalizing the path delays.

3. Self tuned wave-pipelined circuit

The self tuned wave-pipelined circuit is proposed by including a BIST circuit to tune the clock frequencies and clocks with different skews. The block diagram of self tuned wave-pipelined circuit is shown in Figure 3. It consists of different functional blocks namely pseudo random binary sequence (PRBS) generator, signature analyzer, Counter, Programmable Clock generator Circuit, Programmable skew generator circuit and FSM.

Operation: A self tuned wave-pipelined has two modes of operation namely test mode and normal mode. TM signal is used to select the mode of operation. In test mode, FSM first gives Clock Select line (CS) = 0 and Skew Select line (SS) = 0 for the clock generator and skew generator circuit respectively. The programmable clock generator circuit generates the first clock and it is applied to PRBS circuit, programmable skew generator circuit and input register.

The PRBS block is used for exhaustive testing and it generates all 2^n combinations of the inputs for an n-bit input. The programmable skew generator circuit generates skew and the skewed clock is applied to the output register and counter circuit. The counter is used to keep track of the number of test vectors fed to the combinational block and it generates the enable signal (sig_en) after all the test vectors have been applied.

Instead of comparing every output with the expected output, a signature is generated from the outputs corresponding to all the applied inputs using PRBS generator and it is compared with stored value in signature analyzer circuit. The signature analyzer gives two control signals (sig_in and chng) to the FSM block which indicates the match or not. If chng is 1, no match occurs and select lines are changed, otherwise, the select lines are fixed. Depending upon the control signals received from signature analyzer, CS and SS values to the Clock and Skew generator circuits are generated. If there is no match, FSM changes the SS value from 0-7 for every CS value. Even after all the skews are applied for a particular CS value, if there is no match, it changes the CS value. In this way, FSM changes CS and SS values until it finds a match. When match is found, FSM fixes CS and SS values and the circuit is placed in normal mode by changing TM=0. In normal mode, user inputs are applied.

For large word size, exhaustive testing will take unduly long time. In this case, the testing time can be minimized by finding the optimal test vector set. This is one of the challenges in the testing of wave-pipelined

circuits. Alternatively, a set of random vectors may be used for testing the wave-pipelined circuits. By repeating the test with different set of random vectors, the confidence level of the testing scheme may be improved.

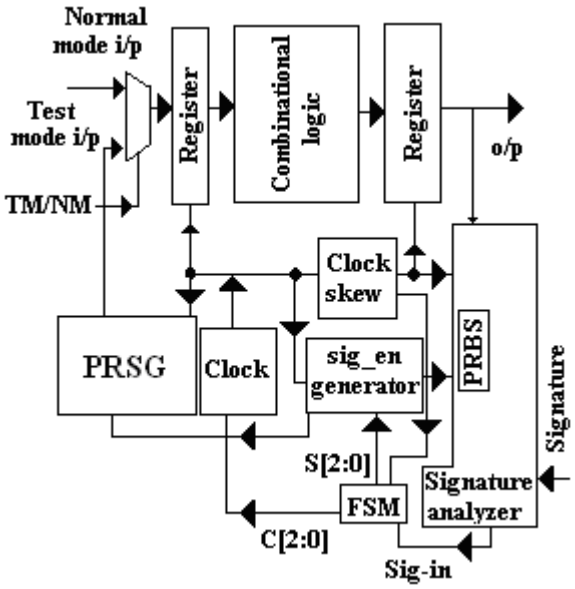


Figure 3. Block diagram of the self tuned wave-pipelined circuit

3.1 Programmable clock generator

The block diagram of Programmable clock generator is shown in Figure 4. It consists of buffers, multiplexers and an inverter. The frequency of the clock to be generated is controlled by select lines C0-C2. The FSM block systematically varies the C0-C2 values and gives as input to the Clock generator circuit. In Figure 4, d1, d2, d3……d8 are buffers. The signal cflag is set to 0 initially in order to initialize the clock state to be 1. After about 10 ns, it is set to be 1 for entire simulation. When cflag is 1 feedback path is selected.

3.2 Programmable skew generator

The Programmable skew generator is obtained by modifying the circuit in Figure 4 by removing the feed back from the output to the input. The amount of the skew to be introduced for the clock is controlled by select lines S0-S2 (the labels C0-C2 in Figure 4 are replaced by S0-S2 for skew generators). The FSM block systematically varies the S0-S2 values and gives as input to the skew generator circuit.

4. An overview of CORDIC algorithm

CORDIC [9] is an iterative arithmetic algorithm introduced by Volder and later refined by Walther and others. CORDIC unit uses only shifts and adds to perform a wide range of functions including vector rotations, certain trigonometric, hyperbolic, linear and logarithmic functions.

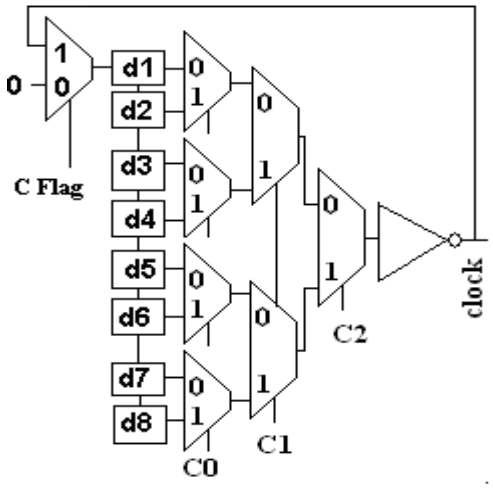


Figure 4. Programmable clock generator

CORDIC algorithm is used in diverse applications such as mathematical coprocessor units, calculators, waveform generators, universal modulator, demodulator digital filters carrier as well as bit time recovery circuits and digital modems. In the rotation mode, CORDIC may be used for converting a vector in polar form to rectangular form. In the vector mode, it converts a vector in rectangular form to polar form [10].

The iterative equations for the rotating a vector (x_{in}, y_{in}) in a Cartesian plane by an angle θ to another vector with the coordinates (x_{fin}, y_{fin}) using N iterations is given by

$$\delta_i = \mathrm{sgn}(z_i) \quad (1)$$

$$x_{i+1} = x_i - \delta_i y_i 2^{-i} \quad (2)$$

$$y_{i+1} = y_i + \delta_i x_i 2^{-i} \quad (3)$$

$$z_{i+1} = z_i - \delta_i \tan^{-1}(2^{-i}) \quad (4)$$

$$z_0 = \theta \quad (5)$$

$$(x_{fin}, y_{fin}) = \left(\frac{x_N}{\Pi_i^N \cos\theta_i}, \frac{y_N}{\Pi_i^N \cos\theta_i}\right) \quad (6)$$

The value of θ_i for i = 1, 2.., N is chosen such that $\tan\theta_i$ is 2^{-i}. The shift and add operation required for

each of the iteration is carried out using a single shift and add block in the serial CORDIC scheme (also referred to as the folded CORDIC scheme). Separate hardware blocks are used for each iteration in the case of Parallel or Unrolled CORDIC scheme [11]. In this paper, Unrolled CORDIC scheme is used for the implementation.

5. An overview of DAA based FIR filter

Distributed Arithmetic plays an important role in Digital Signal processing. DA can be optimized for area efficiency, speed efficiency or for both. For efficient implementation of DA, a number of algorithms such as Read Only Memory (ROM) decomposition technique and offset binary coding have been proposed in the literature [12]. In the ASIC implementation no ROMs are used for storing the coefficient but hardwired logic is used to implement the coefficient ROM tables, which greatly decreases the area and limits the flexibility. Normally, for the computation of vector dot product using DA, the content of DA ROM is stored assuming multiplication using 2's complement arithmetic with sign extension technique.

The computation of the output of an N-tap Linear Time Invariant (LTI) filter and computation of transform of an Nx1 vector can be generalized as the problem of computation of the sum of products given by [13].

$$y(n)=\sum_{k=0}^{N-1}a(n,k)x(k) \qquad (7)$$

In the case of LTI filters and transform computation, a(n,k) is time invariant and only x(k) varies with time. In view of this, y(n) can be computed by using the look up tables for multiplication. This can be achieved as follows: The input samples x(k) may be assumed to be represented in 2's complement representation using W bits and can be written as

$$x(k)=-x(W-1,k)+\sum_{m=1}^{W-1}x(W-1-m,k)2^{-(m)} \qquad (8)$$

Substituting equation (8) in (7) and interchanging the order of summation with respect to m and k, we get

$$y(n)=-S(W-1)+\sum_{m=0}^{W-2}S(m)2^{-(W-1-m)} \qquad (9)$$

Where $$S(m)=\sum_{k=0}^{N-1}x(m,k)a(n,k) \qquad (10)$$

It may be noted that x(m,k), for m= 0,1, … W-1, takes binary values 1 or 0. Hence, S(m) can be computed using ROM with address as the bits x(m,0), x(m,1), … x(m,N-1). Furthermore, the contents of S(m) is the same for all values of m.

5.1. Full parallel DA algorithm

To compute y(n), W ROMs, ROM 0 – ROM (W-1) can be used. ROM 0 – ROM (W-2) contain the same content and correspond to S(0) – S(W-2). ROM (W-1) corresponds to [-S(W-1)] and is actually the 2's complement of the content of the other ROMs. The MSBs of all the samples are fed as the address to the $(W-1)^{th}$ ROM. The next bits of all the samples are fed to the $(W-2)^{th}$ ROM address bits. Similarly, the LSBs of all the samples are fed as address to the 0^{th} ROM. For W=8, y(n) can be computed using four stages of adders. y(n) is expressed using S(0) – S(7) in equation (11).

$$y(n)=\{[-S(7)+S(6)2^{-1}]+[S(5)+S(4)2^{-1}]\}$$
$$+\{[S(3)+S(2)2^{-1}]+[S(1)+S(0)2^{-1}]2^{-2}\}2^{-4} \qquad (11)$$

Equation (11) requires multiplication of the numbers by 2^{-1}. If 2's complement multiplication with sign extension is used, this requires shifting the number towards right i times and replicating the Most Significant Bit (MSB) i times. For example, multiplication of a number 10100101 represented in 2's complement form by 2^{-4} results in the number 1111 1010 0101.

5.2. ROM Organization

If n taps are used then all 2^n values are stored in single ROM. However, to reduce the hardware complexity ROM decomposition technique which uses two or more ROM of smaller length is adopted in this paper. Figure 5 shows Distributed Arithmetic using ROM decomposition.

6. Implementation Results

The self tuned wave-pipelined circuit in Fig. 3 is studied by using either a 13-bit CORDIC or DAA based FIR filters with 8 and 16 taps as the combinational logic block. These circuits are implemented using 0.35um technology in ASIC. Verilog HDL language is used to describe the functionality of the circuit and after the circuit is described in HDL, functionality is verified using Synopsys VCS simulation tool. Synopsys Design Vision is used for synthesizing the circuit. In synthesis, first target libraries and next HDL files are read. After specifying the design constraints, circuit is compiled and synthesized net list is stored in HDL file. This HDL file is used for pre-layout simulation. Synopsys VCS simulation tool is used to do pre-layout simulation. After the pre-layout simulation results are satisfactory, layout is drawn.

Cadence Encounter is used for layout. After the design is placed and routed, routed net list is saved in HDL file. Parasitics are extracted into .spf and .spef files and Delay is calculated and saved to file with .sdf

extension. All these files are used to perform post-layout simulation. Synopsys VCS tool is used for post-layout simulation. A C-program is used to generate the Signature for the test inputs and it is stored in the signature analyzer for comparison.

One of the major advantages of this approach is that the time for simulation is of the order of few seconds. On the other hand, the SPICE level simulation takes a few hours even for a smaller word size. This is one of the major contributions of this paper.

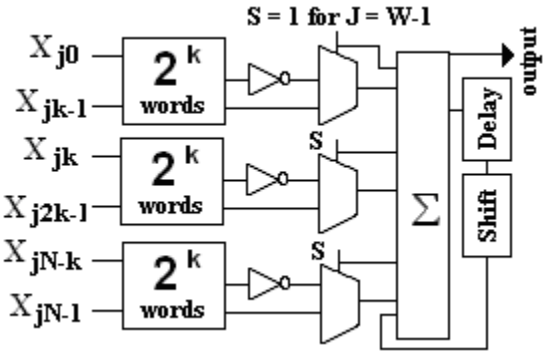


Figure 5. Distributed arithmetic using ROM decomposition.

Table 1. Implementation results for CORDIC unit.

Schemes	Area	Freq. (MHz)	Power (mW)	Power at normalized frequency mW/Hz
Wave-Pipelining	232779	84	30.98	30.98
Pipelining	257635	122	53	36
Non-Pipelining	259046	50.7	50.7	-

CORDIC unit and the FIR filters are implemented using three different schemes: Pipelining, Wave-pipelining and Non-pipelining. With the help of self adjustable clock and clock skew, wave-pipelined design is made to operate at higher frequency. Table 1 gives the results for 13 bit CORDIC unit using the three schemes.

From the results, it is observed that wave-pipelined CORDIC unit is 33% faster than non pipelined CORDIC whereas pipelined circuit is 45.3% faster than wave-pipelined circuits but this is achieved with less than 14 % increase in area that of wave-pipelined circuit.

Table 2 shows the synthesis result of the 8 tap-8 bit FIR filter using distributed ROM. From the results, it is observed that wave-pipelined circuits are 29% faster than non pipelined circuits whereas pipelined circuits are 41.6% faster than wave-pipelined circuits but this is achieved with less than 28 % increase in area that of wave-pipelined circuits.

Table 2 also shows the synthesis result of the 16 tap-8 bit FIR filter using distributed ROM. From the results it is observed that wave-pipelined circuits are 21% faster than non pipelined circuits whereas pipelined circuits are 44% faster than wave-pipelined circuits but this is achieved with less than 18 % increase in area than that of wave-pipelined circuits.

Table 2. Synthesis results of FIR filter (8 tap-8 bit, 16 tap-8 bit) using DAA.

Taps	Schemes	Area	Freq. MHz	power mW	Power at normalized frequency mW/Hz
8 tap 8 bit	Wave-Pipelining	254050	58.8	2.4	2.4
	Pipe lining	325078	83.3	3.5	2.47
	Non-Pipelining	238662	45.5	1.78	-
16 tap 8 bit	Wave-Pipelining	254050	58.8	2.4	2.4
	Pipelining	325078	83.3	3.5	2.47
	Non-Pipelining	238662	45.5	1.78	-

In order to assess the superiority of wave-pipelining with regard to power dissipation, both wave-pipelined and pipelined circuits are operated at the same frequency (corresponding to the maximum operating frequency of the wave-pipelined circuit) and the results are given in Table 1-2. The pipelined CORDIC unit and 8 tap FIR filter dissipate 16% and 3% more power respectively than wave-pipelined circuits. For the 16 tap filter, power dissipation of pipelined filters is lower by 55%. The power dissipation due to overheads decreases with the number of taps as the filter with higher number of taps operates at a lower frequency. At lower logic depths, the overheads make the wave-pipelining to be inefficient. At higher logic depths, overheads along with the increased capacitance make the wave-pipelined DA filter to be less efficient with regard to power dissipation.

Figure 6, Figure 7 and Figure 8 show the layout diagram of CORDIC unit, 8-bit FIR filter with 8 taps and 16 taps using BIST approach in 0.35um technology respectively. Cadence encounter is used for drawing the layout. Post-layout simulation results of the 8-bit FIR filter with 8 taps and 16 taps agree with the results obtained through C program.

7. Conclusion

The automation scheme proposed for the ASIC implementation of wave-pipelined CORDIC unit and distributed arithmetic based 8 and 16 tap FIR filters are implemented in 0.35u technology. Synopsys Design Vision is used for synthesis, Synopsys VCS is used for simulation and Cadence Encounter is used for layout. From the implementation results it is observed that wave-pipelined circuits are 21-29% faster than non-pipelined circuits. The pipelined circuits are 22-48% faster compared to wave-pipelined circuits but at the cost of about 18-28% increase in area. When both pipelined and wave-pipelined circuits are operated at the same frequency, the superiority of one over the other with regard to power dissipation depends on the logic depth of the circuit. From the implementation results, it is observed that, only for medium logic depths as well as word sizes, the ASIC based wave-pipelined circuit is found to be more efficient in both area and power dissipation than pipelined circuit.

The technique adopted in this paper for back annotation enables the post-layout simulation to be carried out in few seconds. The alterate approach using SPICE simulation typically requires few hours even for small word sizes. This is one of the major contributions of this paper.

The BIST approach proposed in this paper may also be used for tuning the wave-pipelined circuit to take care of process, voltage and temperature (PVT) variations.

8. References

[1] L. Cotton, "Maximum rate pipelined systems," in *Proc.AFIPS Spring Joint Comput. Conf.*, 1969.

[2] J. Nyathi and J. G. Delgado-Frias, "A hybrid wavepipelined network router," *IEEE Transactions on Circuits and Systems I*:Fundamental Theory and Applications, vol. 49, no. 12, pp. 1764–1772, Dec. 2002.

[3] O. Hauck., A. Katoch and S. A. Huss, "VLSI system design using asynchronous wave pipelines: a 0.35 μm MOS 1.5 GHz elliptic curve public key cryptosystem chip," *Proc. of Sixth Intl. Symposium on Advanced Research in Asynchronous Circuits and Systems, 2000,(ASYNC 2000)*, pp. 188 –197, April 2000.

[4] W. P. Burleson, M. Ciesielski, F. Klass, and Liu, "Wave-pipelining: a tutorial and research survey," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 6, no. 3, pp. 464 –474, Sep.1998.

[5] C. Thomas Gray, W. Liu and R. Cavin, "Wave Pipelining: Theory and implementation," *Kluwer Academic Publishers*, 1993.

[6] Woo Kim, Yong Kim, "Automating Wave- Pipelined Circuit Design", IEEE Design & Test of Computers, Vol. 20, Nov 2003.

[7] G.Seetharaman, B.Venkataramani and G..Lakshminarayanan, "Design and FPGA implementation of self-tuned wave-pipelined filters," *IETE journal of research,* vol 52, no. 4, pp. 305-313, July- August 2006.

[8] C. T. Gray, W. Liu, and R. Cavin III, "Timing constraints for wave pipelined systems," *IEEE Trans. Computer-Aided Design*, vol. 13, pp.987–1004, Aug. 1994.

[9] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. on Electronic Computers,* vol. EC-8, no. 3, pp. 330-4, Sept. 1959.

[10] W.Tuttlebee, "Software defined radio: Baseband technology for 3G," *Wiley*, 2004.

[11] R.Andraka "A Survey Of CORDIC Algorithm For FPGAs," *Proc. of ACM/SIGDA 6th international symposium of FPGAs (FPGA'98),* Monterey, CA, pp.191-200, 1998.

[12] K. K. Parhi, "VLSI Digital Processing Systems:Design and Implementation," *John Wiley & Sons*,1999.

[13] Mintzer, L., "FIR filters with the Xilinx-FPGA,"*FPGA'92 ACM/SIGDA Workshop on FPGAs*,pp. 129-134, 1992.

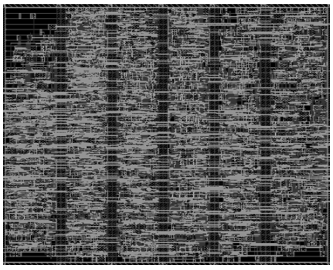


Figure 6. Layout of 13 bit CORDIC using BIST

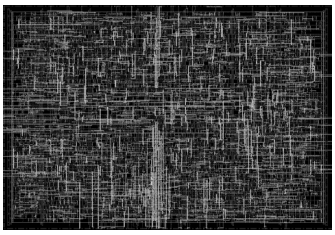


Figure 7. Layout of the 8 tap FIR filter using BIST

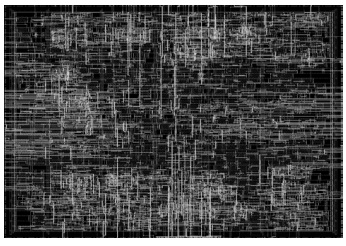


Figure 8. Layout of 16 tap FIR filter using BIST

2009 22nd International Conference on VLSI Design

WOR-BIST: A Complete Test Solution for Designs Meeting Power, Area and Performance Requirements

Chunhua Yao, Kewal K. Saluja and Abhishek A. Sinkar
Department of Electrical and Computer Engineering, University of Wisconsin-Madison
{yao, saluja, sinkar}@ece.wisc.edu

Abstract

A complete Built-In Self-Test (BIST) solution based on word oriented Random-Access Scan architecture (WOR-BIST), is proposed. Our WOR-BIST scheme reduces the test power consumption significantly due to reduced switching activity during scan operations. We also provide a greedy algorithm to reduce the test data volume and test application time. We performed logic simulation of the test vectors to show its impact on the average and peak power during testing. We implemented the scheme to demonstrate its impact on the chip area and timing performance. Application of our scheme to large ISCAS and ITC benchmark circuits shows that our scheme is superior in area, power and performance to the conventional multiple serial scan.

1. Introduction

Testing of integrated circuits is now a front-end issue in the semiconductor industry because of the rapid increase in the complexity of modern VLSI circuits. Due to the poor controllability and observability of flip-flops, testing large sequential circuits has always been challenging. Evolution of Design for Testability (DFT) technology has simplified this to certain extent. Recent DFT research has focused on reducing the test data volume, test application time and test power consumption. Serial Scan (SS) is the popular scan-based DFT technique because of its simplicity and relatively low hardware overhead. However, an important drawback of serial scan is that the switching activity and resulting power consumption during testing is much higher than normal operation of the circuit. The excessive heat dissipation caused by high power consumption can produce incorrect responses even when the circuit is fault-free [1] or permanently damage the circuit under test (CUT).

Unlike serial scan, Random Access Scan (RAS) [2, 3, 4, 5] allows individual access to all flip-flops in the circuit. RAS-based DFT method was shown to have the ability to reduce the test application time, test data volume and test power consumption [3]. The original RAS technique is impractical because of its large hardware overhead.

Recently, many new architectures have been proposed to improve the testing performance of RAS with relatively low overhead. A modified scheme of RAS is described in [4]. To minimize the routing complexity, RAS-based scan-cells are studied in [5, 6]. Progressive Random Access Scan (PRAS) is a RAS-based architecture [7] proposed to simultaneously reduce the test power, test data volume and test time with a comparable area overhead to that of serial scan.

While the cost of manufacturing a transistor has been reducing rapidly, the cost of testing eachtransistor has been almost constant for the past many years. Built-In Self-Test (BIST) is a promising technique because of its advantage of low cost compared to external testing using automatic test equipment (ATE). Pseudo-random testing is widely used in BIST by adopting a linear feedback shift register (LFSR) to generate random test patterns [8, 9]. However, the existence of random-pattern-resistant faults limits the coverage of pseudo-random testing. Two directions have been proposed to enhance the fault coverage achieved with pseudo-random BIST: 1) modify the CUT by inserting test points or redesigning the circuit [10], 2) modify the test pattern generator. Reseeding [11] and weighted pseudo-random patterns [12] are the two popular methods for BIST pattern generation.

In spite of numerous advantages of RAS, no BIST scheme has been studied in RAS environment to the best of our knowledge. In this paper, we propose a word oriented BIST scheme based on RAS architecture. In our scheme, one row of RAS scan-cells is written at each clock cycle with a random row address. Test power consumption can be reduced significantly due to reduced switching activity during scan operations. In our scheme high fault coverage can be achieved without a phase shifter [13]. We also provide an algorithm to reduce the test data volume and test application time. We compare the WOR-BIST implementation with serial scan based BIST by presenting the power, area and timing simulation results for several benchmark circuits.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.74

The paper has the following three contributions: 1) It extends the RAS architecture to a practical level by employing it in a real IC design flow, 2) The implementation of the WOR-BIST architecture carries out an extensive study of all aspects of the design flow including cell design and characterization, design synthesis, scan insertion, place&route and performance analysis etc., 3) It proposes an algorithm to reduce the test data volume and test application time, along with the mathematical formulation of the associated problem, while providing high fault coverage in BIST environment.

The rest of this paper is organized as follows: The WOR-BIST architecture is introduced in section 2. In section 3, we analyze the test data volume, test application time and test power consumption for our WOR-BIST scheme. The design flow using WOR-BIST architecture is introduced in section 4. Section 5 reports the experimental results for large ISCAS and ITC benchmark circuits. Finally, section 6 concludes the paper.

2. WOR-BIST architecture and test method

Figure 1 illustrates the proposed WOR-BIST architecture. It consists of the RAS scan-cells, a Row Address Decoder to provide the write and read row address, an LFSR to generate pseudo-random patterns, an Multiple Input Shift Register (MISR) to collect the responses of the circuit, a comparator to compare the signature from the MISR and the fault-free reference signature, a test control logic and ROM to store the seeds and run-lengths of each seed. The RAS scan-cells are configured as a $m \times n$ SRAM-like grid structure. More details of the RAS scan-cell can be found in [7]. The test-per-scan WOR-BIST scheme works as follows.

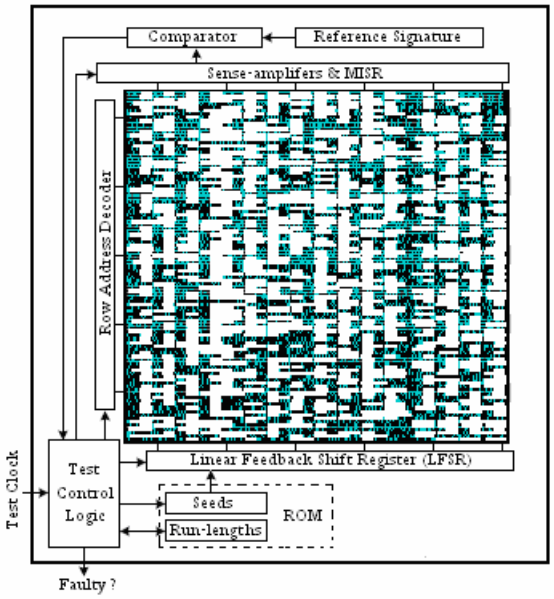


Figure 1. WOR-BIST architecture

First, the LFSR loads one seed from ROM and the run-length l_r of the corresponding seed is sent to the test control logic. The run-length of seed specifies the number of vectors that are to be generated by the same seed. The computation of run-length will be discussed in a later section. The size of the LFSR is decided by the number of columns (n_c) in the RAS scan-cells grid. After the LFSR is loaded with a seed, n_c bits are generated in each test clock cycle to be written into the scan-cells of the particular row whose address is provided by the Row Address Decoder. After number of rows (n_r) cycles, all the rows, and therefore every flip-flop, would have been written. When the system clock is enabled one test vector (PIs and PPIs) is applied to CUT. After that the responses of the CUT are collected row by row by the MISR in the same manner as they were written.

The same procedure is repeated for all the vectors generated by the seed. After l_r is reached the LFSR loads the next seed. This process is repeated until all the seeds in the ROM are applied. After all the responses are collected by the MISR, the signature produced by the MISR is compared with the reference signature, which is obtained from the response of the fault-free circuit. If the two signatures are different, the CUT is said to be faulty.

The pseudo code for the WOR-BIST scheme is given in Figure 2.

```
Initialize the Test Control Logic and LFSR.
For each seed s_i , ( i = 1 to n_seed )
    Load seed s_i into LFSR;
    Send l_r to Test Control Logic;
    For each vector v_i , ( i = 1 to l_r )
        For each row r_i , ( i = 1 to n_r )
            Write row r_i ;
        Endfor // write one vector
        Apply the PIs;
        Apply the system clock;
        For each row r_i , ( i = 1 to n_r )
            Read row r_i and update MISR;
        Endfor // read one vector
    Endfor // one seed applied
Endfor // all seeds applied
Compare the signatures and generate the result.
```

Figure 2. WOR-BIST algorithm

In LFSR-based pseudo-random BIST architecture, a phase shifter network is often adopted to eliminate the correlations of adjacent vectors which are generated by the same seed to improve the fault coverage [13]. The advantage of the WOR-BIST scheme is that high fault coverage can be achieved without the usage of phase shifter. In WOR-BIST, each row of the scan-cells can be

accessed individually and a Row Address Decoder is used instead of a Row Enable Shift Register [7]. This means that the row address generated at each cycle is not sequential from 1 to n_r, but can be in any random order. The experimental results in section 5 show that writing the rows in a random manner has the same effect as a phase shifter to eliminate the correlation of adjacent vectors and achieve high fault coverage.

3. Test cost analysis

3.1 Test data volume and test application time

In this section, we will analyze the test data volume and test application time of the proposed WOR-BIST scheme. A greedy algorithm is proposed to reduce the test application time and test data volume of the WOR-BIST scheme.

For a pseudo-random pattern generator with a constant number of seeds and constant run-length of each seed, the total number of vectors V is:

$$V = N \times L \tag{1}$$

where N is the number of seeds and L is the run-length. Thus in this case each seed is used to generate exactly L test vectors. Assuming the test clock period C_t is constant, the test application time is proportional to the total number of test clock cycles T. For the proposed BIST scheme, each vector needs n_r cycles to write all the rows, one cycle to apply the PIs, one cycle to apply the system clock and n_r cycles to read all the rows. After all the vectors have been applied, one extra cycle is needed to compare the signatures and generate the test outcome (Pass/Fail). Thus, T can be computed using the following equation:

$$T = N \times L \times (2 \times n_r + 2) + 1 \tag{2}$$

The performance of the pseudo-random testing is directly affected by the quality of the seeds. For seeds selected using a purely random procedure, experimental results have shown that after a certain number of seeds are applied, the remaining seeds can only improve the fault coverage by a very small percentage. Similarly, for each seed, after some vectors are generated, the remaining vectors can detect very few or no new faults.

To obtain good seeds and shorter run-length to achieve high fault coverage, a greedy heuristic is used to generate seeds and compute their run-lengths. The proposed greedy algorithm works as follows.

The run-length of one seed is determined by setting a fault coverage improvement threshold. For each vector generated by this seed, if the fault coverage improvement is smaller than the threshold, we stop generating vectors from the current seed.

For seeds selection procedure, n_s seeds are generated using a random number generator in each iteration. The run-length of the seed is determined using the method described above. The overall fault coverage of that seed can be acquired by applying all the vectors generated by that seed. Among all n_s seeds, the one with highest fault coverage is chosen as the seed in that iteration. The seed selection procedure stops when the desired fault coverage has been reached. A moderate value of n_s is chosen to contain the computation time. Better seeds may not be generated with too small n_s and too large n_s will result in unacceptable long run-time.

By using the greedy algorithm, the number of seeds N reduces to a smaller number N_g and the run-length for each seed is different. Assuming the run-length for a seed i is l_i, the total number of vectors V_g is equal to:

$$V_g = \sum_{i=1}^{Ng} l_i \tag{3}$$

For the proposed BIST scheme, because only the seeds are stored on the on-chip ROM, the test data volume reduction ratio R_v can be calculated by:

$$R_v = \frac{N - N_g}{N} \tag{4}$$

After applying the greedy heuristic, the total number of vectors reduces while the number of cycles to apply one vector remains the same, the reduced test application time T_g is:

$$T_g = \sum_{i=1}^{Ng} l_i \times (2 \times n_r + 2) + 1 \tag{5}$$

The test application time reduction ratio R_t is equal to:

$$R_t = \frac{(N \times L - \sum_{i=1}^{Ng} l_i) \times (2 \times n_r + 2)}{N \times L \times (2 \times n_r + 2) + 1} \tag{6}$$

3.2 Test power consumption

In CMOS circuits, the dominant portion of power consumption is the dynamic switching power of the circuit elements. For our WOR-BIST scheme, one row of scan-cells is written in one cycle. Thus in the worst case, at most n_c scan-cells change their states. However, in serial scan, all the scan-cells are updated during the shift operation and at most n_f number of flip-flops will switch the states, which results in high switching activities and high test power consumption in each test clock cycle.

In serial scan, the scan power consumption includes both the scan-in (write) and the scan-out (read) power consumption. The scan-in procedure and scan-out procedure are symmetric. For the WOR-BIST scheme proposed in this paper, the scan-out power consumption is almost negligible due to the advantage of our word-oriented BIST architecture. The test responses are directly sent to MISR hence there are no transitions on the scan-

cells and combinational gates during the scan-out procedure. The only contribution to the power consumption comes from the transitions in the MISR, Row Address Decoder and the Test Control Logic, which are very small.

4. Design flow

Figure 3 compares the design flow for the WOR-BIST architecture with the design flow using serial-scan based BIST. First, BIST logic is added to the given netlist. Then, for SS-based BIST, the serial scan insertion and synthesis can be done by some commercial tools, such as Synopsys DFT Compiler. For WOR-BIST, we added the RAS scan-cell to the standard cell library and then performed the scan insertion manually. The details of how to add the new cell and how to do the RAS scan insertion will be described in next section. The scanned netlist is synthesized and the post-synthesis netlist is sent to layout tools, like Cadence Encounter, to do the placement, routing and timing analysis. The important thing to note here is that both methods use similar design flow.

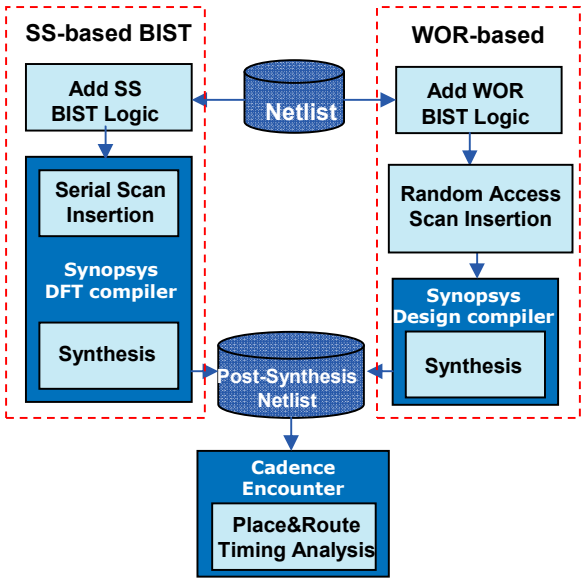


Figure 3. Design Flow

5. Experimental results

We applied the proposed WOR-BIST scheme and SS-based BIST to five ISCAS89 benchmark and four ITC99 benchmark circuits and performed the power, area and timing comparison. Table 1 gives the statistics of the circuits: the name of circuits, the number of PIs, the number of POs, the number of flip-flops, the number of combinational gates and the number of detectable faults, respectively.

Table 1. Statistics of circuits

Circuits	PIs	POs	FFs	Gates	Faults
s13207	62	152	638	7,951	9,664
s15850	77	150	534	9,772	11,336
s35932	35	320	1,728	16,065	35,110
s38417	28	106	1,636	22,179	31,015
s38584	38	304	1,426	19,253	34,797
b17	37	97	1,415	27,852	76,485
b20	32	22	490	20,226	45,395
b21	32	22	490	20,571	46,090
b22	32	22	735	21,772	67,472

5.1 Test time, volume and power study

To evaluate the performance of the greedy algorithm, we first generate 100 purely random seeds and set a constant run-length of 100. So the total number of vectors is 10,000. Assuming the fault coverage of these 10,000 vectors is *fc*, Table 2 summarizes the test data volume and test application time reduction using greedy algorithm to achieve the same *fc*. The second column lists the number of seeds N_g needed and the third column gives the average of run-length l_i. The fourth column gives the reduction of test data volume R_v as per equation (4) in section 3. Total number of vectors V_g is listed in the fifth column and the test application time reduction R_t is given in the sixth column according to the equation (6) in section 3. Finally, the fault coverage is given in the seventh column.

Table 2. Test data volume and test application time reduction

Circuit	N_g	l_i	R_v	V_g	R_t	*fc*
s13207	44	64	56%	2818	71.8%	89.4%
s15850	41	75	59%	3089	69.1%	92.7%
s35932	4	40	98%	163	98.4%	99.9%
s38417	49	88	51%	4302	57.0%	91.9%
s38584	55	70	45%	3861	61.4%	95.8%
b17	33	93	67%	3065	69.5%	65.7%
b20	48	86	52%	4137	58.6%	89.8%
b21	66	90	34%	5945	40.5%	88.4%
b22	53	82	47%	4352	56.5%	90.7%

In all the experiments, the fault coverage improvement threshold is defined as the average of newly detected faults by three successive vectors. When the average of newly detected faults by three successive vectors generated from the seed is less than 5, we stop generating vectors from this seed. Using the average of three successive vectors instead of one single vector avoids the unwanted very small run-length when there exists abnormally small value of newly detected faults at the beginning. In seed selection the value of n_s, the number of seeds generated in each iteration, is set to 8.

Table 3. Test power consumption reduction

Circuit	Average Switching Activities			Peak Switching Activities		
	SS-BIST	WOR-BIST	Reduction	SS-BIST	WOR-BIST	Reduction
s13207	2,533	203	92.0%	3,101	2,320	25.2%
s15850	2,470	234	90.5%	3,587	2,412	32.8%
s35932	4,873	507	89.6%	8,523	8,523	0%
s38417	8,368	378	95.5%	10,032	6,304	37.2%
s38584	5,154	305	94.1%	6,237	5,795	7.1%
b17	9,573	459	95.2%	11,035	7,027	36.3%
b20	6,236	352	94.4%	7,234	5,826	19.5%
b21	6,375	488	92.3%	8,739	7,239	17.2%
b22	7,942	403	94.9%	9,572	5,831	39.1%
Average	5,947	370	93.8%	7,562	5,697	24.7%

From the table, we observe that the proposed method can reduce the test data volume by 56.6% on average and achieve about 2.8X speed up in test application time. Notice that benchmark s35932 reaches almost 100% fault coverage after very few random patterns are applied. The reduction in test data volume and the reduction in test application time is a tradeoff. When the run-length of each seed is shortened, more seeds are needed. On the other hand, fewer seeds require relatively long run-length of each seed. It is possible to use deterministic patterns as seeds or as top-up stored vectors to obtain 100% fault coverage, but the results given in Table 2 still remain valid. As we set the number of scan chains of SS and the number of columns of WOR-BIST to be the same, the test application time and test data volume are identical for SS-Based BIST and WOR-BIST because they both write one row at each cycle.

Table 3 compares the test power consumptions. We measure the dynamic power consumption by counting the number of switching activities of combinational logics during scan operations for both the serial scan based BIST and the proposed WOR-BIST. The second block of the table shows the average switching activities and the third block compares the peak switching activities. The proposed WOR-BIST scheme can reduce average switching activities by 93.8% and peak switching activities by 24.7%. During the experiment, we also found that for the proposed WOR-BIST scheme, the peak switching activities always happen at the phase when the system clock is applied. The switching activities during the scan operations are very small compared to the peak value.

5.2 Area and performance study

To implement the WOR-BIST architecture on the above benchmarks, a new RAS flip-flop [7] was added to an existing standard cell library [14, 15]. A 0.18µm CMOS process was used for the technology mapping. A multiplexer-based serial scan flip-flop was also added to the library and both cell layouts were characterized using the characterization procedure outlined in [15] to enable fair comparison. The characterization results were compiled into the file formats required by the synthesis and place&route tools. The layout area for the RAS flip-flop is 36.7% smaller than the SS flip-flop. Figure 4 shows the schematic and layout of the RAS flip-flop.

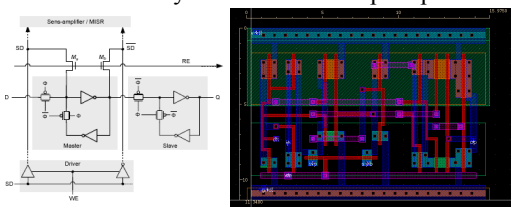


Figure 4. Schematic and Layout of RAS flip-flop

Table 4 shows the results of area, routing and timing simulation for SS-based BIST and WOR-BIST scheme. In this table we include two large ITC99 benchmarks, b18 and b19, to show the practicality of the proposed WOR-BIST scheme to larger circuits. As shown in figure 3, the Synopsys DFT compiler is used for SS insertion. For placement and routing, the Cadence Encounter is used. For a fair comparison, the number of columns in our scheme is set to be the same as the number of scan chains in SS. The area of WOR-BIST scheme is smaller because the WOR-BIST RAS cell is smaller than the conventional scan cell. For ISCAS benchmarks, area reduction is between 17% and 25%. For ITC99 benchmarks, area reduction is from 3% to 9%. The routing overhead is increased due to the extra pins in the RAS cell but it is still in an acceptable range. Increase in routing overhead is between 14% and 21% for ISCAS benchmarks and is between 6% and 15% for the ITC benchmarks. Also, it can be observed that WOR-BIST achieves better timing performance than SS-BIST in six out of nine benchmarks. The reason is that the multiplexer is removed from the critical path and the two access transistors needed for the RAS cell are not on the critical path, resulting in smaller setup time for the flip-flop. Even in the three cases in which WOR-BIST has lower timing performance the slack degradation is very small.

Table 4 Area, routing and timing results

Circuit	Area (μm^2)		Total routing wire-length ($10^6 \mu m$)		Best clock period/slack (ns)	
	SS-BIST	WOR-BIST	SS-BIST	WOR-BIST	SS-BIST	WOR-BIST
s13207	598 x 587	520 x 510	0.424	0.468	6/0.224	5/0.884
s15850	602 x 589	530 x 521	0.404	0.457	6/0.258	6/0.155
s35932	992 x 985	900 x 882	0.975	1.245	5/0.638	4/0.324
s38417	1,001 x 996	912 x 906	0.805	0.938	10/0.337	9/0.45
s38584	1,014 x 997	923 x 906	1.081	1.335	7/0.533	7/0.128
b17	1,378 x 1,363	1,313 x 1,306	1.898	2.034	18/0.783	17/0.145
b18	2,489 x 2,475	2,432 x 2,421	6.589	7.036	14/0.576	13/0.093
b19	3,477 x 3,423	3,415 x 3,387	13.112	14.380	25/0.173	23/0.626
b20	1,135 x 1,117	1,099 x 1,088	1.235	1.465	14/0.869	14/0.230
b21	1,124 x 1,109	1,092 x 1,088	1.220	1.433	14/0.337	14/0.543
b22	1,407 x 1,389	1,339 x 1,337	1.909	2.147	19/0.589	18/0.769

6. Conclusion

In this paper we proposed a word-oriented BIST architecture. In the WOR-BIST scheme, the scan-cells are written row by row in the random order. The average test power consumption is reduced dramatically because of reduced switching activity during the writing operations and almost negligible switching activity during the read operations. We also proposed a greedy algorithm to reduce the test data volume and test application time. Application of our scheme on large ISCAS89 and ITC99 benchmark circuits shows that nearly 94% reduction in average test power consumption, 53% reduction in testdata volume and 2.8X speedup in test application time. The area, routing and timing simulation results for the benchmark circuits show that WOR-BIST scheme is a comprehensive test solution for designs meeting power, area and performance requirements.

Acknowledgment

The authors would like to thank Professors Seiji Kajihara, Xiaoqing Wen, Kohei Miyase, and their students for their considerable help. This work is in part supported by National Science Foundation under Grant 6741641.

References

[1] X.Wen, Y. Yamashita, S. Kajihara, L.-T. Wang, K.K. Saluja, and K. Kinoshita, “On Low-Capture-Power Test Generation for Scan Testing”, Proc. VLSI Test Symposium, pp.265-270, May 2005

[2] H. Ando, “Testing VLSI with Random Access Scan”, Proc. of the COMPCON, pp.50-52, Feb. 1980

[3] D.H. Baik, K.K. Saluja, and S. Kajihara, “Random Access Scan: A Solution to Test Power, Test Data Volume and Test time”, Proc. International Conf. VLSI Design, pp.883-888, Jan. 2004

[4] B. Arslan and A. Orailoglu, “Test cost Reduction through a Reconfigurable Scan Architecture”, Proc. International Test Conf., pp.945-952, Oct. 2004

[5] A.S. Mudlapur, V.D. Agrawal and A.D. Singh, “A Random Access Scan Architecture to Redhuce Hardware overhead”, Proc. of ITC, Oct. 2005

[6] A.S. Mudlapur, V.D. Agrawal and A.D. Singh, “A Novel Random Access Scan Flip-Flop Design”, Proc. 9th VLSI Design and Test Symp., Aug. 2005

[7] D.H. Baik and K.K. Saluja, “Progress Random Access Scan: A simultaneous solution to test power, test data volume and test time”, Proc. of ITC, Oct. 2005

[8] P.Bardel, W.H. McAnney, and J. Savir, “Built-in test for VLSI”, Hohn Wiley and Sons, 1987

[9] B. Koenemann, “LFSR-Coded Test Patterns for Scan Design”, Proc. of ETC, pp.237-242, Apr. 1991

[10] N. Tamarapalli and J. Rajski, “Constructive Multiple-phase Test Point Insertion for Scan-based BIST”, Proc. of ITC., pp.604-611, Oct. 1996

[11] S. Hellebrand , J. Rajski etc. Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers, IEEE Trans. on Comp., v.44 n.2, p.223-233, February 1995

[12] A.Jas, C.V. Krishna, and N.A. Touba, “Weighted Pseudo-random Hybrid BIST”, IEEE Trans. VLSI Systems, 12(12), pp.1277-1283, Dec. 2004

[13] J. Rajski, N. Tamaraphalli, and J. Tyszer, “Automated Synthesis of Phase Shifters for Built-In Self-Test Applications”, IEEE Tran. on CAD, Vol. 19, No. 10, pp.1175-1187, Oct. 2000

[14] J. B. Sulistyo, J. Perry, and D. S. Ha, " Developing Standard Cells for TSMC 0.25um Technology under MOSIS DEEP Rules", Department of Electrical and Computer Engineering, Virginia Tech, Technical Report VISC-2003-01, Nov. 2003.

[15] Jos. B. Sulistyo and Dong S. Ha," A New Characterization Method for Delay and Power Dissipation of Standard Library Cells", VLSI Design 15 (3), pp. 667-678, 2002.

An Error Model to Study the Behavior of Transient Errors in Sequential Circuits

Karthikeyan Lingasubramanian and Sanjukta Bhanja
Nano Computing Research Group (NCRG)
University of South Florida
Tampa, Florida, USA
klingasu@mail.usf.edu, bhanja@eng.usf.edu

Abstract

In sequential logic circuits the transient errors that occur in a particular time frame will propagate to consecutive time frames thereby making the device more vulnerable. In this work we propose a probabilistic error model for sequential logic that can measure the expected output error probability, given a probabilistic input space, that account for both spatial dependencies and temporal correlations across the logic, using a time evolving causal network. We demonstrate our error model using MCNC and ISCAS benchmark circuits and validate it with HSpice simulations. Our observations show that, significantly low individual gate error probabilities produce at least 5 *fold higher output error probabilities. The average error percentage of our results with reference to HSpice simulation results is only* 4.43%*. Our observations show that the order of temporal dependency of error varies for different sequential circuits.*

1. Introduction

Nano-electronic devices have increased propensity to transient errors due to extremely low device dimensions. Irrespective of the basic technology (CMOS, SET, RTD, CNT, QCA, Nanowire, Magnetic RAM), all these devices are prone to inherent transient failures. In this work, we propose to model *sequential* logic circuits, where the fundamental logical elements are probabilistic logic as opposed to deterministic one. We focus mostly on error probability given an input probability space. In our model, every gate has equal chance of becoming erroneous unlike stuck-at fault models and single event upset models which deals with single gate errors. Our abstract error model can be used for any technology, given the base logic structure governing the circuit and the failure patterns.

There has been some significant works that deal with reliability in logic circuits. Probabilistic frameworks like Probabilistic Transfer Matrix (PTM) [6], Markov Random Field (MRF) [1] and probabilistic model checking [2] has been used for reliability modeling. Study on reliable computing using unreliable logics was first started by von Neumann in [11] where he found an error bound on a NAND logic and also proposed NAND multiplexing. This study has been enhanced and used for reliability modeling in nano-domain [10, 4]. The calculation of average and maximum output error probabilities was given in [9] and [7] respectively. All the exact probabilistic inferences are NP-hard for the combinational circuit itself. To our knowledge, sequential probabilistic logics are not dealt with, due to additional temporal complexity and modeling various masking effects that are device-sensitive.

In this work, we propose to use a time evolving probabilistic model (Temporal Dependency Model TDM) to handle the temporal behavior of the sequential circuits. We form the TDM model (Fig. 1(d)) by unrolling the basic probabilistic model, which represents the combinational block of a sequential circuit, into sufficiently large number of time slices and connecting the present state node of each time slice PS_{t_i} to the next state node of the previous time slice $NS_{t_{i-1}}$ thereby maintaining the temporal correlations. Then using TDM model, the error model is created based on the miter circuit technology where the ideal behavior of the circuit is compared with the erroneous behavior of the circuit.

2 Sequential Logic model

We model the sequential circuits into a time evolved probabilistic network, named as temporal dependency model (TDM), which handles temporal dependencies. In this section we provide the details on the modeling of a sequential logic into a TDM model.

2.1 TDM model

Let us consider the sequential circuit shown in Fig. 1(a) where the present state node is represented as PS, the next

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.73

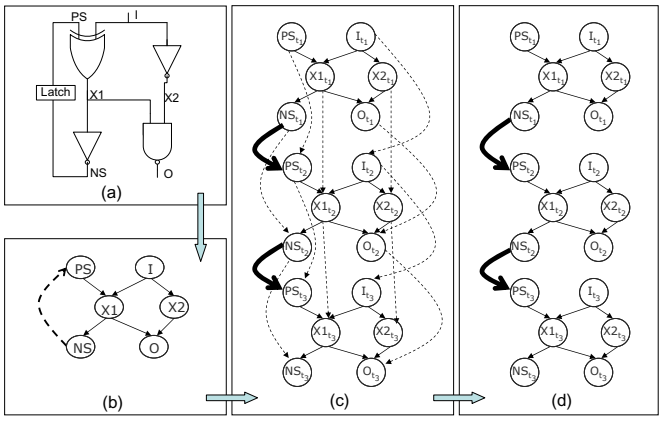


Figure 1. (a) Digital logic circuit (b) Corresponding probabilistic model (c) DAG representation which is not minimal (d) TDM model

state node is represented as NS, the primary input is represented as I, the primary output is represented as O and the internal nodes are represented as $X1$ and $X2$.

The equivalent probabilistic model shown in Fig. 1(c) can be represented by $G_{t_i} = (V_{t_i}, E_{t_i})$. The nodes of the probabilistic model, V, are the union of all the nodes for each time slice.

$$V = \bigcup_{i=1}^{n} V_{t_i} \tag{1}$$

where n is the number of time slices. In our example $V_{t_i} = \{PS_{t_i}, NS_{t_i}, I_{t_i}, O_{t_i}, X1_{t_i}, X2_{t_i}\}$. The edges, E, of the probabilistic model are not just the union of the edges in a single time slice, E_{t_i}, but also includes the edges between time slices, that is, temporal edges, $E_{t_i,t_{i+1}}$. It has to be noted that the copies of the same variable X_i in all time slices follow a markov property such that the following two sets $\{X_{i,t_1}, \cdots, X_{i,t_{i-1}}\}$ and $\{X_{i,t_{i+1}}, \cdots, X_{i,t_{i+k}}\}$ are independent given X_{i,t_i}. For example, in Fig. 1(c), $X1_{t_1}$ and $X1_{t_3}$ are independent of each other given $X1_{t_2}$. So the temporal edges can be defined as

$$E_{t_i,t_{i+1}} = \{(X_{i,t_i}, X_{i,t_{i+1}}) | X_{i,t_i} \in V_{t_i}, X_{i,t_{i+1}} \in V_{t_{i+1}}\} \tag{2}$$

where X_{i,t_i} is any node in time slice t_i and $X_{i,t_{i+1}}$ is the replica of the same node in the adjacent time slice t_{i+1} as shown in Fig. 1(c). Thus, the complete set of edges E is

$$E = E_{t_1} \cup \bigcup_{i=2}^{n} (E_{t_i} + E_{t_{i-1},t_i}) \tag{3}$$

In the probabilistic model (Fig. 1(c)), apart from the dependencies from one time slice, we also have the dependencies over two copies of the same variable X_j across adjacent time slices. But it is evident that X_{j,t_i} and $X_{j,t_{i-1}}$ are independent of each other given the present state node PS_{j,t_i}.

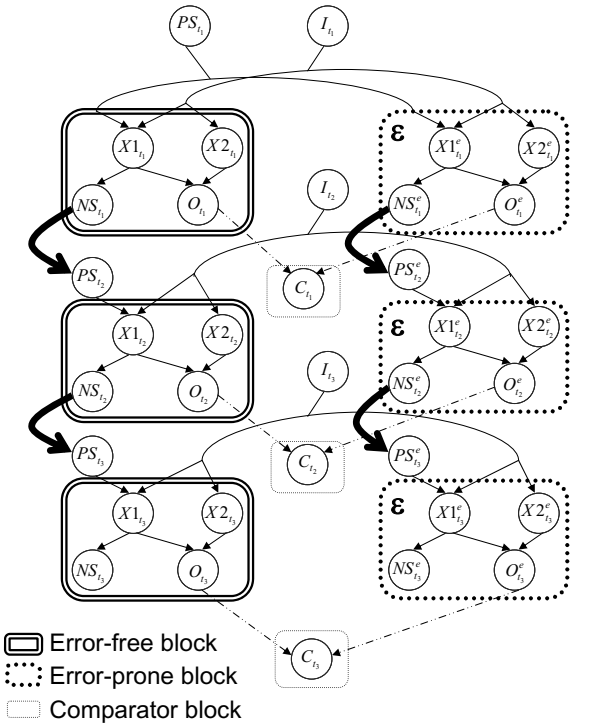


Figure 2. Error model obtained from TDM model with 3rd order temporal dependence

For example the nodes $X1_{t_1}$ and $X1_{t_2}$, from Fig. 1(c), are independent of each other given the present state node PS_{t_2}; so even if we remove the temporal edges connecting these nodes at consecutive time slices the underlying structure will still be intact. The same can be told for $X1_{t_2}$ and $X1_{t_3}$.

So in the probabilistic model all the temporal edges except those connecting the present state and next state nodes of adjacent slices (bold lines in Fig. 1(c)) can be removed to achieve a *minimal* representation as shown in Fig. 1(d), which is termed as the *TDM model*. In our example, the necessary temporal edges can be given as,

$$E_{t_i,t_{i+1}} = \{(NS_{t_i}, PS_{t_{i+1}}) | NS_{t_i} \in V_{t_i}, PS_{t_{i+1}} \in V_{t_{i+1}}\} \tag{4}$$

3 Error Model

From the TDM model of a given sequential circuit, an error model is designed where the erroneous behavior of the circuit is compared with the ideal error-free behavior of the circuit.

3.1 Structure

The error model contains three sections, (i) *error-free logic* where the gates are ideal, (ii) *error-prone logic* where

each gate goes wrong independently by an error probability ε and (iii) XOR based *comparator logic* that compare between the error-free and error-prone primary outputs. At first two copies of the TDM model, of the given sequential circuit, are created where one copy represents the error-free behavior of the circuit while the other represents erroneous behavior of the circuit. Fig. 2 illustrates the error model for the sequential circuit given in Fig. 1(a). The *Error-free block* includes nodes representing the ideal combinational part of all the time slices. The *Error-prone block* includes nodes representing the erroneous combinational part of all the time slices. At each time slice t_k an XOR logic based node C_{t_k} is added to compare between the error-free and error-prone primary outputs O_{t_k} and $O^e_{t_k}$ respectively. These additional nodes are included in the *Comparator block*. Note that at every time slice t_k both error-free and error-prone logic has to be fed from the same primary input node I_{t_k} and at the first time slice t_1 both error-free and error-prone logic has to connect to the same present state (PS) node PS_{t_1}. Also the present state nodes, PS_{t_k} and $PS^e_{t_k}$, for all time slices t_k are error-free, since we assume ideal latches. The comparator nodes C_{t_k} and the primary input nodes I_{t_k} for all time slices t_k are also assumed to be error-free.

3.2 Probability distributions of the random variables in error model

Any given probability function $P(x_1, x_2, \cdots, x_N)$ can be written as [1] $P(x_1, \cdots, x_N) = \prod_v P(x_v|Pa(X_v))$ where $Pa(X_v)$ are the parents of the variable X_v, representing its direct causes. This factoring of the joint probability function can be denoted as a graph with links directed from the random variable representing the inputs of a gate to the random variable representing the output. Our error model is one such graph structure where the probabilities $P(x_v|Pa(X_v)$ are provided by *Conditional Probability Tables* (CPTs) as shown in Table 1. It gives the CPTs for the nodes O_{t_k} whose parents are $X1_{t_k}$ and $X2_{t_k}$, and $O^e_{t_k}$ whose parents are $X1^e_{t_k}$ and $X2^e_{t_k}$ from Fig. 2. The nodes are governed by NAND logic.

The CPTs represent the underlying logic function of each gate. In this setup it is easier to incorporate the individual gate error probability ε by just changing the probabilities in the CPT. For example Table. 1 gives the CPTs for error-free O_{t_k} and error-prone $O^e_{t_k}$. In error-prone CPT we just have to replace the probability values 0 by ε and 1 by $1-\varepsilon$. This indicates that there is $(\varepsilon \times 100)\%$ chance for the signal to go to state "1" when it has to go to state "0" and $(\varepsilon \times 100)\%$ chance for the signal to go to state "0" when it has to go to state "1".

[1] Probability of the event $X_i = x_i$ will be denoted simply by $P(x_i)$ or by $P(X_i = x_i)$.

Table 1. Conditional Probabilistic Tables for Error-free and Error-prone NAND Logic

Error-free NAND			
$P(X1_{t_k})$	$P(X2_{t_k})$	$P(O_{t_k}=0)$	$P(O_{t_k}=1)$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0
Error-prone NAND			
$P(X1^e_{t_k})$	$P(X2^e_{t_k})$	$P(O^e_{t_k}=0)$	$P(O^e_{t_k}=1)$
0	0	ε	1-ε
0	1	ε	1-ε
1	0	ε	1-ε
1	1	1-ε	ε

Table 2. Conditional Probabilistic Table for Error-prone NAND Logic having variable gate error probabilities, ε_0 and ε_1

Error-prone NAND			
$P(X1^e_{t_k})$	$P(X2^e_{t_k})$	$P(O^e_{t_k}=0)$	$P(O^e_{t_k}=1)$
0	0	ε_1	1-ε_1
0	1	ε_1	1-ε_1
1	0	ε_1	1-ε_1
1	1	1-ε_0	ε_0

Also, in our model we can provide unequal gate error probabilities for any variable $X^e_{t_k}$ at any time slice t_k, such that if $P(X_{t_k} = 0) = 1$, then $P(X^e_{t_k} = 0) = 1 - \varepsilon_0$ and $P(X^e_{t_k} = 1) = \varepsilon_0$; if $P(Xt_k = 1) = 1$, then $P(X^e_{t_k} = 0) = \varepsilon_1$ and $P(X^e_{t_k} = 1) = 1 - \varepsilon_1$. The corresponding CPT of this implementation for an error-prone NAND logic is given in Table. 2. ε_0 is basically the error probability of logic "0" and ε_1 is the error probability of logic "1" at the output of a gate. Increasing ε_0 indicates that the circuit has more $0 \rightarrow 1$ errors, whereas increasing ε_1 indicates that the circuit has more $1 \rightarrow 0$ errors. With this implementation, we can use our error model to study the effect of these errors in the output of the circuit.

3.3 Inference scheme

The inference scheme basically calculates the joint probability distribution $P(x_1, \cdots, x_N)$ efficiently by propagating the probability distributions $P(x_v|Pa(X_v)$ of locally connected variables and thereby calculates the updated individual probability distributions of all random variables. The inference or propagation of belief on the probabilistic error model is done using the Hugin architecture [8, 5] which is an *exact* method. The inference on our model can be performed by forming clusters of nodes (*cliques*) which are directly dependent on each other and performing computations on those clusters, thereby enabling local computing. The network that is formed using these cliques is called *jointree*, where information can be propagated between cliques using message passing mechanism. Since extensive literature is already available, we will not be explaining the inference scheme in detail. Interested readers please refer to [8, 5].

In order to obtain a jointree, a *moral graph* is created from the error model, by adding undirected links between the parents of each common child node, and it is triangulated, to ensure that there are no cycles with more than three nodes, to obtain a *chordal graph*. Then the cliques are formed from the chordal graph and they are linked accordingly to form the jointree. Each adjacent cliques will have one or more common variables which are termed as *separators*.

To perform local computation, each clique C_i and separator set S_j are associated with probability potentials ϕ_{C_i} and ϕ_{S_j} respectively. At first, all clique and separator set potentials are initialized and for each variable Y_v, a particular clique C_i which contains Y_v along with its parents $Pa(Y_v)$ is selected and the conditional probability potential of Y_v from its CPT is multiplied to the clique potential ϕ_{C_i}.

$$\phi_{C_i} = \phi_{C_i} P(y_v | Pa(Y_v)) \tag{5}$$

After initialization the clique potentials are not consistent with their separator set potentials and so the joint probability of the entire network is not perfectly realized. To achieve this consistency *message passing* between cliques is performed where at first the marginal probability of the variables in the separator set S_r, between the message sending clique C_p and the receiving clique C_q, has to be computed from ϕ_{C_p} and then it is used to scale ϕ_{C_q}.

$$\phi_{S_r}^{updated} = \sum_{C_p \setminus S_r} \phi_{C_p}; \phi_{C_q} = \phi_{C_q} \frac{\phi_{S_r}^{updated}}{\phi_{S_r}} \tag{6}$$

where $C_p \setminus S_r$ are the set of variables in C_p that are not in S_r. The transmission of this scaling factor is the primary necessity for updating and message passing. Message passing in a jointree has to be done in both directions, from root to leaf termed as *outward pass* and from leaf to root termed as *inward pass*. An inward pass followed by an outward pass will completely update all the cliques in the jointree. Then the individual probability distribution for each variable can be calculated by choosing a clique C_i containing the variable Y_v and marginalizing its potential ϕ_{C_i} over all the other variables $C_i \setminus Y_v$ as, $P(y_v) = \sum_{C_i \setminus Y_v} \phi_{C_i}$.

4 Experimental Results

The output error probabilities for various sequential circuits are calculated using our experimental setup. We have performed our experiments on standard MCNC and ISCAS benchmark circuits. We have used HUGIN tool [5] to perform inference on the error model and we validate these results with equivalent HSpice simulation.

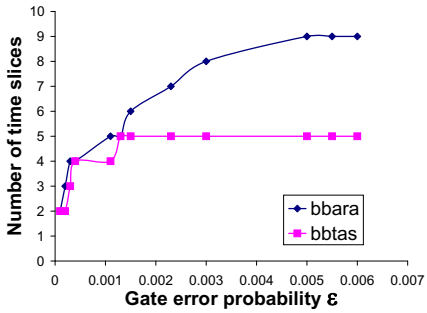


Figure 3. Number of time slices needed by $bbara$ and $bbtas$ for $\varepsilon = 0 - 0.006$

4.1 Experimental procedure

At first for a given ε value the probabilistic error model is obtained. The primary input nodes I_{t_k} for all the time slices t_k and the present state nodes PS_{t_1} for the first time slice t_1 are set to be equally probable to have state "0" or state "1". The model is then inferenced and the output error probability is obtained by noting the probability of state "1" at the comparator node, $P(C_{t_k} = 1)$ at every time slice t_k. This inference is an *exact* one and it also handles reconvergence and spatio-temporal dependencies. $P(C_{t_n} = 1)$ of the final time slice t_n and $P(C_{t_{n-1}} = 1)$ of the previous time slice t_{n-1} are checked for convergence. If they do not converge the time slices at both error-free and error-prone blocks are increased by 1 and the procedure is repeated. Thus the circuits are inferenced with different number of time slices iteratively and stopped when the output error probability values converge at consecutive time slices.

4.2 Output error probabilities

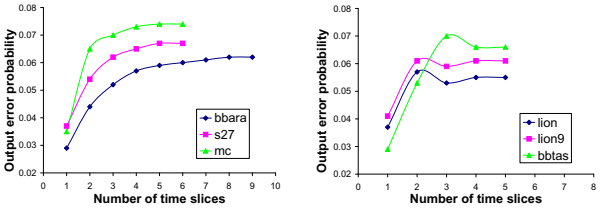
Table 3 gives the output error probabilities for gate error probabilities, $\varepsilon = 0.001, 0.003, 0.005, 0.01$. For a slight increase in ε value from 0.001 to 0.003, there is at least 2.87 fold increase in the corresponding output error probabilities. Also, for a considerably low influx of error at the gates for $\varepsilon = 0.005(0.5\%)$, the output error probability of most of the circuits exceed 3% with mc producing the highest output error probability of 3.99% which is almost 8 fold higher than the individual gate error probability. The same can be seen for $\varepsilon = 0.01(1\%)$, where the output error probability of most of the circuits exceed 6%.

4.3 Number of time slices

Fig. 3 shows the number of time slices needed by $bbara$ and $bbtas$ for $\varepsilon = 0 - 0.006$. It can be seen that the circuits needed less number of time slices for small ε values

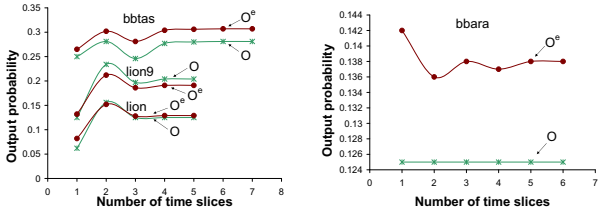
Table 3. Output error probabilities at $\varepsilon = 0.001, 0.003, 0.005, 0.01$ compared with HSpice simulation results.

	$\varepsilon = 0.001$			$\varepsilon = 0.003$			$\varepsilon = 0.005$			$\varepsilon = 0.01$		
Circuits	Error model	HSpice	% diff	Error model	HSpice	% diff	Error model	HSpice	% diff	Error model	HSpice	% diff
train11	0.0055	0.0057	3.51	0.0161	0.0159	1.26	0.0265	0.0263	0.76	0.0511	0.0497	2.82
lion	0.0060	0.0063	4.76	0.0177	0.0171	3.51	0.0288	0.0277	3.97	0.0545	0.0522	4.41
lion9	0.0069	0.0066	4.55	0.0200	0.0208	3.85	0.0326	0.0339	3.83	0.0614	0.0607	1.15
bbara	0.0074	0.0070	5.71	0.0213	0.0208	2.40	0.0341	0.0345	1.16	0.0621	0.0595	4.37
bbtas	0.0072	0.0069	4.35	0.0211	0.0203	3.94	0.0344	0.0354	2.82	0.0653	0.0671	2.68
s27	0.0075	0.0080	6.25	0.0220	0.0217	1.38	0.0357	0.0345	3.48	0.0676	0.0638	5.96
mc	0.0084	0.0088	4.55	0.0246	0.0250	1.60	0.0399	0.0391	2.05	0.0747	0.0733	1.91



(a) (b)

Figure 4. (a) Transition of output error probability across time slices for $bbara$, $s27$ and mc with $\varepsilon = 0.01$, (b) Transition of Output error probability across time slices for $lion$, $lion9$ and $bbtas$ with $\varepsilon = 0.01$



(a) (b)

Figure 5. (a) Transition of error-free and error-prone output probabilities across time slices for $bbtas$, $lion9$ and $lion$ with $\varepsilon = 0.01$, (b) Transition of error-free and error-prone output probabilities across time slices for $bbara$ with $\varepsilon = 0.01$

and then the needed number of time slices gradually increases along with ε value. For $bbtas$, the needed number of time slices gets set to 5 pretty fast at $\varepsilon = 0.0013$ while $bbtas$ takes up more time slices and finally gets set at 9 for $\varepsilon = 0.005$. The number of time slices needed is completely dependent on the circuit structure. The following studies will shed more light on this aspect.

4.4 Output error propagation across time slices

Fig. 4(a) & (b) gives the transition of output error probability across time slices for $\varepsilon = 0.01$. Here we show two sets of results that shows the difference in the transition of output error across time slices. Fig. 4(a) shows the output error transition for $bbara$, $s27$ and mc, where the output error increases gradually across time slices and finally gets converged. Whereas in Fig. 4(b), which shows the output error transition for $lion$, $lion9$ and $bbtas$, the output error reaches a maximum value and then gradually gets back to a steady value. This behavior can be attributed to the relation between the present state nodes and the primary input nodes which have random unbiased state distribution. If the present state nodes are closely connected to the input nodes resulting in having an unbiased state distribution, the output error will not be significant. With a biased state distribution in the present state nodes, the output error becomes more significant and reaches a maximum value in the early time slices as shown in Fig. 4(b).

Fig. 5(a) & (b) gives the transition of error-free (O) and error-prone (O^e) output probabilities across time slices for $\varepsilon = 0.01$. The results in Fig. 5(a) show that, for some circuits, the temporal dependence of the erroneous output conforms with that of the ideal error-free output. Whereas in circuits like $bbara$ this is not the case as shown in Fig. 5(b). Due to this, the output error probability in $bbara$ takes more time to converge.

4.5 Output error probabilities for $\varepsilon_0 \neq \varepsilon_1$

As we explained earlier in section 3.2, in our model we can provide unequal gate error probability values, ε_0 and ε_1, to study the effect of $0 \rightarrow 1$ and $1 \rightarrow 0$ errors on the output of a circuit. Fig. 6 gives the output error probabilities for ($\varepsilon_0 = 0.01, \varepsilon_1 = 0.02$) and ($\varepsilon_0 = 0.02, \varepsilon_1 = 0.01$). It can be clearly seen that when $\varepsilon_0 > \varepsilon_1$ the output error probabilities are higher for all circuits. This indicates that a $0 \rightarrow 1$ error can make the outputs more erroneous and signals that stay at logic "0" more often can be vulnerable to this effect. This might be favorable in CMOS technology, since the $0 \rightarrow 1$ bit flip is much harder than $1 \rightarrow 0$ bit flip due

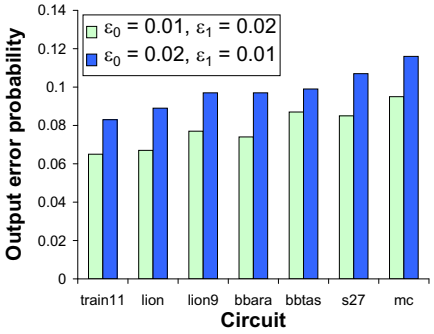


Figure 6. Output error probabilities for ($\varepsilon_0 = 0.01, \varepsilon_1 = 0.02$) and ($\varepsilon_0 = 0.02, \varepsilon_1 = 0.01$)

to the less error proneness of pMOS as compared to nMOS since holes are tougher to be dislodged by external particle bombardments. It can also signify that circuits with series pMOS connections can be less error prone as compared to circuits with parallel pMOS connections.

4.6 Validation using HSpice simulation

We validate our results by comparing them with HSpice simulation results. Even though, our error model can be used for any technology, the lack of benchmark circuits in any of the other emerging technologies has forced us to compare our model with simulations using 45nm CMOS technology. Using external voltage sources error can be induced in any signal and it can be modeled using HSpice [3]. In our HSpice model we have induced error, using external voltage sources, in every gate's output. Consider signal O_f is the original error free output signal and the signal O_p is the error prone output signal and E is the *piecewise linear* (PWL) voltage source that induces error. The basic idea is that the signal O_p is dependent on the signal O_f and the voltage E. Any change of voltage in E will be reflected in O_p. If $E = 0v$, then $O_p = O_f$, and if $E = Vdd(supply\ voltage)$, then $O_p \neq O_f$, thereby inducing error. The data points for the PWL voltage source E are provided by computations on a finite automata which incorporates the individual gate error probability ε. The width of every error pulse is fixed to $1ns$. The results are obtained by running the circuits for $5\ million$ random input vectors and sampling the comparator outputs. Table 3 gives the comparison between the output error probabilities obtained from inference in the error model and Hspice simulation for different circuits with gate error probability $\varepsilon = 0.001, 0.003, 0.005, 0.01$. The % difference is calculated as, ((Error model - Hspice) / Hspice) x 100. The highest relative difference between the inference results and HSpice results is just 6.25% and on an average the relative difference is only 4.43%.

5 Conclusion

We have proposed a compact probabilistic model that can handle error in sequential logic and we have presented experimental results on ISCAS and MCNC benchmark circuits. We have observed that for low gate error probabilities like $\varepsilon = 0.005(0.5\%)$, the output error probabilities are at least 5 fold higher and at most 8 fold higher. Also, our observations showed that the degree of temporal dependence differs for various sequential circuits. Another interesting observation indicated that $0 \rightarrow 1$ errors affects the circuit output more than $1 \rightarrow 0$ errors. We have also validated our model using HSpice simulation results and the average % difference is only 4.43%. Our future effort will be to obtain real time ε values through device physics and fabrication to model error. Also we will explore the error masking effects for various noise width and magnitude through a statistical study of sequential circuits.

References

[1] R. I. Bahar, J. Mundy, and J. Chen. A probabilistic based design methodology for nanoscale computation. *International Conference on Computer Aided Design*, pages 480–486, Nov 2003.

[2] D. Bhaduri and S. K. Shukla. Nanoprism: A tool for evaluating granularity vs. reliability trade-offs in nano architectures. *Great Lakes Symposium on VLSI*, pages 109–112, 2004.

[3] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. S. Akgul, and L. N. Chakrapani. A probabilistic cmos switch and its realization by exploiting noise. *IFIP International Conference on VLSI*, 2005.

[4] J. Han, J. Gao, P. Jonker, Y. Qi, and J. A. B. Fortes. Toward hardware-redundant fault-tolerant logic for nanoelectronics. *IEEE Transactions on Design and Test of Computers*, 22(4):328–339, July-Aug 2005.

[5] Hugin inference tool. http://www.hugin.com/.

[6] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes. Accurate reliability evaluation and enhancement via probabilistic transfer matrices. *Design Automation and Test in Europe (DATE)*, 1:282–287, March 2005.

[7] K. Lingasubramanian and S. Bhanja. Probabilistic maximum error modeling for unreliable logic circuits. *ACM Great Lakes Symposium on VLSI*, pages 223–226, 2007.

[8] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[9] T. Rejimon and S. Bhanja. Scalable probabilistic computing models using bayesian networks. *IEEE Midwest Symposium on Circuits and Systems*, 1:712–715, Aug 2005.

[10] S. Roy and V. Beiu. Majority multiplexing-economical redundant fault-tolerant designs for nano architectures. *IEEE Transactions on Nanotechnology*, 4(4):441–451, July 2005.

[11] J. von Neumann. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In Automata Studies *by C. E. Shannon, W. R. Ashby and J. McCarthy*, pages 43–98. Princeton University Press, Princeton, N.J., 1954.

Session 8A

Advanced Nanodevice Modeling

Analysis Of The Energy Quantization Effects On Single Electron Inverter Performance Through Noise Margin Modeling

Surya Shankar Dan[1] and Santanu Mahapatra[2]
Nano-Scale Device Research Laboratory, Centre for Electronics Design and Technology
Indian Institute of Science, Bangalore - 560012
[1]dsurya@cedt.iisc.ernet.in & [2]santanu@cedt.iisc.ernet.in

Abstract

Possible integration of Single Electron Transistor (SET) with CMOS technology is making the study of semiconductor SET more important than the metallic SET and consequently, the study of energy quantization effects on semiconductor SET devices and circuits is gaining significance. In this paper, for the first time, the effects of energy quantization on SET inverter performance are examined through analytical modeling and Monte Carlo simulations. It is observed that the primary effect of energy quantization is to change the Coulomb Blockade region and drain current of SET devices and as a result affects the noise margin, power dissipation, and the propagation delay of SET inverter. A new model for the noise margin of SET inverter is proposed which includes the energy quantization effects. Using the noise margin as a metric, the robustness of SET inverter is studied against the effects of energy quantization. It is shown that SET inverter designed with $C_T : C_G = 1/3$ (where C_T and C_G are tunnel junction and gate capacitances respectively) offers maximum robustness against energy quantization.

1. Introduction

The concept of hybridization of Single Electron Transistor (SET) with Complementary Metal Oxide Semiconductor (CMOS) technology has attracted much attention [1, 2, 3] as such integration can offer new functionalities, which are very difficult to achieve either by pure CMOS or by pure SET approaches. As a result, silicon SETs are appearing to be more promising than metallic SETs for their possible integration with CMOS. SETs are normally studied on the basis of the classical Orthodox Theory [4], where quantization of energy states in the island is completely ignored. Though this assumption greatly simplifies the physics involved, it is valid only when the SET is made of metallic island. As one cannot neglect the energy quantization in a semiconductive island, it is extremely important to study the effects of energy quantization on silicon SET logic performance.

2. Single Electron Transistors With Discrete Energy States

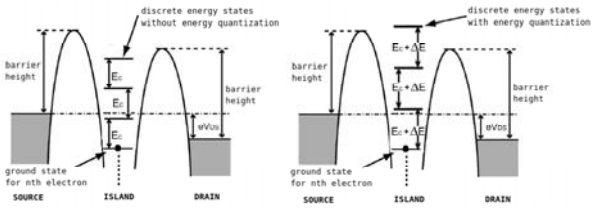


Figure 1. Schematic of the energy diagram of an SET with and without energy quantization.

There are two contributions to the energy gaps in the quantized SET: one, due to the electrostatic energy, known as "Addition Energy", and the other due to quantization of the energy states electrons can occupy, often referred to as the "Excitation Energy". We have considered the simplest case of parabolic potential well approximation,which makes the individual differences between subsequent energy states in the island constant for all the discrete states within the dot. Therefore we can say that the energy state in the island where the electron can tunnel into (or from) gets perturbed by an amount ΔE due to energy quantization. Before the introduction of quantization, the energy states where electrons could tunnel into (or from) were separated by gaps given only by the electrostatic energy giving rise to the standard Coulomb Blockade in case of metallic SETs. After the introduction of quantization, the separation between subsequent energy states in the island where the electron can

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.34

tunnel into (or from), has increased by an amount equal to the energy quantization as shown in Fig.1. Now, the net energy change ΔF of the tunneling electron must take into account the cumulative effect of both electrostatic as well as the quantum effects, *as if* the Coulomb Blockade has increased. From this *compact* formulation, we can see that the separate treatment of the 'addition energy' and 'excitation energy' is not necessary.

3. Calibration Of The Monte Carlo Simulator

3.1. Comparison Between Orthodox And Quantum Models Of Single Electron Tunneling

In the orthodox theory of single electron tunneling [4], the electron energy quantization in the island is ignored, i.e., the electron energy spectrum is assumed to be continuous. In this model, the transition (tunneling) rate $\Gamma(\Delta F)$ is $\Gamma(\Delta F) = \Delta F / \left[e^2 R_T \{1 - exp(-\Delta F/k_B T)\}\right]$, where ΔF denotes the change in Gibb's free energy of the electron during tunneling, e,k_B and T denote the electron charge, Boltzmann constant and the temperature (in Kelvin) respectively. In the non-orthodox quantum model, $\Gamma(\Delta F)$is given by [4, 5, 6] $\Gamma(\Delta F) = \Gamma_0 / [1 + exp(-\Delta F/k_B T)]$, where Γ_0 is the seed tunneling rate. The characteristics of the single electron transistors can be calculated by putting the appropriate values of $\Gamma(\Delta F)$ from the above transition rate equations into the steady state master equation [3, 6, 7].

3.2. Monte Carlo Simulation of SET with Discrete Energy States

One may conceptualize a metallic SET to be equivalent to a non-metallic one in which the energy states of the island extend from lower bound $E_{min} \rightarrow -\infty$ to upper bound $E_{max} \rightarrow +\infty$ with the energy gaps between successive energy states $\Delta E \rightarrow 0$. In order to study the energy quantization effects, we first simulate an SET with metallic (continuous energy spectrum) island for a particular set of device parameters (C_G,$C_T \sim aF$ and $R_T \sim M\Omega$ where C_G and C_T are the gate and tunnel junction capacitance while R_T is the tunnel junction resistance respectively). Then, for the same set of device parameters, we simulate a non-metallic SET with discrete states, where $E_{min} = -1eV$, $E_{max} = 1eV$ and $\Delta E = 0.01meV$. As E_{min} and E_{max} values are much larger and ΔE is much smaller than the charging energy ($\sim 40meV$) of the SET, we can expect that such a device should behave as metallic SET if the W and H parameters are properly tuned.

In order to calculate the total tunnel rate through a discrete energy state, one typically starts from Fermi's golden rule, as is done in orthodox theory, and then summed up over all possible states on both sides of the barrier, which spectrum a sum of δ functions. A more realistic treatment is to include finite lifetime broadening which introduces Lorentzian functions of the form [8]

$$D(E) = \frac{\hbar}{2\pi} \Sigma_n \frac{\Gamma}{(E - E_n)^2 + (\hbar\Gamma/2)^2} \qquad (1)$$

where E_n are the energy levels. For $\Gamma \rightarrow 0$, the Lorentzian approaches the δ function. SIMON (a popular Monte Carlo simulator) allows us to insert the Γ values through the height H and the width W parameters. After exhaustive simulations we have found that for $H = 0.04$ and $W = 0.001$, the $I - V$ characteristics of the non-metallic SET with discrete energy states completely super-imposes over the characteristics obtained from the metallic SET and these values of H and W are completely independent of device capacitances and resistances as long as C_G, $C_T \sim aF$ & $R_T \sim M\Omega$. Using these calibrated values of H and W, keeping E_{max} and E_{min} constant, we increase the value of ΔE in order to simulate the effects of energy quantization on SET device and inverter performance.

4. Results And Discussions

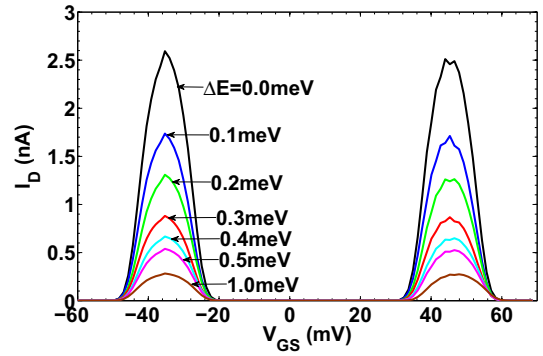
In this paper, quantization effects are studied by gradually increasing the energy gaps ΔE between successive energy levels in the island of a pre-calibrated SET as described in section 3. In this work ΔE is treated as an *electrical parameter* and we use parabolic potential well so that all ΔE are equal to avoid complication. When energy quantization is introduced, the net change in the free energy of electrons during tunneling becomes the sum of electrostatic energy contributed by coulomb blockade and the energy gaps between the quantized energy levels. Consequently, including the quantization term ΔE into the expression for the net electron energy change ΔF we obtain [3, 5]

$$\begin{matrix} \Delta F_{s,i} \\ \Delta F_{i,s} \end{matrix} = \frac{e}{C_\Sigma} \left[\pm C_T V_{DS} \pm C_G V_{GS} \mp en - \frac{e}{2} \right] - \sum_1^n \Delta E \qquad (2)$$

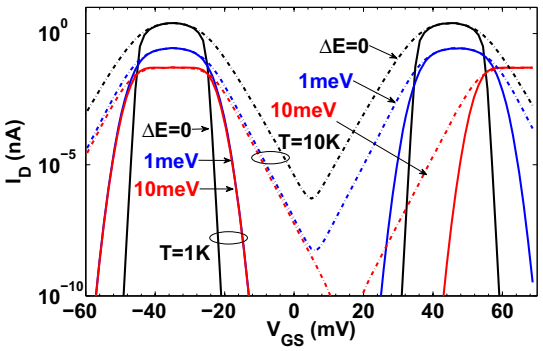
$$\begin{matrix} \Delta F_{i,d} \\ \Delta F_{d,i} \end{matrix} = \frac{e}{C_\Sigma} \left[\pm (C_T + C_G) V_{DS} \mp C_G V_{GS} \pm en - \frac{e}{2} \right] - \sum_1^n \Delta E \qquad (3)$$

Throughout this paper, in multiple equations like (2) and (3), the upper term in the left hand side equates the upper symbol sequence on the right hand side and vice-versa for the lower term in the left hand side. Here n denotes the number of discrete energy states in the island and $\Delta F_{initial,final}$ denotes the net free energy change for the electron tunneling from '$initial$' to '$final$' which may be any of the source 's', island 'i' or drain 'd' regions.

4.1. Analysis Of The Energy Quantization Effects On SET Device Performance



(a)



(b)

Figure 2. Influence of energy quantization ΔE on $I_D - V_{GS}$ characteristics of an SET plotted on (a) linear and (b) logarithmic scale (simulated for $V_{DS} = 20mV$, tunnel resistance $R_T = 1M\Omega$, gate capacitance $C_G = 2aF$, and tunnel capacitance $C_T = 1aF$ and temperature $T = 1K$).

It is evident from Fig.2(a) that increasing energy quantization ΔE reduces the drain current I_D, *as if* the effective tunnel resistance R_T has increased while Fig.2(b) shows that the plots shift towards the right with increasing ΔE, thus increasing the Coulomb Blockade region. It is inferred that energy quantization increases the coulomb blockade periodicity from $e/2C_G$ to $e/2C_G + \Delta E/e$. Fig.2(b) also shows that increase of temperature T reduces the coulomb blockade region. Figure 3 shows the influence of energy quantization on the $I_D - V_{DS}$ characteristics of the SET device, which also shows the decrease of drain current with increasing energy quantization.

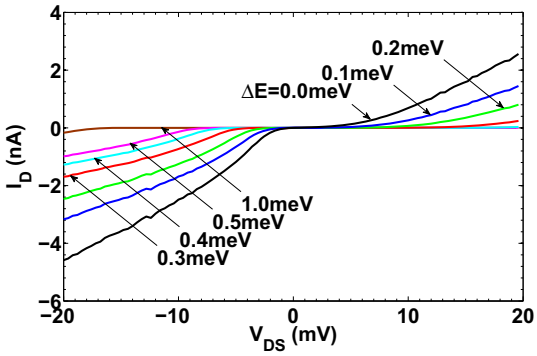


Figure 3. Influence of energy quantization ΔE on $I_D - V_{DS}$ characteristics of an SET (simulated for $V_{GS} = 1V$, $R_T = 1M\Omega$, $C_G = 2aF$ and $C_T = 1aF$).

4.2. Analysis Of The Energy Quantization Effects On SET Inverter Performance

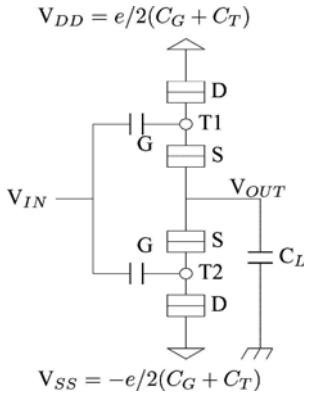


Figure 4. Schematic diagram of a voltage-state SET inverter (SET_1 and SET_2 are identical and have C_T, $C_G \sim aF$, $R_T \sim M\Omega$ and load capacitance $C_L \sim fF$).

Fig.4 shows the schematic of the voltage-state SET inverter used in this work, where both the transistors T_1 and T_2 are completely identical in terms of device capacitances and resistances. Fig.5 shows the influence of energy quantization on SET inverter transfer (V_{OUT} vs. V_{IN}) characteristics. These plots indicate that increasing energy quantization ΔE shifts the inverter V_{OUT} vs. V_{IN} characteristics towards the right, implying that larger input voltage V_{IN} is required for switching of a non-metallic SET with quantized energy states than its metallic counterpart. This

is analogous to the influence of fixed positive background charges in the island, as described in [11].

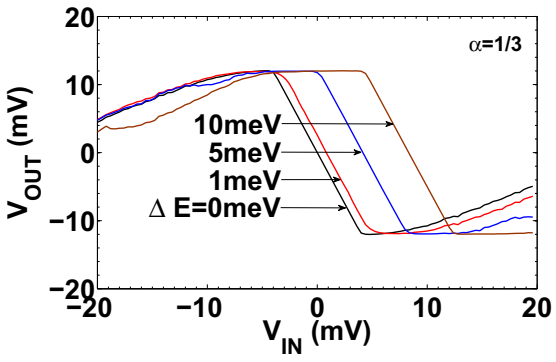


Figure 5. Influence of energy quantization ΔE on V_{OUT}vs.V_{IN} characteristics for $C_G = 3aF$, $C_T = 1aF$ ($\alpha = C_T/C_G = 1/3$) (simulated for $V_{DD} = 20mV$, $V_{SS} = -20mV$, $R_T = 1M\Omega$, and $C_L = 1fF$ at $T = 1K$).

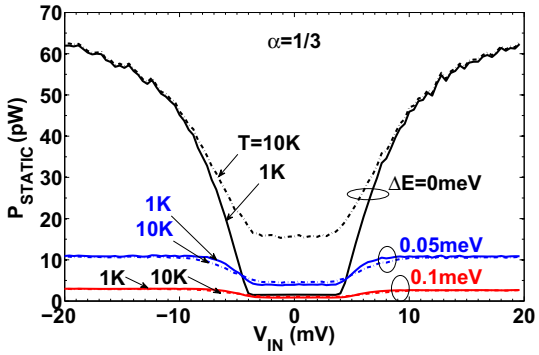


Figure 6. Variation of static output power dissipation P_{STATIC} in pW with input voltage V_{IN} in mV using energy quantization ΔE in meV and temperature T in K as the metrics. (simulated for $V_{DD} = 20mV$, $V_{SS} = -20mV$, $R_T = 1M\Omega$ and $C_G = 3aF$, $C_T = 1aF$ *(i.e.* $\alpha = 1/3$), $C_L = 1fF$ at $T = 1K$).**

Fig.6 shows the influence of capacitance ratio α, energy quantization ΔE and temperature T on the static power P_{STATIC} dissipated by the SET inverter. Here P_{STATIC} is calculated as $P_{STATIC} = (V_{DD} - V_{SS})\, I_{STATIC}$ where I_{STATIC} is the steady state current flowing from V_{DD} to V_{SS}[3, 11]. It can be inferred Fig.6 that there is an enormous reduction in the power dissipation due to the degradation of I_D with increasing energy quantization and that power dissipation increases with increasing temperature because of the increase of leakage current in Coulomb Blockade region as shown in Fig.2(b).

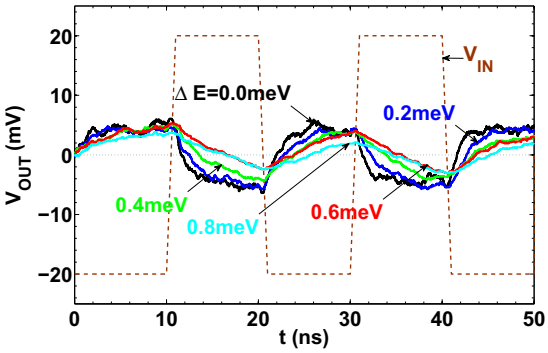


Figure 7. Influence of energy quantization ΔE on delay characteristics of the SET inverter (simulated for $V_{DD} = 20mV$, $V_{SS} = -20mV$, $R_T = 1M\Omega$, $C_G = 2aF$, $C_T = 1aF$, $C_L = 1fF$ at $T = 1K$).

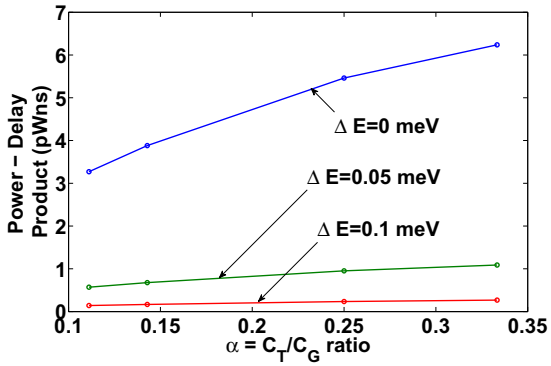


Figure 8. Variation of power - delay product (in pWns) with capacitance ratio $\alpha = C_T/C_G$ using different values of energy quantization ΔE(meV) as metric.

Fig.7 shows the variations of V_{OUT} with time t indicating the effect of energy quantization on the propagation delay of the SET inverter. It can be seen that energy quantization simply deteriorates the delay characteristics as the drive current decreases with increasing ΔE. Fig.8 shows the variation of power - delay product with different values of the capacitance ratio $\alpha = C_T : C_G$ using different values of ΔE as the metric. It is seen that power-delay product

decreases with increasing energy quantization as well as decreasing α.

5. Modeling The Noise Margin Of SET Inverter With Quantized Energy Levels In The Islands

5.1. Development Of The Analytical Model

The expressions for orthodox noise margin (NM) parameters were derived in [11] as

$$V_{OH} = -V_{OL} = \frac{\alpha V_{DD}}{2\alpha^2 + \alpha + 1} \tag{4}$$

$$V_{IH} = -V_{IL} = \frac{\alpha^2 V_{DD}}{2\alpha^2 + \alpha + 1} \tag{5}$$

$$NM = NM_H = NM_L = \frac{\alpha\left(1-\alpha\right)V_{DD}}{2\alpha^2 + \alpha + 1} \tag{6}$$

Inclusion of energy quantization alters the above noise margin parameters into the following expressions

$$\begin{matrix} V'_{OH} \\ V'_{OL} \end{matrix} = \begin{matrix} V_{OH} \\ V_{OL} \end{matrix} - \left(\frac{1}{2\alpha^2 + \alpha + 1}\right)\frac{\Delta E}{e} \tag{7}$$

$$\begin{matrix} V'_{IH} \\ V'_{IL} \end{matrix} = \begin{matrix} V_{IH} \\ V_{IL} \end{matrix} + \left(\frac{2\alpha^2 + 2\alpha + 1}{2\alpha^2 + \alpha + 1}\right)\frac{\Delta E}{e} \tag{8}$$

Throughout this paper, *primed variables* refer to the quantities including energy quantization effects, while unprimed variables refer to the ideal classical situation with the metallic SETs, following the orthodox theory of single electron tunneling. Equations (8) indicate that the entire transfer characteristics shift towards the right by the amount $\Delta E\left(2\alpha^2 + 2\alpha + 1\right)/\left(2\alpha^2 + \alpha + 1\right)e$ with respect to the orthodox characteristics. Equations (7) show that both the output voltage levels decrease simultaneously by $\Delta E/\left(2\alpha^2 + \alpha + 1\right)e$. From the basic definitions of noise margins $NM_H \triangleq V_{OH} - V_{IH}$ and $NM_L \triangleq V_{IL} - V_{OL}$ we obtain

$$\begin{matrix} NM'_H \\ NM'_L \end{matrix} = NM \mp 2\left(\frac{\alpha^2 + \alpha + 1}{2\alpha^2 + \alpha + 1}\right)\frac{\Delta E}{e} \tag{9}$$

In eq.(9) '$-$' refers to the NM'_H while '$+$' refers to the NM'_L relations. From (9) it is evident that energy quantization increases the noise margin for low logic and decreases the noise margin for high logic by the same amount $2\Delta E\left(\alpha^2 + \alpha + 1\right)/\left(2\alpha^2 + \alpha + 1\right)e$. Thus it is seen that for the continuous spectrum ($\Delta E = 0$), (9) reduces to orthodox model of noise margin as proposed in our earlier work [11]. It is worth noting that in (9), the change in noise margins due to energy quantization i.e. $2\Delta E\left(\alpha^2 + \alpha + 1\right)/\left(2\alpha^2 + \alpha + 1\right)e$, vary intensely with α.

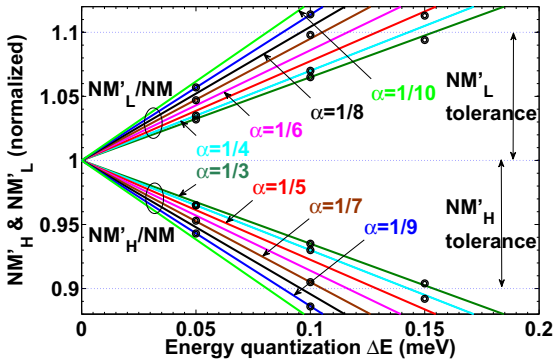


Figure 9. Variation of normalized noise margins NM'_H/NM and NM'_L/NM of the SET inverter with energy quantization ΔE for different $\alpha = C_T/C_G$ ratios at $V_{DD} = -V_{SS} = 20mV$. The symbols represent simulated data while solid lines indicate the results predicted by the model (noise margin tolerance is taken as 10% of the orthodox value).

From the inverter transfer characteristics simulated in SIMON for different α and ΔE, the values of V_{OH},V_{OL},V_{IH} and V_{IL} are recorded and the corresponding NM_H and NM_L are calculated. These values are plotted in Fig.9, and it shows the variation of normalized noise margins with energy quantization. Figure 9 also demonstrates the excellent agreement between the proposed model (9) and the simulated results.

5.2. Robustness Of SET Logic Inverter Against Energy Quantization

The maximum allowable energy quantization ΔE_{max} which the SET inverter can withstand before the noise margin falls below the specific tolerable value can be tuned by changing α. Equating the relation for normalized noise margin with the tolerable value (which is taken to be 10% in this work), we get

$$\frac{NM'_H}{NM} = 1 - \frac{2}{NM}\left(\frac{\alpha^2 + \alpha + 1}{2\alpha^2 + \alpha + 1}\right)\frac{\Delta E}{e} = 0.9 \tag{10}$$

from which we finally obtain

$$\Delta E_{max} = \frac{e\alpha V_{DD}\,(1-\alpha)}{20\,(\alpha^2+\alpha+1)} \tag{11}$$

The optimal C_T/C_G ratio of the SET inverter circuit for which the maximum robustness can be achieved, is calculated by maximizing (11). Now solving $\partial \Delta E_{max}/\partial\alpha = 0$ it is found that the condition for maximum robustness occurs at $\alpha = 0.366$ and the maximum energy quantization an SET can tolerate is$\Delta E_{max} = 0.1547meV$. It is worth noting that in our earlier paper [11] we had shown that $C_T/C_G = 1/3$ design criteria also provides maximum robustness against background charge and device parameter variation. On the other hand, for a given ΔE, the α value, for which one can get maximum noise margin, can be obtained from the equation $\partial NM'_H/\partial\alpha = 0$. It results in the following quadratic relationship between α and ΔE.

$$\alpha\,(\Delta E)\Big|_{NM'_{max}} = \left[\pm 1 + \sqrt{\left(\frac{eV_{DD}}{\Delta E}\right)^2 + 1 \mp \frac{3}{2}\left(\frac{eV_{DD}}{\Delta E}\right)} - \frac{1}{2}\left(\frac{eV_{DD}}{\Delta E}\right)\right] \Big/ \left[\frac{3}{2}\left(\frac{eV_{DD}}{\Delta E}\right) \mp 1\right] \tag{12}$$

Here $(+,-,-)$ sequence is used for $NM'_{H_{max}}$ and $(-,+,+)$ sequence is used for $NM'_{L_{max}}$. Eq.(12) indicates that one needs to design SET with lower α as energy quantization increases.

6. Conclusion

Using analytical models and Monte Carlo simulation the effect of energy quantization on Single Electron Transistor device and logic inverter is studied. It is observed that energy quantization in SET Island mainly changes the Coulomb Blockade region and the drain current of SET devices and thus it affects the noise margin, power dissipation, and the propagation delay of SET inverter. Including energy quantization term a new noise margin model for SET inverter is proposed and validated against Monte-Carlo simulation. This noise margin model is then used to study the robustness of the SET inverter against energy quantization effects. It is found that SET inverter designed with $C_T : C_G \sim 1/3$ offers the maximum robustness against energy quantization and the maximum tolerable value of energy quantization is found to be $0.1547meV$ for 10% deviation over orthodox noise margin.

7. Acknowledgement

This work is supported by the Council of Scientific and Industrial Research (CSIR), India under Grant 22 (0453)/07/EMR II.

References

[1] A.M. Ionescu, M.J. Declercq, S. Mahapatra, K. Banerjee, and J. Gautier, “Few Electron Devices: Towards Hybrid CMOS-SET Integrated Circuits”, Proceedings of 39th Design Automation Conference, pp 323–326, 2002.

[2] S. Mahapatra, V. Vaish, C. Wasshuber, K. Banerjee, and A.M. Ionescu, “Analytical Modeling of Single Electron Transistor for Hybrid CMOS-SET Analog IC Design”, IEEE Transactions on Electron Devices, pp 1772–1782, 2004.

[3] S. Mahapatra and A. M. Ionescu, “Hybrid CMOS Single Electron Transistor Device and Circuit Design”, Artech House Publication, ISBN 1- 59693-069-1, 2006.

[4] K.K. Likharev, “Single-Electron Devices and Their Applications”, Proceedings of the IEEE, vol. 87, no. 4, pp. 606–632, 1999.

[5] K. Miyaji, M. Saitoh and T. Hiramoto, “Compact Analytical Model for Room-Temperature Operating Silicon Single-Electron Transistors with Discrete Quantum Levels”, IEEE transactions on Nanotechnology, vol. 5, no. 3, pp. 167–173, 2006.

[6] H. Averin, A.N. Korotkov, “Correlated Single-Electron Tunneling via Mesoscopic Metal Particles: Effects of the Energy Quantization”, Journal of Low Temperature Physics, vol. 80, no. 3/4, pp. 173–185, 1990.

[7] H. Inokawa, Y. Takahashi, “A Compact Analytical Model for Asymmetric Single Electron Tunneling Transistors”, IEEE Transactions on Electron Devices, vol. 50, no. 2, pp. 455–461, 2003.

[8] Christoff Wasshuber, “Computational Single-Electronics”, Springer New York.

[9] C.Wasshuber, H. Kosina, and S. Selberherr, “SIMON-A Simulator for Single-Electron Tunnel Devices and Circuits”, IEEE transactions on Computer- Aided Design of Integrated Circuits and Systems, vol. 16, no. 9, pp. 937– 944, 1997.

[10] Personal communication with C. Wasshuber concerning the operation of SIMON simulator.

[11] C. Sathe, S.S. Dan and S. Mahapatra, “Assessment of SET Logic Robustness through Noise Margin Modeling”, IEEE Transactions on Electron Devices, vol. 55, no. 3, pp 909–915, 2008.

2009 22nd International Conference on VLSI Design

Exploring Carbon Nanotube Bundle Global Interconnects for Chip Multiprocessor Applications

Sudeep Pasricha*, Nikil Dutt†, Fadi J. Kurdahi†,

*Colorado State University, Fort Collins, CO
sudeep@engr.colostate.edu

†University of California, Irvine, CA
{dutt, kurdahi}@uci.edu

Abstract

The current paradigm of using Cu interconnects for on-chip global communication is rapidly becoming a serious performance bottleneck in ultra-deep submicron (UDSM) technologies. Carbon nanotube (CNT) based interconnects have been proposed as an alternative, because of their remarkable conductive, mechanical and thermal properties. In this paper, we investigate the system level performance of single-walled CNT (SWCNT) bundles, and mixed SWCNT/multi-walled CNT (MWCNT) bundles. Detailed RLC equivalent circuit models for conventional Cu and CNT bundle interconnects are described and used to determine propagation delays. These models are then incorporated into a system-level environment to estimate the impact of using CNT bundle global interconnects on the overall performance of several multi-core chip multiprocessor (CMP) applications. Our results indicate that the CNT bundle alternatives have a slight performance advantage over Cu global interconnects. With further improvements in CNT fabrication technology, we show how CNT bundle-based interconnects can significantly outperform Cu interconnects.

1. Introduction

As technology scales into the ultra-deep submicron (UDSM) region, interconnect design is becoming a major roadblock in realizing emerging multi-core chip multiprocessors (CMPs) that have tens to hundreds of components integrated on a single chip [1]. Interconnects used in CMPs can be classified into two categories: *local interconnects*, that are used for short distance communication, and have a delay of less than a clock cycle, and *global interconnects*, that are used for long distance communication to distribute data, clock, power supply and ground across the chip, and have a delay spanning multiple clock cycles [23]. According to the International Roadmap for Semiconductors (ITRS) 2005 [4], global interconnect performance has become one of the semiconductor industry's topmost challenges. Conventional copper (Cu) global interconnects have become increasingly susceptible to electromigration at high current densities ($>10^6$ A/cm^2) leading to considerable degradation in reliability [2]. Additionally, as interconnect dimensions are scaled down, rising crosstalk coupling noise and parasitic resistivity due to electron-surface and grain-boundary scatterings cause global interconnect delay to increase rapidly [3]. There is therefore a critical need to investigate innovative global interconnect alternatives to Cu.

Recently, there has been tremendous interest in carbon nanotubes (CNTs) as a possible replacement for Cu interconnects in future technologies [5]-[8]. Depending on the direction in which they are rolled (called *chirality*) CNTs can behave either as semiconductors or conductors. Conducting (or metallic) CNTs possess many extraordinary properties that make them promising candidates for implementing interconnects in UDSM technologies. Due to their covalently bonded structure, they are highly resistant to electromigration and other sources of physical breakdown [5]. They can support very high current densities with very little performance degradation. For instance, it was shown in [15] that the current carrying capacity of CNTs did not degrade even after 350 hours at current densities of ~10^{10} A/cm^2 at 250 ºC. CNTs possess high thermal conductivity in the range of 1700–3000 W/m·K [16]. They also have much better conductivity properties than Cu owing to longer electron mean free path (MFP) lengths in the micrometer range, compared to nanometer range MFP lengths for Cu [7].

CNTs can be broadly classified into single-walled carbon nanotubes (SWCNTs) [9]-[11] and multi-walled carbon nanotubes (MWCNTs) [12]-[13]. SWCNTs consist of a single sheet of graphene rolled into a cylindrical tube, with a diameter in the nanometer range. MWCNTs consist of two or more SWCNTs concentrically wrapped around each other, with diameters ranging from a few to several hundred nanometers. For on-chip global interconnects, bundles of SWCNTs and mixed SWCNT/MWCNTs (Fig. 1) are of special interest because of their superior conductivity properties. An SWCNT bundle consists of several SWCNTs packed together in parallel [12], whereas a mixed SWCNT/ MWCNT bundle consists of a combination of SWCNTs and MWCNTs packed together in parallel [13].

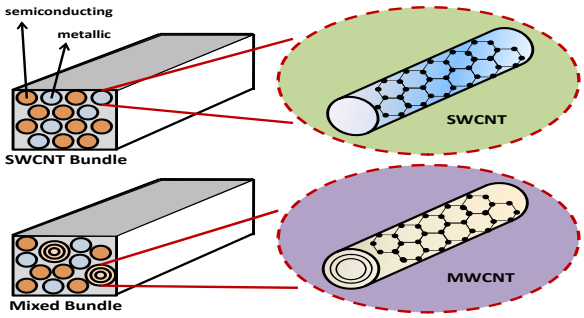


Fig. 1. Carbon nanotube (CNT) bundle interconnect alternatives: SWCNT bundle and mixed SWCNT/MWCNT bundle

The question arises: how do CNT bundles compare against Cu as global interconnect materials? While there has been a lot of interest in CNT-based interconnects in recent years, there are unfortunately few studies that compare CNT bundle interconnects with Cu interconnects at the system level. As complex chip multiprocessor (CMP) systems become the norm, it is critical to evaluate the impact of using promising new interconnect paradigms such as CNT bundles in such systems. Such an analysis allows designers to understand potential benefits and limitations of CNT bundle technology. It also gives them a true insight into realistic gains that can be achieved by switching to CNT bundle-based global interconnects in the future.

In this paper, we present such a comparative performance analysis of conventional Cu global interconnects with the CNT bundle global interconnect alternatives shown in Fig. 1. We investigate the performance impact of using CNT bundle global interconnects in place of conventional Cu global interconnects at the system-level for several CMP applications. Detailed RLC equivalent circuit models for Cu and CNT bundle interconnects are described and used to determine propagation delays. The CNT equivalent RLC circuit models capture the statistical distribution of metallic nanotubes while accurately incorporating recent experimental and theoretical results on inductance, crosstalk capacitance and ohmic resistance. These models are then incorporated into a system level environment to

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.84

estimate the impact of using CNT bundle-based global interconnects on the overall performance of several heterogeneous CMP applications with diverse data traffic profiles. Our results indicate that CNT bundle alternatives have a slight performance advantage over Cu global interconnects, providing up to a 1.5× speedup for the applications explored. With further improvements in CNT fabrication technology, we show how CNT bundle interconnects can significantly outperform Cu interconnects.

2. Related Work

In the last few years, there has been a lot of interest in studying the properties carbon nanotubes (CNTs). Researchers have developed RLC circuit models for CNT interconnect alternatives and compared their performance with Cu interconnects at the circuit level. RLC circuit models have been developed for isolated SWCNTs [9]-[11], SWCNT bundles [8][12][14] and MWCNTs [12]-[13]. Recently, conductance and inductance models for mixed SWCNT/MWCNT bundles were also introduced [13][40]. Other studies have presented some interesting discussions on the impact of process variations on CNT performance [18] and the possibility of CNTs replacing Cu interconnects in future FPGA fabrics [19]. However, none of these studies have analyzed the impact of using CNT interconnects instead of Cu interconnects at the system-level. Recently, [17] explored the possibility of using CNTs as global interconnect buses. The authors described a dual-walled CNT global bus and presented experiments to show that it outperforms SWCNT global buses. In this work, we explore the possibility of using CNT bundles as global interconnects. Unlike [17], we present comparisons with conventional Cu global interconnects, and investigate CNT bundle performance for several heterogeneous CMP applications.

3. SWCNT and MWCNT Circuit Models

In this section, we present an overview of SWCNT and MWCNT equivalent RLC interconnect circuit models derived from literature. These models are used as the foundation for the circuit models of the CNT bundle interconnects, described in Section 4.

3.1 SWCNT

The fundamental (quantum) resistance of an SWCNT can be determined using the Landauer-Buttker formula [5], $R_Q=h/4e^2$ and is about 6.45kΩ. This is the constant resistance of an SWCNT of length less than or equal to its mean free path length λ. For longer lengths, SWCNT resistance has been shown to depend on its length and bias voltage [11]. For bias voltage less than the critical bias (< 0.16V for global wires [22]), the SWCNT resistance is:

$$R_{SWCNT} = R = \frac{h}{4e^2}, \quad l \le \lambda$$
$$= \frac{h}{4e^2}\left(\frac{l}{\lambda}\right), \quad l > \lambda \tag{1}$$

where h is Planck's constant, e is the charge of an electron, l is the SWCNT length and λ is the mean free path (MFP) length. Ideally, the mean free path (λ) value is in the order of several μm. However, a rigorous analysis has shown that practically, λ is around 1 μm [10].

For SWCNT interconnects, three types of capacitance should be considered: an electrostatic capacitance with the ground (C_{EG}), a coupling capacitance with any adjacent SWCNTs (C_{EC}), and an intrinsic quantum capacitance (C_Q) [6]. The electrostatic capacitance per unit length between an SWCNT and the ground plane is:

$$C_{EG} = \frac{2\pi\epsilon}{cosh^{-1}\left(\frac{2H}{d_t}\right)} \tag{2}$$

where ε is the permittivity of the dielectric and H is the distance between the SWCNT and ground plane. Similarly, the electrostatic capacitance per unit length between two parallel SWCNTs is:

$$C_{EC} = \frac{\pi\epsilon}{cosh^{-1}\left(\frac{s}{d_t}\right)} \tag{3}$$

where s is the inter-SWCNT spacing and d_t is the SWCNT diameter. Finally, the SWCNT quantum capacitance per unit length is:

$$C_Q = \frac{2e^2}{hv_F} \tag{4}$$

SWCNTs have been shown to consist of four co-propagating quantum channels, and therefore the effective SWCNT quantum capacitance is $4C_Q$ [6]. As per the analysis in [24], $4C_Q$ is in series with a parallel combination of C_{EG} and C_{EC}. Because electrostatic coupling between non-adjacent SWCNTs is very weak compared to coupling between adjacent SWCNTs, the coupling capacitance between nonadjacent SWCNTs can be neglected in parallel global interconnects with more than two lines [17].

An SWCNT also has two types of inductance associated with it – kinetic and magnetic. The kinetic inductance (L_K) is due to charge carrier inertia, since electrons do not instantaneously react to an applied electric field. The series inductance which represents this phenomenon is [6]:

$$L_K = \frac{h}{2e^2v_F} \tag{5}$$

Since a nanotube has four co-propagating quantum channels, the effective value of kinetic inductance in the equivalent circuit is $L_K/4$ [17]. Kinetic inductance typically dominates magnetic inductance in SWCNTs [6].

3.2 MWCNT

An MWCNT consists of two or more concentric SWCNTs. Since the basic building blocks of MWCNTs are SWCNTs of varying diameters, many of the properties of SWCNTs hold for MWCNTs. The number of shells (N_s) in an MWCNT is diameter dependent:

$$N_s = 1 + \frac{D_{outer} - D_{inner}}{2\delta} \tag{6}$$

where $\delta=0.34nm$ (van der Waals distance) is the spacing between adjacent concentric shells, and D_{outer} and D_{inner} are the maximum and minimum shell diameters. The ratio of D_{outer}/D_{inner} has been observed to vary from 0.35 to 0.8 [12]. The approximate number of conduction channels per shell for an MWCNT is [13]:

$$N_{chan/shell}(d) \approx \begin{cases}(ad+b)P_m, & d > 6\ nm\\ 2P_m, & d < 6\ nm\end{cases} \tag{7}$$

where a = 0.1836 nm^{-1}, b = 1.275, d is the shell diameter and P_m = 1/3 (similar to an SWCNT bundle). Then the resistance for the ith SWCNT shell with diameter d_i is:

$$R_{SWCNTi}(d_i, l) = \frac{R_{SWCNT}}{N_{chan/shell}(d_i)} \tag{8}$$

Each shell has its own d_i, λ and $N_{chan/shell}$ that can be derived from D_{outer}. The total MWCNT resistance (R_{MWCNT}) is a parallel combination of the resistances of all the concentric SWCNTs:

$$(R_{MWCNT}(D_{outer}, l))^{-1} = \sum_{N_s}(R_{SWCNTi}(d_i, l))^{-1} \tag{9}$$

where R_{SWCNTi} is the resistance of the ith concentric SWCNT. An inter-shell resistance (R_i ≈10 kΩ/μm) must also be considered to account for the inter-shell tunnel transport phenomenon [17].

The metallic shell in an MWCNT constitutes an effective electrostatic shield for its inner shells [26]. Thus the capacitance between an internal shell and ground, and between non-adjacent shells can be safely neglected. The electrostatic capacitance per unit length between the outermost nanotube in a MWCNT and the ground is given by (2), where d_t is the diameter of the outermost shell. Similarly, the coupling capacitance between the outer shells of adjacent MWCNTs is given by (3) and the quantum capacitance is

given by (4). The electrostatic coupling capacitance between adjacent shells in a MWCNT (C_{ESC}) is derived from a conventional metallic coaxial configuration [27]:

$$C_{ESC} = \frac{2\pi\epsilon}{ln(d_1/d_2)} = \frac{2\pi\epsilon}{ln(d_1/(d_1 - 2\delta))} \quad (10)$$

The quantum capacitance ($4C_Q$) of an external shell is coupled with a parallel combination of the three capacitances C_{ESC}, C_{EG} and C_{EC} in the equivalent RLC circuit.

Finally, the total MWCNT inductance is given by a relation similar to that for an SWCNT bundle (8):

$$L_{MWCNT} = \left(\frac{L_k}{2.\sum_{N_s} N_{chan/shell}(d_i)} + L_m\right).l \quad (11)$$

where L_k is obtained from (5) and $N_{chan/shell}(d_i)$ from (7). L_m is calculated using the equivalent conductivity method [40].

4. CNT Bundle Circuit Models

In this section, we present the equivalent RLC circuit models for SWCNT bundle and mixed SWCNT/MWCNT bundle interconnects. These models are used to determine CNT bundle interconnect propagation delay in Section 5.

4.1 SWCNT Bundles

An SWCNT bundle consists of several individual SWCNTs in parallel. An important parameter associated with an SWCNT bundle is its metallic density (P_m) which refers to the probability that an SWCNT in the bundle is metallic (i.e., conducting). The value of P_m ≈ 1/3 [1] with today's best fabrication techniques, which implies that only 1/3 of the SWCNTs in a bundle are conducting. Fig. 2 shows a schematic of a SWCNT bundle interconnect, taking into account P_m of the bundle by considering the spacing x between SWCNTs (x=$d_t/\sqrt{P_m}$) [8]. The number of conducting SWCNTs in a bundle (n_{CNT}) of width w and height h is:

$$n_w = \frac{w - d_t}{x}, n_h = \frac{h - d_t}{(\sqrt{3}/2)x} + 1 \quad (12)$$

$$n_{CNT} = n_w n_h - \frac{n_h}{2} \quad if\ n_h\ is\ even$$

$$= n_w n_h - \frac{n_h - 1}{2} \quad if\ n_h\ is\ odd$$

where n_w and n_h are the number of SWCNTs in a row and along the height of the bundle, respectively.

The resistance of an SWCNT bundle is simply the parallel combination of n_{CNT} metallic SWCNTs [8]:

$$R_{SWCNT\ bundle} = \frac{R_{SWCNT}}{n_{CNT}} \quad (13)$$

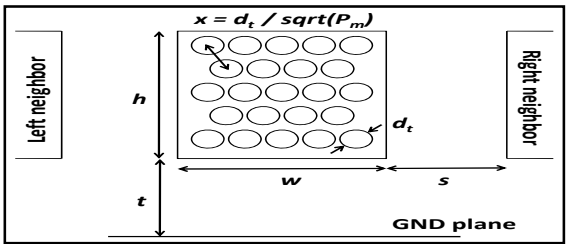


Fig. 2. Schematic of SWCNT bundle interconnect geometry

As far as SWCNT bundle capacitance is concerned, the electrostatic capacitance to the ground and coupling capacitances arise mainly from the SWCNTs lying at the edges of the SWCNT bundle. The coupling and electrostatic capacitances to the ground of SWCNT bundles have been analyzed extensively in [10] (using the field solver RAPHAEL) and [22] (using the 3D field solver FastCap), and found to be equal to the respective capacitances of a Cu wire with the same cross-sectional dimensions. The effective quantum capacitance of a SWCNT bundle is further reduced in a bundle and found to be negligible compared to its electrostatic counterparts [22].

The kinetic inductance of an SWCNT bundle is the parallel combination of individual SWCNT kinetic inductances. The magnetic inductance remains relatively constant with wire dimensions and cannot be ignored anymore, as was done in the case of individual SWCNTs. The mutual inductance between SWCNTs in a bundle is accounted for using the partial element equivalent circuit (PEEC) model [25]. The total SWCNT bundle inductance is:

$$L_{SWCNT\ bundle} = \left(\frac{L_k}{4n_{CNT}} + L_m\right).l \quad (14)$$

4.2 Mixed SWCNT/MWCNT Bundles

A mixed SWCNT/MWCNT bundle consists of SWCNTs with a diameter d and MWCNTs with various diameters $D_{inner} \le d_i \le D_{outer}$. It has been shown [2] that the outer diameters follow a normal (Gaussian) distribution.

The resistance for a mixed SWCNT/MWCNT bundle is [13]:

$$R_{mixed\ bundle} = \left(\int \frac{N(D_{outer})\,\partial D_{outer}}{R_{MWCNT}(D_{outer}, l)}\right)^{-1} \quad (15)$$

where $R_{MWCNT}(D_{outer}, l)$ is obtained from (9) and $N(D_{outer})$ is the tube count according to D_{outer}'s, with a normal (Gaussian) distribution, a mean diameter $\overline{D_{outer}}$, and a standard deviation $\sigma_{D_{outer}}$ [2]. For N_{bundle} CNTs in the bundle, the tube count for a given D_{outer} is:

$$N(D_{outer}) = \frac{N_{bundle}}{\sigma_{D_{outer}}\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{D_{outer} - \overline{D_{outer}}}{\sigma_{D_{outer}}}\right)^2\right] \quad (16)$$

This relation can be used to derive a distribution curve for the tube count. The resulting curve and the corresponding MWCNT resistance curve can be used to determine total resistance of the mixed bundle from (15). This resistance formulation has been validated with experimental results from [2] in [13].

The capacitive characteristics of a mixed SWCNT/MWCNT bundle have been shown to predominantly be determined by the cross sectional dimensions of a bundle [1]. Therefore, similar to the case of the SWCNT bundle, the mixed bundle electrostatic capacitance to ground and coupling capacitances is assumed to be the same as that of a Cu wire with identical cross-sectional dimensions. The quantum capacitance is similarly negligible compared to its electrostatic counterparts [22].

Finally, the total kinetic inductance of a mixed bundle is the parallel inductance value of all the conduction channels in the bundle, similar to (11) and (14). Magnetic inductance L_m is calculated using the equivalent conductivity method [40].

5. Experiments

We now present the results of several experiments that explore the impact of using CNT bundles as global interconnects. First we compare the interconnect propagation delays for CNT bundle and Cu interconnects. Subsequently, the performance of CNT bundle and Cu global interconnects is compared for several multi-core heterogeneous CMP applications at the system-level. Finally, we predict how CNT bundle global interconnects will perform with inevitable advances in fabrication technology, in coming years.

5.1 Propagation Delay Comparison

In our first experiment we compared the global wire delay of Cu and CNT bundle interconnect alternatives across the 45-22 nm UDSM process technology nodes. We use the equivalent circuit models described in the previous section to determine wire delay for CNTs. The equivalent circuit model for Cu wires is obtained from [28]-[30], and used to derive the wire delay for Cu. We consider optimal repeater sizing and insertion for both Cu and CNT wires,

using the formulations presented in [31]. The node driver resistance, load capacitance and process parameters were obtained from ITRS specifications [4]. Since global wire width is typically much larger than minimum wire width (W_{min}) to improve delay and bandwidth characteristics, we considered an aspect ratio of 1 and used a global wire width value of $5W_{min}$ which is shown to optimize the power-delay product for Cu wires [32]. This wire width is in the shallow RLC region, where the difference between RC and RLC model latencies is only 10% [10]. For comparison purposes, we used the same width and aspect ratios for the SWCNT bundles and mixed SWCNT/MWCNT bundles. For the mixed SWCNT/MWCNT bundle, we assumed $\overline{D_{outer}}$ = 4.2nm and $\sigma_{D_{outer}}$ = 1.25nm [2]. The SWCNT diameter was assumed to be 1nm, and the ratio between outer to inner diameters for MWCNTs in the bundle was assumed to be 0.5. The spacing between adjacent wires was assumed to be the same as the wire width ($5W_{min}$). The CNT mean free path $\lambda = 1\mu m$ and metallic density $P_m = 1/3$, while Cu mean free path was assumed to be 40 nm. All of these values are practically achievable today by using prevalent fabrication techniques.

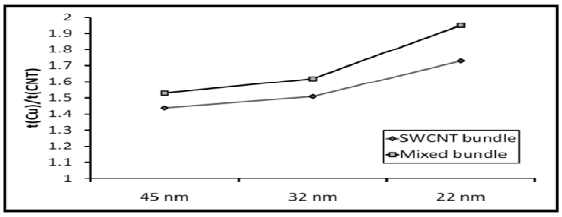


Fig. 3. Global interconnect delay comparison between Cu and CNT bundle alternatives (45-22 nm)

Fig. 3 shows the ratio of propagation delay of copper *t(Cu)* and the propagation delay of CNT bundles *t(CNT)*, for global wires (> 1mm in length) across the 45-22nm technology nodes. It can be seen that both the SWCNT bundle and mixed SWCNT/MWCNT bundle interconnects have lower propagation delays compared to Cu. The mixed SWCNT/MWCNT bundle interconnect has many large MWCNTs with several shells and more conduction channels than SWCNT bundles of the same dimensions. This is the reason why mixed SWCNT/MWCNT bundles perform better as conductors, compared to SWCNT bundles.

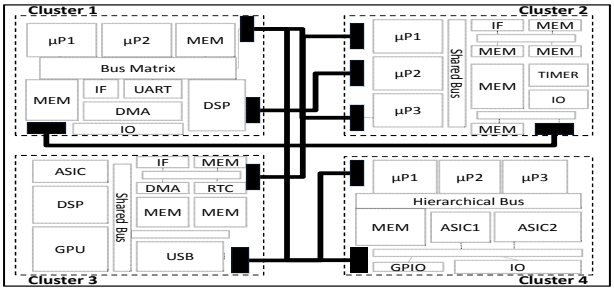


Fig. 4. A 4 cluster CMP layout, with global (pipelined) interconnects

In addition to the comparison of propagation delays between CNT bundle and Cu interconnects, for the sake of completeness, we also performed a comparison of the propagation delay for an isolated SWCNT global interconnect with a Cu global interconnect. The isolated SWCNT was found to have a much higher propagation delay (almost 100× more) compared to Cu. This large propagation delay is due to the very high SWCNT resistance, because of its extremely small cross-section area. We conclude from this result that isolated SWCNTs are unsuitable as global interconnects. They may however someday replace Cu at the local interconnect level because of their much lower lateral capacitance that improves latency for short distances, as some recent studies suggest [10].

5.2 System-level Performance Analysis

For our next experiment, we selected several multi-core CMP applications to analyze the overall performance impact of using CNT bundle-based global interconnects at the system-level. The applications are selected from the well known SPLASH-2 benchmark suite (*Barnes*, *Ocean*, *FFT*, *Radix*) [33], as well as from the networking domain (proprietary benchmarks *Netfilter*, *Datahub* and *SecurePck*). These applications are parallelized and implemented on multiple cores. The entire chip is assumed to be partitioned into clusters (or islands) of heterogeneous computation cores. Each cluster consists of tightly coupled cores (processors, memories etc.) optimized for dedicated tasks (e.g., packet encryption, image processing, etc.) and interconnected via local bus-based Cu links that support high data bandwidths. Fig. 4 shows an example layout of a four cluster multi-core CMP. Global pipelined links (shared bus or point-to-point) are used to connect computation clusters with each other, and facilitate inter-cluster data transfers. Two clusters can be interconnected by multiple global links if higher inter-cluster bandwidth needs to be supported. Table 1 summarizes the implementation details of the CMP applications, such as number of cores (including memories, peripherals, and processors), programmable processors, computation clusters and inter-cluster global links on the chip.

Table 1. CMP Application Implementation Characteristics

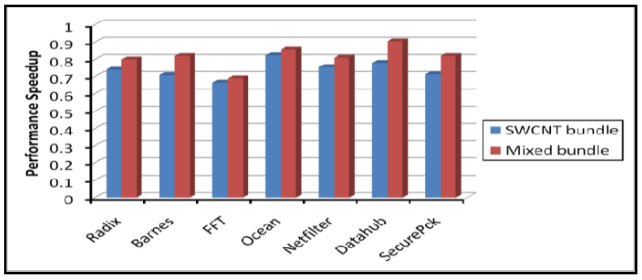
CMP applications	Description	cores	prog. processors	clusters	global links
Radix	Integer radix sort	18	4	3	10
Barnes	Evolution of galaxies	26	6	4	18
FFT	FFT kernel	28	6	4	12
Ocean	Ocean movements	35	10	5	24
Netfilter	Packet processing and forwarding	49	22	6	32
Datahub		68	26	8	34
SecurePck		94	30	8	28

The CMP applications were modeled in SystemC [35] using a fast and accurate transaction-based bus cycle accurate (T-BCA) modeling abstraction [33][34][36]. The cores in each cluster (e.g. processors, memories, peripherals) were modeled at the behavioral level granularity, while the communication at the inter- and intra-cluster level was modeled at a cycle accurate granularity. Each of the applications was simulated with testbench traffic to quickly (~few hours) and accurately estimate performance of the implementations. The various cores were interconnected using the AMBA AXI [37] standard interface protocol, with the address bus width set to 32 bits and the separate read and write data bus widths set to 64 bits.

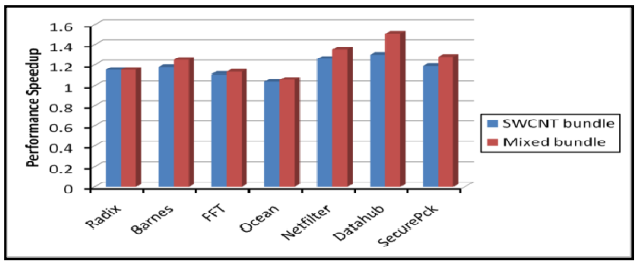
Our analysis was performed for the 22-nm process technology node (as predicted for the 2016 node of ITRS 2005 [4]) and the interconnect fabric was clocked at frequencies ranging from 1-10 GHz to support data transfer rates in the hundreds of Gbps range for future high performance systems. The die size was assumed to be 20mm×20mm. A high level simulated annealing floorplanner based on sequence pair representation (PARQUET [38]) was used to create an early layout of the CMP application on the die, and Manhattan distance based wire routing estimates were used to determine wire lengths. For the global interconnects, in addition to repeater insertion, latch insertion is performed based on wire length, wire delay and clock frequency of the bus, to pipeline the interconnect and ensure correct operation [39]. For instance, a global Cu wire of length 10 mm has a projected delay of 2 ns in the 22-nm technology node, for a $5W_{min}$ wire width. To support a frequency of 10 GHz (clock period of 0.1 ns), approximately 20 latches need to be inserted to ensure correct multi-cycle operation.

Fig. 5 (a) and (b) show the performance improvement (speedup in application execution time) when using SWCNT bundle and mixed

SWCNT/MWCNT bundle global interconnects, instead of Cu global interconnects, for the 22-nm technology node, with the interconnect fabric clocked at 10 GHz. Experiments with the interconnect fabric clocked at lower frequencies within the 1-10 GHz range yielded speedup results within 5% of the results shown in the figures (assuming computation core frequencies are scaled down by the same factor as the interconnect fabric), and hence the other results are not presented for brevity.



(a)



(b)

Fig. 5. System level performance speedup when using CNT bundles instead of Cu for global interconnects, assuming (a) imperfect metal-CNT contacts, (b) perfect metal-CNT contacts

Fig. 5 (a) shows the results for the case when imperfect metal-CNT contacts are assumed. A high metal-CNT contact resistance of 100 KΩ is added to account for the excessive electron scattering at imperfect metal-CNT junctions. It can be seen that the application performance with CNT bundle global interconnects is worse than with Cu global interconnects. This is primarily an artifact of using poor metal-CNT contacts, typically constructed using Gold, Palladium or Rhodium. Many recent studies [20]-[21] with state-of-the-art fabrication techniques have however managed to reduce this contact resistance down to a very small value (a few hundred Ω). This has dramatically improved the viability of successful integration of CNTs with CMOS technology.

Fig. 5 (b) shows the application performance improvement under the assumption of perfect contacts with negligible contact resistance. The applications now perform better with CNT bundle global interconnects than with Cu global interconnects because of the lower CNT bundle interconnect delays. The lower CNT bundle delays lead to more widely spaced and hence fewer pipeline latches on global inter-cluster interconnects, resulting in global data transfers taking fewer clock cycles and improving application performance. Among the applications, *Ocean* can be seen to have a much smaller speedup compared to other applications. This is because *Ocean* has fewer inter-cluster data transfers, which reduces the advantage of having faster CNT bundle-based global interconnects. On the other hand, the speedup for *Radix* and *FFT* is lower than other applications because of the smaller length of their global interconnects, which again reduces the impact of faster CNT bundle-based global interconnects on application performance. Overall, the results indicate that SWCNT bundle global interconnects provide a speedup of up to 1.3×, while mixed bundles achieve speedups of up to 1.5×.

At first glance, these modest performance improvements over Cu interconnects may appear to be not so significant as to justify a migration to CNT bundle interconnects. However in the next section we present results that indicate more significant CNT bundle performance gains with inevitable improvements in CNT fabrication technology.

5.3 Impact of MFP and CNT Metallic Density

Our system-level experiments above assumed practical values for CNT metallic density (P_m=1/3) and mean free path (λ=1μm) while calculating application performance gains. The modest CNT bundle speedup obtained above can be improved if breakthroughs in fabrication technology in the future allow for greater metallic density (P_m) and longer MFP lengths (λ) for CNT bundles.

Fig. 6 shows the interconnect delay speedup over Cu for SWCNT bundles and mixed SWCNT/MWCNT bundles, with increasing values of metallic density P_m, ranging from 0.3 (practical) to 1 (ideal), for a constant MFP length $\lambda=1\mu m$. It can be seen that there is a crossover point beyond which SWCNT bundle performance improves upon mixed SWCNT/MWCNT bundle performance. This is because as the metallic density increases, the mixed bundle has fewer metallic nanotubes than the SWCNT bundle due to large MWCNTs which prevent a tight packing. The SWCNT bundle has more tightly packed and consequently more metallic tubes which improve conductivity and performance. Fig. 7 shows a similar trend when bandwidth density of CNT bundle interconnects is compared with Cu, for varying metallic densities. It is clear from the figure that SWCNT bundle bandwidth density is superior to that of Cu for metallic densities greater than 0.3. SWCNT bundle bandwidth density also improves upon mixed SWCNT/MWCNT bundle bandwidth density for higher metallic densities, due to more tightly packed metallic tubes.

Fig. 8 and 9 show the combined performance improvement for various configurations of metallic density (P_m) and MFP (λ) lengths for the SWCNT bundle and mixed SWCNT/MWCNT bundle interconnects, respectively. It is clear from the results that increasing MFP length and metallic densities can lead to a substantial improvement in delay (and hence performance) speedup for CNT bundle interconnects – as much as 4.9× for SWCNT bundles and up to 4.4× for mixed SWCNT/MWCNT bundles. Improving CNT fabrication technology in order to achieve such performance gains is an area of tremendous research activity today [41]-[42].

6. Conclusion and Future Work

In this paper, we presented a comprehensive comparative analysis of the performance impact of using CNT bundle-based global interconnects over Cu for heterogeneous multi-core CMP applications. Our experimental results indicate that SWCNT bundles and mixed SWCNT/MWCNT bundles lead to performance gains over Cu global buses of up to 1.3× and 1.5× respectively. These gains can be further improved if the CNT mean free path (MFP) lengths and metallic densities are increased with advances in fabrication technology that are actively being explored today. Ultimately, while many manufacturing and technological factors will contribute to the realization of CNT global interconnects, our preliminary results indicate that SWCNT bundle and mixed SWCNT/MWCNT bundle based global interconnects have the properties to be viable replacements for Cu global interconnects in future high performance CMP applications, as process technology scales. Our future work will consider the impact of using CNTs as local interconnects and analyze trade-offs between CNT interconnect power, performance and bandwidth density.

Acknowledgements

This research was partially supported by grants from SRC (2005-HJ-1330 and 1617.001) and NSF (CCF-0702797).

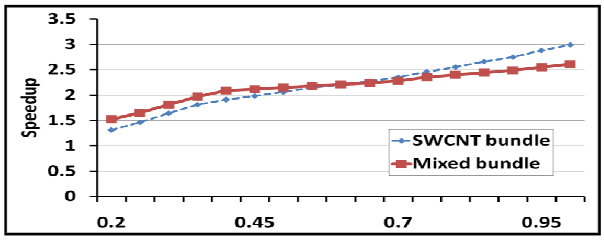


Fig. 6. SWCNT bundle and mixed bundle delay speedup over Cu for varying metallic densities P_m (x-axis)

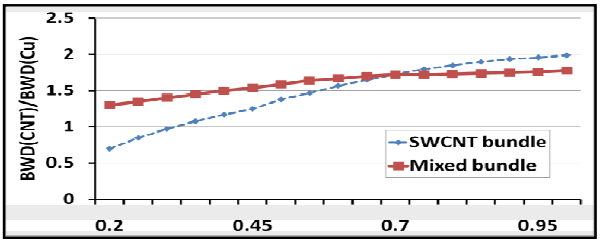


Fig. 7. SWCNT bundle and mixed bundle bandwidth density speedup over Cu for varying metallic densities P_m (x-axis)

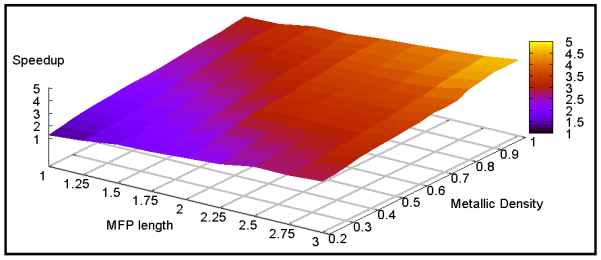


Fig. 8. SWCNT bundle speedup over Cu for different combinations of metallic density and MFP

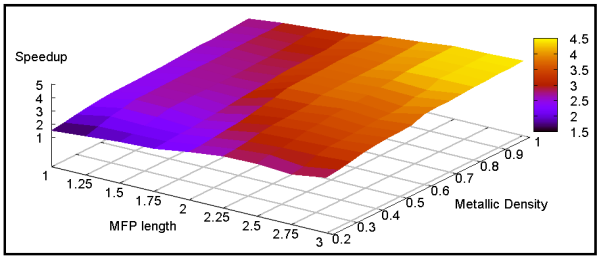


Fig. 9. Mixed SWCNT/MWCNT bundle speedup over Cu for different combinations of metallic density and MFP

References

[1] A. Naeemi, et al., "On-Chip Interconnect Networks at the End of the Roadmap: Limits and Nanotechnology Opportunities", IITC 2006.

[2] J. Li, et al., "Bottom-up approach for carbon nanotube interconnects," Appl. Phys. Lett., Apr. 2003.

[3] S. M. Rossnagel, T. S. Kuan, "Alteration of Cu Conductivity in the Size Effect Regime," JVST, Jan. 2004.

[4] ITRS, International Technology Roadmap for Semiconductors, 2005.

[5] M. S. Dresselhaus, et al, "Carbon nanotubes: synthesis, structure, properties, and applications", Springer, 2001.

[6] P. J. Burke, "Luttinger liquid theory as a model of the gigahertz electrical properties of carbon nanotubes," Trans. NANO, Sep. 2002.

[7] F. Kreupl, et al., "Carbon nanotubes for interconnect applications," IEDM Tech. Dig., Dec. 2004.

[8] N. Srivastava, K. Banerjee, "Performance Analysis of Carbon Nanotube Interconnects for VLSI Applications," ICCAD 2005.

[9] A. Raychowdhury, K. Roy, "A circuit model for carbon nanotube interconnects: comparative study with Cu interconnects for scaled technologies", ICCAD 2004.

[10] A. Naeemi et.al, "Design and Performance Modeling for Single-Walled Carbon Nanotubes as Local, Semiglobal, and Global Interconnects in Gigascale Integrated Systems", Trans EDL Jan 2007.

[11] A. Raychowdhury, K. Roy, "Modeling of Metallic Carbon Nanotube Interconnects for Circuit Simulation and a Comparison with Cu Interconnects for Scaled Technologies," TCAD Jan 2006.

[12] A. Naeemi et al., "Performance Modeling and Optimization for Single- and Multi-Wall Carbon Nanotube Interconnects," DAC 2007

[13] S. Haruehanroengra, W. Wang, "Analyzing Conductance of Mixed Carbon-Nanotube Bundles for Interconnect Applications", IEEE EDL Aug 2007.

[14] A. Nieuwoudt, et al., "Predicting the Performance and Reliability of Carbon Nanotube Bundles for On-Chip Interconnect," ASPDAC, 2007.

[15] B. Q. Wei, R. Vajtai, P. M. Ajayan, "Reliability and current carrying capacity of carbon nanotubes," Appl. Phys. Lett., 2001.

[16] J. Hone, et al, "Thermal conductivity of single-walled carbon nanotubes," Phys. Rev., B, vol. 59, no. 4, 1999.

[17] D. Rossi, et al., "Modeling Crosstalk Effects in CNT Bus Architectures", IEEE Trans. NANO Mar 2007

[18] A. Nieuwoudt, Y. Massoud, "Assessing the Implications of Process Variations on Future Carbon Nanotube Bundle Interconnect Solutions," ISQED, 2007.

[19] S. Eachempati, et al., "Assessing Carbon Nanotube Bundle Interconnect for Future FPGA Architectures," DATE, 2007.

[20] S. Sato et al., "Novel approach to fabricate carbon nanotube via interconnects using size-controlled catalyst nanoparticles," IITC 2006.

[21] O. Hjortstam, et al., "Can we achieve ultra-low resistivity in carbon nanotube-based metal composites?" App. Phy MSP 2004.

[22] H. Cho, et al., "Modeling of the performance of carbon nanotube bundle, cu/low-k and optical on-chip global interconnects", SLIP 2007.

[23] S. Pasricha, and N. Dutt. "On-Chip Communication Architectures", Morgan Kauffman, Apr 2008.

[24] P. J. Burke, "Quantitative theory of nanowire and nanotube antenna performance," ArXiv Cond. Matter E-Prints, Aug. 2004.

[25] M. W. Beattie, L. T. Pileggi, "Inductance 101: Modeling and Extraction," DAC, 2001.

[26] P. G. Collins, P. Avouris, "Multishell conduction in multiwalled carbon nanotubes," App. Phy. MSP, Mar. 2002.

[27] E. R. Dobbs, Basic Electromagnetism. Chapman, Hall 1993.

[28] A. F. Mayadas, M. Shatzkes, "Electrical-Resistivity Model for Polycrystalline Films: the Case of Arbitrary Reflection at External Surfaces," Phys. Review B, vol. 1, 1970.

[29] E. H. Sondheimer, "The mean free path of electrons in metals," Adv. Physics, vol. 1, no. 1, 1952.

[30] C. P. Yue, S. S. Wong, "Physical Modeling of Spiral Inductors on Silicon," Trans. EDL, 2000.

[31] Y. I. Ismail et al., "Effects of Inductance on the Propagation Delay and Repeater Insertion in VLSI Circuits" TVLSI Apr 2000.

[33] S.C. Woo et al."The SPLASH-2 programs: Characterization and methodological considerations", ISCAS, 1995.

[32] M. A. El-Moursy et al, "Optimum Wire Sizing of RLC Interconnect with Repeaters," GLSVLSI, 2003.

[33] S. Pasricha, "Transaction Level Modeling of SoC with SystemC 2.0" In Proc. SNUG 2002.

[34] S. Pasricha, N. Dutt, M. Ben-Romdhane, "Extending the Transaction Level Modeling Approach for Fast Communication Architecture Exploration", In Proc. DAC 2004.

[35] SystemC initiative. www.systemc.org

[36] W. Müller, J. Ruf,, W. Rosenstiel, "SystemC Methodologies and Applications", Norwell, MA: Kluwer, 2003

[37] AMBA AXI Specification www.arm.com/armtech/AXI

[38] S. N. Adya, I. L. Markov, "Fixed-outline Floorplanning: Enabling Hierarchical Design", IEEE Trans TVLSI, Dec. 2003

[39] V. Nookala, S. S. Sapatnekar, "Designing optimized pipelined global interconnects: Algorithms and methodology impact," ISCAS, 2005.

[40] W. Wang, et al. "Inductance of mixed carbon nanotube bundles," Micro & Nano Letters, IET , vol.2, no.2, pp.35-39, June 2007

[41] Liu et al., "Densification of Carbon Nanotube Bundles for Interconnect Application", IITC 2007

[42] Zhu et al., "Assembling Carbon Nanotube Bundles Using Transfer Process for Fine-Pitch Electrical Interconnect Applications", EETC 2007

2009 22nd International Conference on VLSI Design

Impact of Bias Voltage on Magnetic Inductance of Carbon Nanotube Interconnects

K. C. Narasimhamurthy and Roy P. Paily

VLSI Design Laboratory, Dept. of ECE
Indian Institute of Technology Guwahati, Guwahati-781039, Assam, India
e-mail: kcn, roypaily@iitg.ernet.in

Abstract

Single-walled carbon nanotube (SWCNT) bundles have the potential to provide an attractive solution for the resistivity and electromigration problems faced by traditional copper interconnects. This paper discusses the impact of bias voltage variation on magnetic inductance of SWCNT bundle. The variation of bias voltage on inductance was ignored so far. The authors utilize existing models for SWCNT bundle for evaluation. There is a significant variation in inductance value within the available range of bias voltage. This study shows that the inductance change with respect to bias voltages is about 1% to 35% at different lengths of SWCNTs.

1. Introduction

The modeling, design and implementation of on-chip interconnects continue to be a fundamental roadblock to realizing high-performance integrated systems. The International Technology Roadmap for Semiconductors (ITRS) predicts that traditional interconnects will be a major performance and reliability bottleneck when feature sizes become smaller [1]. SWCNT is a graphene roll with a diameter of 0.5 nm to a few nanometers and depending on its chirality, they can be either metallic or semiconductor. SWCNT is very close to a one-dimensional (1-D) system of electrons that gives rise to many unique electrical and thermal properties.

Typically, inductance is seen as a parasitic component in metal interconnects. In analog VLSI Integrated Circuit (IC) design, inductor is one of the major components. Passive inductors using the SWCNT bundles are reported [2] to have high quality factor and high inductance compared to metal inductors. Since the radius of carbon nanotube is several nanometers, the magnetic field (H) induced by the current in carbon nanotube is about one thousand times larger than that induced by the current in normal copper wire whose radius is about several micrometers. In this paper, we have investigated the loop inductance of ground-signal-ground configuration (GSG) of SWCNT bundle and its dependency on the bias voltage. To the best of our knowledge there has been no work addressing the inductance variation with respect to bias voltage. SWCNTs will be incorporated in future ICs and the voltage levels across the SWCNTs will be varying in nature. This study is relevant to the cases in which inductance is considered as parasitic as well as a passive device element. In both cases, a change in inductances of SWCNTs will definitely affect the circuit functionality and its performance. An assessment of the nature and quantity of inductance variation will be pertinent. The rest of the paper is organized in the following manner. Section 2, summarizes the modeling of SWCNT resistance which is essential to appreciate the variation of inductance of SWCNT bundles. In Section 3, we study the modeling of inductance of SWCNT. Section 4, discusses the effect of bias voltage on inductance. In Section 5, simulation results of GSG configurations are discussed. Section 6, concludes the paper.

2. Modeling of Resistance of an isolated SWCNT

Intrinsic resistance, Contact resistance and Ohmic resistance are the 3 different resistances associated with an SWCNT and are discussed below.

2.1. Intrinsic Resistance R_i

The conductance of a carbon nanotube is evaluated using the two-terminal Landauer-Buttiker formula. This formula states that, for a 1-D system with N channels in parallel, the conductance $G = (Ne^2/h)T$, where T is the transmission coefficient for electrons through the sample, h is Planck's

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.21

constant and e is the charge of a single electron. Due to spin degeneracy and sub lattice degeneracy of electrons in graphene, each nanotube has four conducting channels in parallel (N=4). Hence the conductance of a single ballistic SWCNT assuming perfect contacts (T=1), is given by $(4e^2/h) = 155\,\mu s$ which results in a resistance of 6.45 KΩ. This is the intrinsic resistance associated with an SWCNT that cannot be avoided [3]. In other words, R_i can be expressed as

$$R_i = \frac{h}{4e^2} \tag{1}$$

2.2. Contact Resistance R_c

A contact resistance R_c model is due to imperfect metal contacts. As nanotube fabrication and bonding techniques have improved, the additional resistance due to imperfect metal contacts has been significantly reduced and in several experimental cases has approached zero ohm [4]. In this paper we assume R_c = 0.

2.3. Ohmic Resistance R_o

The ohmic resistance (R_o) of an SWCNT is defined as

$$R_o = \frac{h l_b}{4e^2 \lambda_{ap}} \tag{2}$$

where l_b is the length of the nanotube, and λ_{ap} is the mean free path for acoustic phonon scattering [3]. However, resistance of an individual SWCNT depends on the applied bias voltage. For nanotubes operating in the low bias regime ($V_b < 0.1$ Volt), the resistance is the summation of the lumped intrinsic resistance (R_i) and contact (R_c) resistances and distributed per unit length ohmic resistance (R_o).

$$R_{low} = R_i + R_c \quad if \;\; l_b < \lambda_{ap} \tag{3}$$

$$R_{low} = R_o + R_c \quad if \;\; l_b > \lambda_{ap} \tag{4}$$

For high bias voltages ($V_b < 0.1$ Volt), the resistance of an individual nanotube depends on the applied bias voltage.

$$R_{high} = R_{low} + \frac{V_b}{I_o} \tag{5}$$

where I_o is the maximum current that can flow through an individual nanotube, which is approximately 20-25 μA.

3. Modeling Inductance in SWCNT bundles

There are two kinds of inductances associated with SWCNT and are discussed separately below.

3.1. Kinetic Inductance l_k

Inductance has conventionally been defined as the resistance to current change due to Faraday's law, and it represents the energy stored in the magnetic field generated by current$(1/2)LI^2$. Electric current itself is the flow of carriers that have nonzero mass and therefore non-zero kinetic energy [5]. Assuming constant current density in a conductor, the kinetic inductance becomes equal to

$$l_k = \frac{h}{2 v_f \, e^2} \tag{6}$$

Fermi velocity for grapheme and carbon nanotubes is usually taken as $v_f = 8 * 10^5\, m/s$. The overall kinetic inductance per unit length of a nanotube is 4 nH/μm.

3.2. Magnetic inductance

The magnetic inductance captures the impact of the voltage induced by the magnetic fields produced by time varying currents, which is encapsulated in Ampere's and Faraday's laws. Unlike resistance, capacitance and kinetic inductance are per unit length quantities at frequencies where magnetoquasistatic assumptions are valid. The magnetic inductance is dependent on the entire current loop, which typically consists of a signal line and its associated ground return paths. Since the distribution of the current in the loop may not be known apriori, the concept of partial inductance is used to model [6] the magnetic inductance.

Partial inductance is a mathematical construct that assumes that the current in a particular conductor (in the case of partial self-inductance) or the current flowing in adjacent conductors (in the case of partial mutual inductance) has a current return path at infinity. The partial inductance construct has no physical meaning by itself. However, when the partial self and mutual inductances are combined in a particular manner over an entire current loop, the total loop inductance is enhanced.

The partial self-inductance (L_m) of a single nanotube and the mutual inductance (M_m) between two parallel current carrying nanotubes of the length l and diameter d can be calculated using equation (7) and (8) respectively [7].

$$L_m = 200 l_b(\ln(l_b/d) + 0.5 + (2d/3l_b)) \tag{7}$$

$$M_m = 200 l_b \left(\ln\left(r + \sqrt{1+r^2}\right) - \sqrt{1 + \frac{1}{r^2}} + \frac{1}{r} \right) \tag{8}$$

where $r = l_b/s$ and s is the center-to-center spacing between the two nanotubes. To model the magnetic inductance of SWCNT bundles we utilize equivalent width model [8], in which the discrete SWCNTs in the bundle are replaced by a smaller number of conductors with same resistivity as that of ohmic resistivity of an individual nanotube

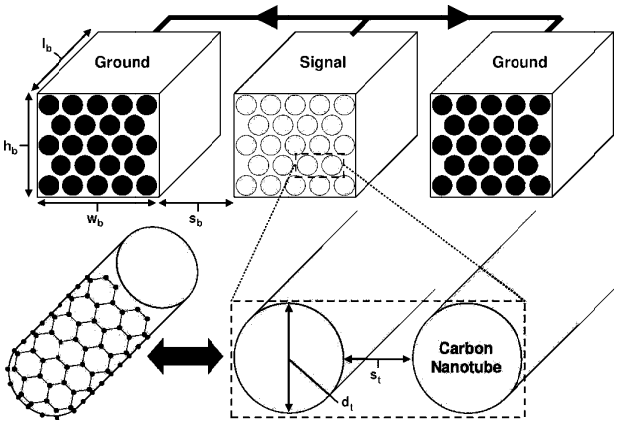


Figure 1. System of Single-walled Carbon Nanotube Interconnect bundles Implementing a signal line and two adjacent ground return paths(GSG) [3].

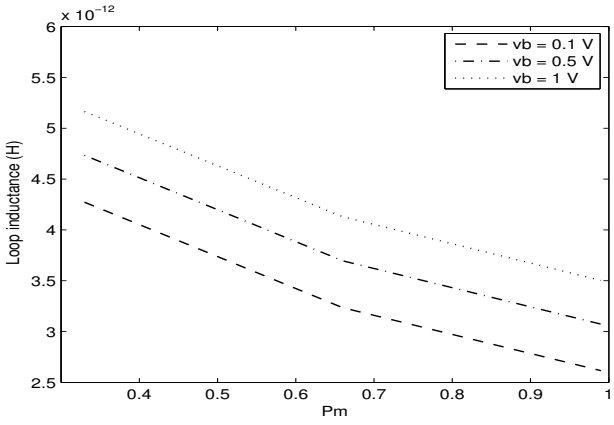


Figure 2. Loop inductance at different bias voltages for a GSG configuration of length $10\mu m$, width=height=$10\,nm$ and separation $1\,nm$

ρ_t. Using this model magnetic inductance of SWCNT can be effectively extracted. 00

00000000

4. Effect of bias voltage on inductance of SWCNT

At high frequencies, current return path is to minimize loop inductance [9]. In this paper we consider closest current return path, a signal line is sandwiched between two ground return paths as in Figure 1. This configuration is referred to as ground-signal-ground (GSG) model. It is also possible to have more return paths.

ITRS prediction of supply voltage is 1 Volt for next generation node. At this high bias voltage ($V_b > 0.1$ Volt), the resistance of SWCNT is a function of bias voltage as in equation (5). The SWCNT bundle resistance is also a function of bundle length as in equation (2). So at high bias, current nonlinearly varies with supply voltage. This property of SWCNT bundle is being exploited to see the change in loop inductance value of GSG configuration and this is the motivation behind this paper. In conventional copper interconnects current linearly varies with bias voltage.

To model the magnetic inductance of GSG configuration, each SWCNT bundle is represented by an equivalent conductor with resistivity same as that of an individual SWCNT at low bias. At different high bias voltages the width and height of the equivalent conductor is adjusted to equate the resistance of equivalent conductor to that of the SWCNT bundle, as in equivalent width model [8]. Loop inductance of conductor representing the SWCNT bundle [8] is

$$L_{loop} = I^T L_{mat} I \tag{9}$$

$$L_{mat} = \begin{bmatrix} L_{m1} & M_{m12} & \cdots & M_{m1n} \\ M_{m21} & L_{m2} & & M_{m2n} \\ \vdots & & \ddots & \\ M_{mn1} & M_{mn2} & & L_{mn} \end{bmatrix} \tag{10}$$

where I is a vector (10) with normalized current in each nanotube and L_{mat} is partial inductance matrix which is constructed from the partial self and mutual inductances for all n SWCNTs in the GSG interconnect configuration.

The size of L_{mat} is proportional to the number of nanotubes in the bundle. For low complexity system of SWCNT bundles, equation (10) can be used directly to calculate the loop magnetic inductance. However to analyze the loop inductance for wide range of geometries we utilize a version of multipole-accelerated field solver, FastHenry, which uses a partial element equivalent circuit (PEEC) formulation similar to equation (10).

5. Analysis of simulation result

Loop inductance of GSG configuration is obtained using FastHenry, by using resistivity of conductor same as that of SWCNT resistivity. Width and height of conductor at different bias voltage (0.1V to 1V) is calculated by equating resistance of the bundle and equivalent conductor, as in equivalent width model [8]. We simulated about 1000 GSG configuration of SWCNT bundles with the following geometric

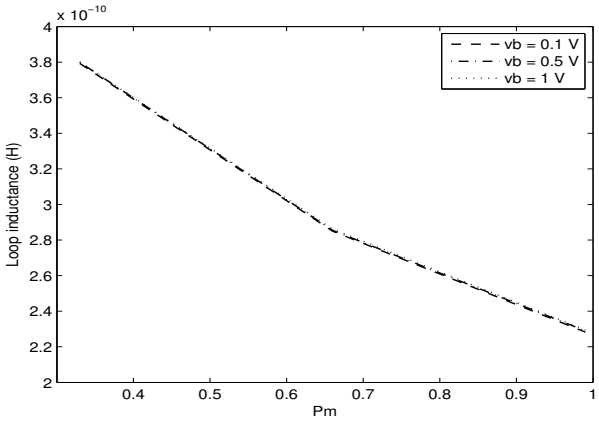


Figure 3. Loop inductance at different bias voltages for a GSG configuration of length $1000\mu m$, width=height=$10\,nm$ and separation $1\,nm$

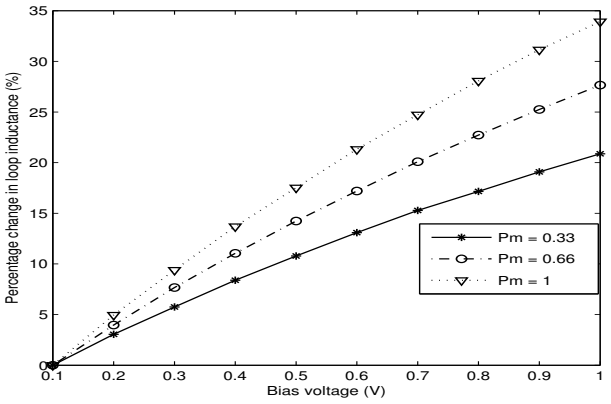


Figure 4. Percentage change in loop inductance of a GSG configuration of length $10\mu m$, width=height=$10\,nm$ and separation $1\,nm$ for different voltages at different P_m

parameters: $10\,\mu m \leq l_b \leq 1000\,\mu m$,$10d \leq w_b \leq 20d$ and $0.1w_b \leq S_b \leq 10w_b$ where, w_b is the bundle width, S_b is edge to edge spacing between bundles and d diameter (d=1 nm). Probability P_m that a given SWCNT in the bundle is metallic is varied from 0.33 to 1. For all simulations, we have taken uniform descritization and conductor is represented by 4 filaments.

From Figure 2 and Figure 3 it is observed that the loop inductance of the GSG configuration will vary more for shorter lines compared to longer lines with the bias voltage. For a given length of line, change in loop inductance across the bias voltage is independent of P_m. The value of loop inductance is proportional to the length of the line as in equations (7-9).

However, the variation in loop inductance is 45% for a length of 10 μm when the P_m changes from 0.33 to 1 for a given bias voltage, but it is 70% for a length of 1000 μm. That means, if the line is a part of analog tuned circuit, then change in P_m of SWCNT bundle during its growth may affect considerably the tuning of the circuit. This shows that there should be a good control over P_m in the growth of SWCNT, to use them in analog circuit applications. However it has been reported that, there will be no change in the capacitance of the SWCNT bundle if the number of metallic CNTs in the bundle change [10].

While analyzing the simulation results of Figure 4 and Figure 5 for the line with $l_b = 10\,\mu m$ and $w_b = h_b = 10\,nm$ it is found that 34% variation in percentage change in loop inductance across V_b for $P_m = 1$, $S_b = 1\,nm$ =1nm and it is 21% when $P_m = 0.33$ and the same is just 9.5% and 8% respectively for $S_b = 100\,nm$. This trend continues for any width and height($w_b = h_b$) of the line. This shows that as the separation increases the variation of loop inductance with respect to bias voltage reduces. The above analysis is extended to different lengths and we obtained the graph in Figure 6.

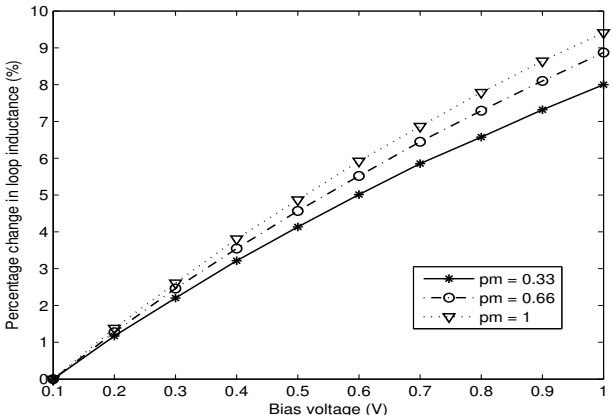


Figure 5. Percentage change in loop inductance of a GSG configuration of length $10\,\mu m$, width=height=$10\,\mu m$ and separation $100\,nm$ for different voltages at different P_m

Figure 6 shows the variation of change in loop inductance for different length and different across the bias voltage. The loop inductance variation is less than 1% for longer lines. However, it is 21% 14% and 8% for S_b of 1 nm, 10 nm, 100 nm respectively. Therefore loop inductance variation at different bias voltage is critical for inter-

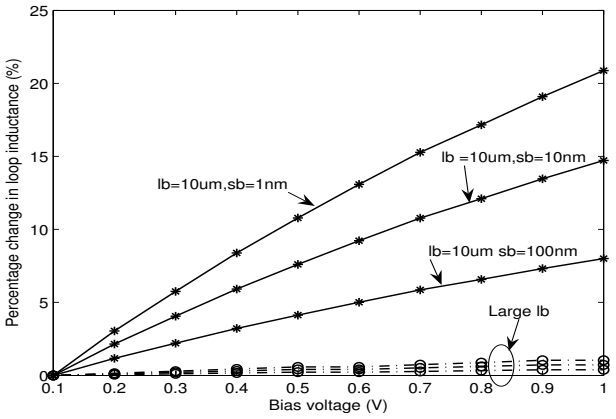


Figure 6. Change in loop inductance as length increases for $P_m = 0.33$ with respect to bias voltage for different S_b

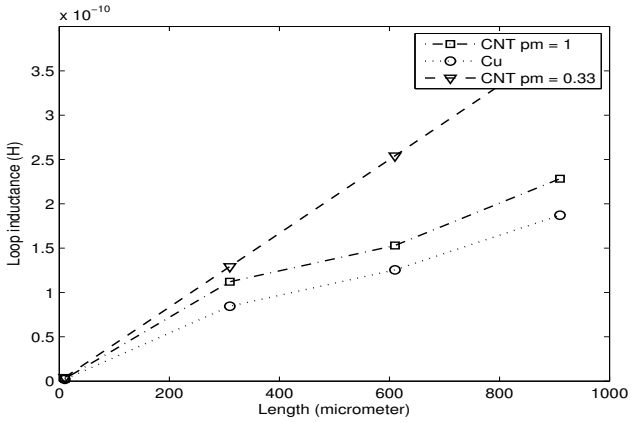


Figure 7. Comparison of loop inductance of GSG configuration with copper and SWCNT bundles for different length at different P_m

mediate interconnects. For local interconnects because of high intrinsic resistance of SWCNT, Cu is preferred.

The loop inductance of GSG configurations of copper (Cu) are obtained using FastHenry. While calculating the resistivity of Cu, surface scattering and scattering due to grain boundary are considered. The comparison of the loop inductance of SWCNT with that of Cu is shown in Figure 7.

Figure 7 shows a large deviation in loop inductance between the Cu and SWCNT at longer length, this is due to lower value P_m which results in high resistance of SWCNT, leading to representation of equivalent conductor by lesser dimension. However for P_m = 1, deviation in inductance values are less because the dimension equivalent conductor are almost same as of Cu. Even though loop inductance of SWCNT is more, at nm dimensions SWCNT is preferred over Cu, because Cu resistance is very large compared to SWCNT.

6. Conclusions

We have briefly discussed the modeling of CNT. The impact of bias voltage on magnetic inductance of SWCNT bundle was evaluated using the existing SWCNT models. This study showes that the inductance change with respect to bias voltages is about 35% to less than 1% at different lengths. The highest change in inductance was for shorter lengths and high P_m. SWCNT is a promising candidate to replace Cu interconnects for semi global and global interconnects [9] for improving delay, power and bandwidth. But selective and lateral growth is a challenge. There are reports of selective production of long, horizontally aligned carbon nanotubes ($> 1\,mm$) [11]. It is anticipated that with improved processing technology SWCNT will replace existing Cu interconnects and the presented study is very relevant in such cases.

References

[1] International technology roadmap for semiconductors. Technical report, 2005. ch. Interconnect.

[2] Artur Nieuwoudt and Yehia Massoud. Predicting the performance of low-loss on-chip inductors realized using carbon nanotubes. *IEEE Trans. Electron devices*, 55(1):298–312, Jan. 2008.

[3] Artur Nieuwoudt and Yehia Massoud. Evaluating the impact of carbon nanotube bundles for vlsi interecon-nect using diameter dependent technique. *IEEE Trans. Electron Devices*, 53(10):517–520, Oct 2006.

[4] P.L. McEuen and J.-Y. park. Electron transport in single walled carbon nanotubes. *Mater.Res.Soc.Bull.*, 29(4):272–275, Apr. 2004.

[5] J.P. Bruke. Luttinger liquid theory as a model of gigahertz electrical properties of carbon nanotubes. *IEEE Trans.Nanotech.*, 1(5):129–144, Sept. 2002.

[6] A.E. Ruehli. Inductance calculation in a complex intergated circuit environment. *IBM J. Res. Develop.*, pages 470–481, Sept. 1972.

[7] F.Grover. *Inductance Calculations: Working Formulas and Tables*. Dover, New York, 1962.

[8] Artur Nieuwoudt and Yehia Massoud. Understanding the impact of inductance in carbon nanotube bundles for vlsi interconnect using scalable modeling techniques. *IEEE Trans. Nanotech.*, 5(6):758–765, Nov. 2006.

[9] So Young Kim, Yehia Massoud, and S. Simon Wong. On the accuracy of return path for loop inductance extraction for 0.1 um technology and beyond. pages 401–404. ISQED, 2003.

[10] Aszad Naeemi, Reza Savari, and James D. Meilndl. Design and performance modelling and optimization for single walled carbon nanotubes as local, semiglobal and global interconnects in gigascale integrated systems. *IEEE Trans.Electron Devices*, 54(1):26–37, Jan 2007.

[11] A. Reina, M. Hofmann, D. Zhu, and J. Kong. *Growth mechanisms of Horizantally aligned Carbon Nanotubes*. CVD MTL annual research report, 2007.

Conservative QCA Gate (CQCA) for Designing Concurrently Testable Molecular QCA Circuits

Himanshu Thapliyal and Nagarajan Ranganathan
Department of Computer Science and Engineering,
University of South Florida, Tampa, FL, USA
E-mail : {hthapliy,ranganat}@cse.usf.edu

Abstract

Nanocircuits based on molecular QCA are prone to high error rates. In this paper, we present a novel conservative logic gate termed 'CQCA' (conservative QCA) to design concurrently testable circuits for molecular QCA. In conservative logic gates, there would be an equal number of 1s in the output as there would be on the input. Thus, conservative logic gates are parity preserving, that is, the parity of the input vectors is equal to the output vectors. CQCA is proposed in this work as molecular QCA is based on majority voting. We analyzed the fault patterns in existing popular conservative Fredkin gate and proposed CQCA gate due to single missing/additional cell defect in molecular QCA. We found that if there is a fault in molecular QCA implementation of Fredkin and CQCA gates, there is a parity mismatch between the input and the output; otherwise the input parity is same as output parity. Thus, any permanent and transient fault in molecular QCA can be concurrently detected if implemented with conservative Fredkin and CQCA gates. We applied novel method of using majority and minority voting to detect the fault in conservative gates. We propose to use CQCA gate compared to existing popular Fredkin gate as CQCA excels Fredkin gate in parameters of complexity(number of majority voter), speed and area. The results are well supported by synthesizing standard benchmark combinational functions. The QCA design of 2 pair 2 rail checker is also presented for the first time ever in literature. The design of QCA layouts and the verification of the designs are performed using the QCADesigner and HDLQ tools.

1. Introduction

The existing CMOS technology is reaching its limits beyond which the down scaling in feature size and proper working of the device is becoming extremely difficult. CMOS devices suffer from heat generation as they have to discharge all the stored energy when flipping from 1 to 0. Quantum dot cellular automata (QCA) is one of the emerging nanotechnologies in which it is possible to achieve circuit densities and clock frequencies much beyond the limit of existing CMOS technology. QCA has significant advantage in terms of power dissipation as it does not have to dissipate all its signal energy hence considered as one of the promising technologies to achieve the thermodynamic limit of computation [1-2]. The basic QCA logic devices comprise the majority voter (MV), the inverter (INV), binary wire and the inverter chain. Figure 1 shows the basic QCA cell and logic devices.

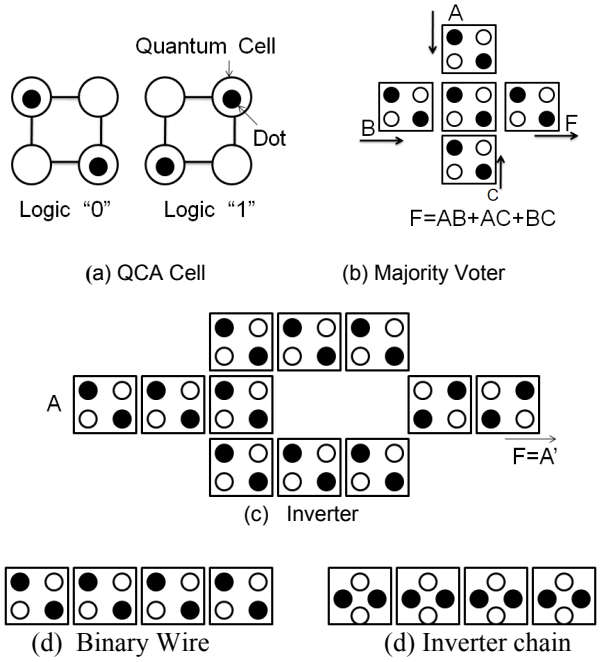


Figure 1. Basic QCA Devices

The Launder four phase clocking scheme is generally used in QCA design, and the present work is also based on this. Due to significant error rates in nano-scale manufacturing, nanotechnologies including

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.75

QCA require extremely low device error rate [5].In manufacturing QCA, defects can occur in the synthesis and deposition phases. However, defects are more likely to take place during the deposition phase [6]. QCA devices are also prone to transient faults caused by thermodynamic effects, radiation and other effects, as the energy difference between the ground and the excited state is small [10,11]. Thus, in literature researchers have used the novel concepts such as reversible logic to improve the testability of molecular QCA [5].

In this work, we explore the concept of conservative logic gates as a means for designing concurrently testable circuits for molecular QCA. Conservative logic gates have equal number of 1s in the output as there would be on the input [15]. Thus, they are parity preserving, that is, the parity of the input is always equal to the parity of the output. As molecular QCA is based on majority voting, the design based on conservative logic will be completely different from conventional CMOS designs. In literature, conservative Fredkin gate is the most popular gate but we find that Fredkin gate is not suitable for all molecular QCA designs as the designs based on it requires more clocking zones, majority gates and area. Since QCA logic is based on majority voting, this led us to propose CQCA gate (conservative QCA gate). CQCA requires only two clocking zones compared to four clocking zones required by the Fredkin gate and it requires only two majority gates compared to six majority gates required by the Fredkin gate. The benefits of using CQCA over Fredkin gate in terms of area and delay is shown by synthesizing standard combinational benchmark functions. To demonstrate the effectiveness of conservative Fredkin and CQCA gates for concurrently testable molecular QCA design, we have done the fault pattern study of conservative Fredkin and CQCA gates due to single missing/additional cell defect in QCA. We found that when there is permanent fault due to above defects, there is parity mismatch between the input and the output of the conservative Fredkin and CQCA gates. Due to parity preserving property, any permanent and transient fault in molecular QCA can be concurrently detected. We demonstrated a novel strategy of using majority and minority voter gates to detect parity mismatch between the input and output of the 3 input 3 output conservative gates instead of XOR gates, as it is costly to implement XOR function in QCA compared to majority/minority voter. The QCA design of 2 pair 2 rail checker is also presented for the first time ever in literature. Thus, this work lays the foundation of concurrent testing of molecular QCA which is susceptible to high error rates.

The paper is organized as follows: Section 2 presents the QCA defects and related work; Section 3 presents existing and proposed conservative gates. Section 4 presents concurrent testing of molecular QCA using conservative logic gates; Section 5 shows the QCA design of 2 pair 2 rail checker; Section 6 presents the comparison between Fredkin and CQCA conservative gates; Section 7 provides the conclusions.

2. Background and related work

In manufacturing QCA, defects can occur in the synthesis and deposition phases. However, defects are more likely to take place during the deposition phase [6]. Researchers assume that QCA cells have no manufacturing defects and in metal QCA faults occur due to cell misplacement. These defects can be characterized as cell displacement, cell misalignment and cell omission [7]. Researchers have proved that molecular QCA cells are more susceptible to missing/additional QCA cell defects [8,17]. Additional cell defect is due to the deposition of an additional cell on the substrate while missing cell defect is due to loss of a particular cell

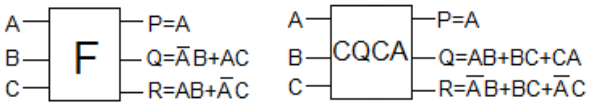
2.1. Related work

The testing of QCA is first time addressed in a seminal work in [6]. In [6], the defect characterization of QCA devices is investigated and is shown how the testing of QCA differs from conventional CMOS. In [8], the modeling of QCA defects at molecular level is done for combinational circuits. Fault characterization is done for single missing/ additional cell defect on different QCA devices such as MV, INV, fan-out, Crosswire and L-shape wire. In [7], test generation framework for QCA is presented. It is seen that additional test vectors can be generated for detecting QCA defects which remain undetected by stuck-at fault model. Bridging fault on QCA wires is also addressed. In [5], reversible logic is used to detect single missing/additional cell defects. It is seen that reversible 1D array is C-testable. In [14], fault-tolerant QCA designs are presented using triple modular redundancy with shifted operands. The strategy is proposed considering the wire delay and faults in wires in QCA.

3. Conservative gates for QCA

There is an existing popular conservative gate called Fredkin gate [15]. Fredkin gate is shown in Fig.2.a. Fredkin gate can be described as mapping (A, B, C)

to (P=A, Q=A'B+AC, R=AB+A'C), where A, B, C are input and P, Q, R are output, respectively. Fredkin gate produces the same number of 1s in the output as on the input. The QCA design of Fredkin gate is shown in Fig. 3 using four-phase clocking scheme, in which the clocking zone is shown by the number next to D (D0 means clock 0 zone, D1 means clock 1 zone and so on, MV in the figure represents majority voter). Thus, it can be seen that Fredkin gate has two level majority voter (MV) implementation and it requires 6 MVs to implement it.



(a). Fredkin gate (b). Proposed CQCA gate

Figure 2. Conservative QCA gates

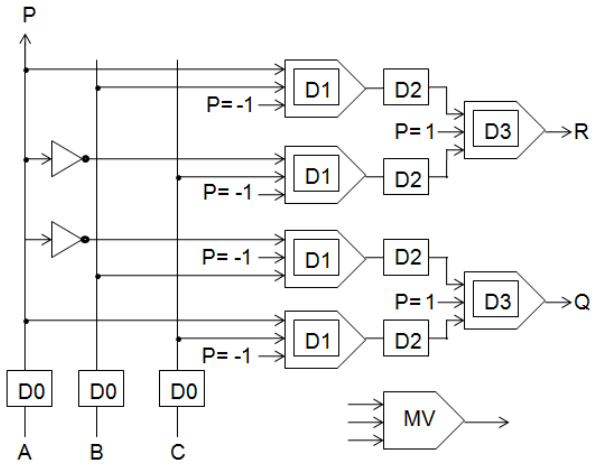


Figure 3. Fredkin gate QCA design (D0 to D3 represent clock zones 0 to 3)

Table 1. Truth table of CQCA gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	1	1	1

3.1. Proposed CQCA gate

The existing conservative Fredkin gate is costly in molecular QCA implementation. This led us to propose novel conservative gate especially suiting molecular QCA MV (majority voter) based design and is termed CQCA (Conservative QCA gate). The input to output mapping of CQCA is: P=A; Q=AB+BC+AC [MV(A,B,C)]; R=A'B+A'C+BC [MV(A',B,C)], where A, B, C are input and P, Q, R are output, respectively. Figure 2.b shows the block diagram representation of CQCA gate. Table 1 shows the truth table of the CQCA gate. It shows that it has the same number of 1's in the input as in the output. Figure 4 shows the QCA implementation of CQCA gate. It is seen that the CQCA can be implemented with one level MV logic and requires only two MVs to implement it. A detailed comparison between CQCA and Fredkin gate is presented in Section 6.

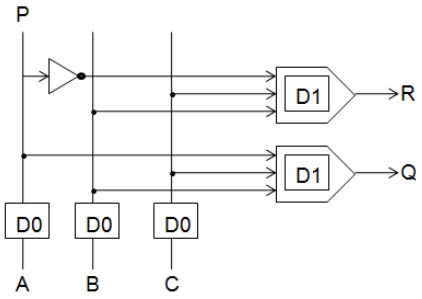


Figure 4. QCA implementation of CQCA gate

4. Concurrent testing of molecular QCA with conservative gates

To the best of our knowledge and as far as existing literature is concerned, concurrent testing for molecular QCA designs has never been addressed and the proposed work is the first attempt in this direction. In this work, our analysis is also based on missing/additional QCA cell defects. Figure 5 shows the QCA layout of the CQCA gate (the QCA layout of the Fredkin gate is described in [12]). We have modeled the Fredkin and CQCA gates QCA layouts, with the presence of all possible single missing/additional cell defects in MV, INV, fan-out, Crosswire and L-shape wire [8]. The modeling is done using HDLQ [16], a design tool which provides the Verilog HDL library of QCA devices, i.e., MV, INV, fan-out, Crosswire, L-shape wire with fault injection capability. The design is simulated in Verilog HDL simulator in the presence of faults to determine the corresponding output.

The exhaustive testing of the Fredkin and CQCA gates with 8 input patterns and all possible single missing/additional cell defects is done using the Active HDL simulator. The exhaustive testing for Fredkin gate generated 20 unique fault patterns. The

exhaustive testing of CQCA generated 8 unique fault patterns as shown in Table 2. In the fault patterns study in Table 2, ai is the 3 bit pattern having an equivalent decimal value of i, for example a0 represents 000 (decimal 0) and a7 represents 111(decimal 7).We carefully observe each fault pattern and found that in the occurrence of a fault, there is a parity mismatch between the output and the input of the Fredkin and CQCA gates (i.e., parity of the input vector is not equal to the output vector). This led us to conclude that Fredkin and CQCA gates can detect concurrently permanent fault by matching the parity. Since Fredkin and CQCA gates are logically parity preserving, they can detect also the transient faults. Hence, Fredkin and CQCA gates can concurrently detect permanent as well as transient fault based on parity preserving in molecular QCA. In CMOS circuits, parity match is checked as A⊕B⊕C=P⊕Q⊕R.. However, implementing the XOR gate in QCA is costly as the process requires 3 majority gates. In QCA, implementing A⊕B⊕C would require 6 majority gates and similarly P⊕Q⊕R would require 6 majority gates. Thus, comparing A⊕B⊕C=P⊕Q⊕R would require a total of 12 majority gates.

In order to check the parity mismatch we propose an alternative strategy to use majority voter for input vector and minority voter for output vector. Let D=MV(A,B,C) where D is the output of the majority gate and A,B,C are the input of the CQCA gate. Let S=mV(P,Q,R) where S is the output of the minority voter and P,Q,R are the output of the CQCA gate. Thus when there is no fault D will be complementary to S, and when there is a fault D will be same as S. The minority voter required can be designed by complementing the majority voter or as a novel design proposed in [13]. The proposed approach requires only 2 majority gates to compare the input and output of conservative CQCA gate. An example of the proposed approach is demonstrated for CQCA gate in Fig. 6 (The strategy is also applicable for Fredkin gate). The CQCA gate along with the majority and minority voter will be referred to as conservative testable block (CTB) in this paper. The reason for generating S and D as complementary in case of fault free condition is to make use of 2 pair 2 rail checker in comparing them. It can be argued that majority and minority voter used for generating D and S may have faults leading to incorrect results. Hence fault-tolerant majority gate are required, one of its design is described in [3]. The correct outputs D and S can be also be generated by using triple modular redundancy approach (TMR) for majority and minority voting [14]. The design of 2 pair 2 rail checker and its use to detect the fault is discussed in the next section.

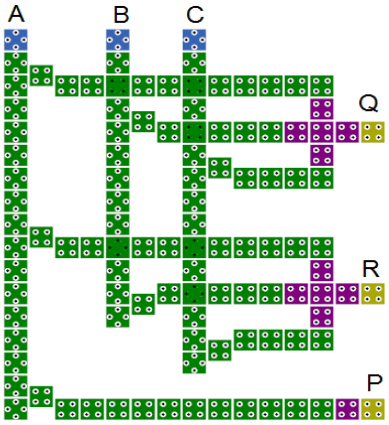


Figure 5. QCA layout of CQCA gate

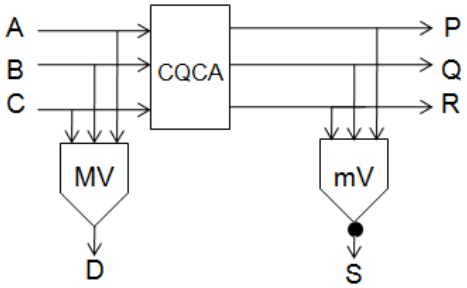


Figure 6. Conservative testable block

Table 2. Fault patterns in CQCA Gate

Input Vector	Fault Free	Fault Patterns							
		1	2	3	4	5	6	7	8
a0	a0	a0	a0	a0	a0	a0	a2	a0	a0
a1	a1	a3	a0	a3	a1	a1	a1	a0	a0
a2	a1	a3	a0	a1	a3	a3	a3	a1	a1
a3	a3	a3	a3	a1	a1	a3	a3	a3	a2
a4	a4	a4	a4	a6	a6	a4	a4	a4	a5
a5	a6	a4	a7	a6	a4	a4	a4	a6	a6
a6	a6	a4	a7	a4	a6	a6	a6	a7	a7
a7	a7	a7	a7	a7	a7	a7	a5	a7	a7

5. 2 pair 2 rail checker

The 2 pair 2 rail checker is required for testing that the output D and S generated by the majority and minority voting, respectively, are complementary or not. The error checking functions required in the 2 pair rail checker are E1=X0Y1+Y0X1 and E2= X0X1+Y0Y1; where X0/Y0 & X1/Y1 are complementary.

The 2 pair 2 rail checker produces the complementary output at E1 & E2 if the input passed to it are complementary. If the input are not complementary, the output E1 & E2 will be identical. We are also presenting a design of 2 pair 2 rail checker

based on MV QCA gate in Fig. 7. To the best of our knowledge, this is the first ever reported QCA design of 2 pair 2 rail checker. This design features 6 MV gates. Figure 8 shows the testing of conservative testable blocks (CTB) as described in Section 4 with 2 pair 2 rail checker. The output D and S (D1 and S1 in Fig. 8) of one testable block will be the input X0 and Y0 of the 2 pair rail checker. The other testable block output, D and S (D2 and S2 in Fig. 8) will form the other input X1 and Y1. Thus, 2 pair 2 rail checker can check 2 testable blocks at a time. The cascading of the 2 pair 2 rail checkers is done in a tree fashion, so that only the final 2 output are externally observable.

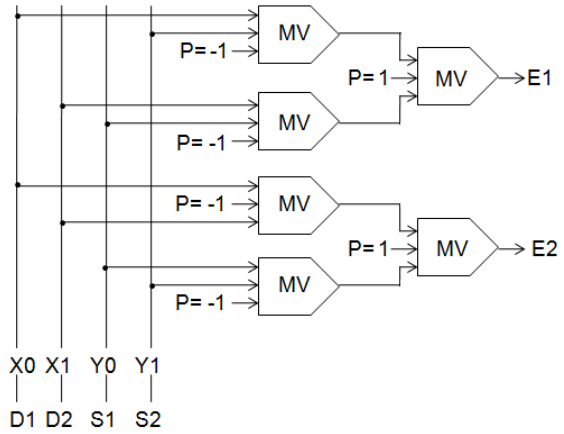


Figure 7. QCA design of 2 pair 2 rail checker (MV represents majority voter)

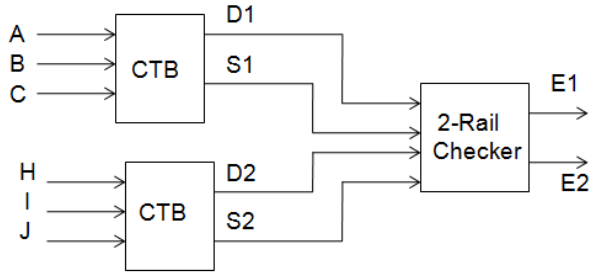


Figure 8. CTB testing using 2 pair 2 rail checker (2 CTBs can be tested with 1 checker)

6. Comparison of Fredkin and CQCA gate

Table 3 shows the comparison between the Fredkin and CQCA gates. In Table 3, since the number of clocking zones required to design CQCA conservative gate is less, it will be faster compared to Fredkin gate. The total number of QCA cell required in CQCA gate is only 47% of cells required by Fredkin gate and the area occupied by CQCA is only 29% of the area occupied by Fredkin gate (Fredkin gate requires 246 QCA cell with the area of 0.37 um^2 while CQCA requires 117 QCA cell with the area of 0.11um^2). Thus, the proposed CQCA gate excels Fredkin gate in all aspects.

6.1. Simulations for verification

The designs were verified using QCADesigner ver. 2.0.3 [9]. In the bistable approximation, we used the following parameters: cell size=18 nm, number of samples=182800, convergence tolerance=0.001000, radius of effect=41 nm, relative permittivity= 12.9, clock high=9.8e-22, clock low=3.8e-23, clock amplitude factor=2.000, layer separation=11.5000nm, maximum iterations per sample=1000.

Table 3. A comparison of Fredkin and CQCA

	Fredkin	CQCA
Clk Zs	4	2
MVs	6	2
Total Cells	246	117
Area	0.37um^2 , where L=0.4812um and W= 0.76984um	0.11um^2 , where L=0.30012 um W= 0.36454um

6.2. Comparison on benchmark functions

In order to have to have the comparison of the Fredkin and proposed CQCA gate for logic synthesis, we have implemented thirteen standard three variable Boolean combinational functions proposed in [4] for molecular QCA. These thirteen functions cover all the 256 Boolean functions for three variables. Table 4 shows the comparison between the two by synthesizing these 13 standard functions. It requires a total of 246 MVs and 136 clock zones to implement the standard functions using Fredkin gate. While it requires only 86 MVs and 62 clock zones when these standard functions are implemented with proposed CQCA gate. Thus implementing with CQCA achieves a reduction of 65% and 54.4% in terms of MVs and clock zones, respectively, which shows that CQCA gate performs better than Fredkin gate in terms of speed and area.

7. Conclusions

We propose the use of conservative logic gates to design concurrently testable circuits for molecular QCA. CQCA gate as presented shows that it is better than most popular Fredkin gate in terms of area and speed. The results are supported by synthesizing standard benchmark functions.

Table 4. Synthesis comparison of thirteen standard functions

	Standard Function	Fredkin Implementation			CQCA Implementation		
		# of Fredkin	# of MVs	Clk Zs	# of CQCA	# of MVs	Clk Zs
1	F=ABC	2	12	8	2	4	4
2	F=AB	1	6	4	1	2	2
3	F=ABC+AB'C'	3	18	12	3	6	4
4	F=ABC+A'B'C'	4	24	12	6	12	8
5	F=AB+BC	2	12	8	2	4	4
6	F=AB+A'B'C	5	30	16	5	10	8
7	F=ABC+A'BC'+AB'C'	6	36	16	6	12	6
8	F=A	1	6	4	1	2	2
9	F=AB+BC+AC	5	30	16	1	2	2
10	F=AB+B'C	1	6	4	3	6	4
11	F=AB+BC+A'B'C'	6	36	16	6	12	8
12	F=AB+A'B'	2	12	8	4	8	6
13	F=ABC+A'B'C+AB'C'+A'BC'	3	18	12	3	6	4
	Total	41	246	136	43	86	62

A novel strategy of majority and minority voting mismatch is proposed to detect fault in the input and output of the conservative gates for molecular QCA. The design of 2 pair 2 rail checker is presented for molecular QCA for the first time ever time in literature. In conclusion, the proposed CQCA gate is of great importance to fault susceptible molecular QCA nano-computing.

8. References

[1] A. O. Orlov, I. Amlani, G. H. Bernstein, C. S. Lent, and G. L. Snider, "Realization of a functional cell for quantum-dot cellular automata," Science,vol. 277, pp. 928–930, 1997.

[2] P. Tougaw and C. Lent, "Logical devices implemented using quantum cellular automata," J. Appl. Phys., vol. 75, no. 3, pp. 1818–1825, 1994.

[3] A. Fijany, B.N. Toomarian, "New Design for Quantum Dots Cellular Automata to Obtain Fault Tolerant Logic Gates", Journal of Nanoparticle Research, vol. 3, pp. 27-37, Feb. 2001.

[4] R. Zhang, K. Walus, W. Wang, and G. A. Jullien, "A method of majority logic reduction for quantum cellular automata," IEEE Trans. Nanotechnol., vol. 3, no. 4, pp. 443--450, Dec. 2004.

[5] X. Ma, J. Huang, C. Metra, F.Lombardi, "Reversible Gates and Testability of One Dimensional Arrays of Molecular QCA", Springer Journal of Electronic Testing, Vol.24, No. 1-3, pp. 297-311,June, 2008.

[6] M. B. Tahoori, J. Huang, M. Momenzadeh, and F. Lombardi, "Testing of quantum cellular automata," IEEE Trans. Nanotechnol., vol. 3, no.4, pp. 432–442, Dec. 2004.

[7] P.Gupta, N.K. Jha, L.Lingappan, "A Test Generation Framework for Quantum Cellular Automata Circuits", IEEE Trans. VLSI Syst., Vol. 15, no.1,pp. 24-36, Jan 2007.

[8] M. Momenzadeh, M. Ottavi, F. Lombardi, "Modeling QCA defects at molecular level in combinational circuits", Proc. DFT in VLSI Systems, Monterey, CA, USA, 3-5 Oct 2005, pp. 208–216.

[9] http://www.qcadesigner.ca/

[10] T. Tanamoto et al.,"Quantum effect device", United States Patent 5710436, Jan 1998.

[11] R. K. Kummamuru et al., "Operation of a Quantum-Dot Cellular Automata (QCA),Shift Registers and Analysis of Errors", IEEE TRANS. Electron Devices, Vol. 50, No. 9, pp 1906-1913, Sep. 2003

[12] H. Thapliyal and N. Ranganathan, "Testable Reversible Latches for Molecular QCA", Proc. IEEE NANO 2008, Arlington,TX, Aug 2008, pp. 699-702. (invited paper after peer review process)

[13] S. Roy, B.Saha and B.K Sikdar,"Design and Characterization of Minority Gate as a Universal Logic for Quantum-Dot Cellular Automata", Journal of Computational and Theoretical Nanoscience, Vol.3, No.5, pp. 684-695,Oct 2006.

[14] T. Wei,K. Wu, R.Karri and A. Orailoglu, "Fault tolerant quantum cellular array (QCA) design using Triple Modular Redundancy with shifted operands",Proc. ASPDAC 2005, Shanghai, China, Jan 2005, pp.1192-1195

[15] E. Fredkin, T Toffoli, "Conservative Logic", International Journal of Theor. Physics, 21(1982),pp.219-253.

[16] M. Ottavi, L. Schiano, and F. Lombardi, "HDLQ: A HDL Environment for QCA Design", ACM Journal on Emerging Technologies in Computing Systems, Vol. 2, No. 4, pp. 243–261, Oct. 2006.

[17] M. Momenzadeh, J. Huang and F. Lombardi, "Defect Characterization and Tolerance of QCA Sequential Devices and Circuits," Proc. Defect and Fault Tolerance in VLSI Systems, CA, Oct 2005. pp. 199-207.

Session 8B

Timing Analysis and Optimization

An Approach to Measure the Performance Impact of Dynamic Voltage Fluctuations Using Static Timing Analysis

Vishweshwara R, vishwa@ti.com
Texas Instruments India

Venkatraman R, rvenkat@ti.com
Texas Instruments India

Udayakumar H, uday@ti.com
Texas Instruments India

Arvind N V, aravind@ti.com
Texas Instruments India

Abstract

Design closure for predictable silicon performance is emerging as the most challenging digital VLSI design problem in advanced deep-submicron technology nodes. One of the significant problems is effective power-grid distribution, and the comprehension of the impact of voltage drops in the power grid on design timing and performance. This paper proposes a way by which the complex interactions between timing and dynamic power drops can be comprehended without being significantly pessimistic, while also not losing out on accuracy. We highlight the heuristics that we have used in this regard to reduce the complexity of the timing analysis, and to reduce the overall computation time. The overall method uses conventional analysis approaches for dynamic voltage-drop and timing. This method proposes options for comprehending effects of dynamic voltage drops during traditional design-closure methods and also highlights means of validating any assumptions made. Comparison results between performance degradation due to voltage drop assumptions and the traditional margin based approaches show significant reduction in the pessimism and these are presented in this paper.

1 Introduction

Effective power-grid design has been one of the key challenges for high-performance digital designs. One of the key aspects of importance in this context is the measurement of the impact of the power grid on design performance. While a poorly designed power grid can cause a significant degradation of design performance, it is usually difficult to assess the impact of the power-grid quality on timing.

In this paper we look at options to analyse the impact of time-varying voltages on design performance using conventional power-grid analysis and timing-analysis methods. The next section presents the motivation behind this work in detail, where we also highlight work already published in this area. Section-III presents the options we have pursued to quantify the performance impact of time-varying voltages in the power-grid. We present the results of our work on real designs in the subsequent section. We conclude with future work that we could extend this work to.

2 Motivation

The main goal of this work is to understand and quantify the impact of time-varying voltages in the power-grid on the design performance. Traditional approaches have been to look at average (static) power-drop on the grid and to model the impact of this drop on the voltage by suitably scaling the delays of the cells in the design. While analysing for the dynamic voltage fluctuations on the grid has also been a standard design practice, one of the significant gaps in design approaches has been the inability to define a target for the voltage drop that would guarantee the performance targets of a design. One of the options for this has been detailed SPICE simulations to understand the overall performance implications, but this is usually resource intensive and quite impractical for very large designs.

There have been a few publications in the literature on this topic of voltage-aware timing analysis. [1] proposes a path-based solution, but this does not address the problem of the analysis being computationally intensive when applied to a full system-on-chip (SoC) design. Another approach for timing analysis comprehending voltage drops is proposed in [2]. This method uses iterative analysis to improve accuracy, and also requires characterisation of the cell library for higher voltage drops, while for lower values of the drop, the analysis uses a linear scaling of the delays with voltage drops. An alternative approach is presented in [3] where determining the impact of voltage fluctuations on delay is formulated as a constrained non-linear optimisation problem where the currents drawn from the supply are the optimisation variables. This method depends on gate-level

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.45

simulation data to help generate the information of currents in the logic blocks. This approach needs changes to the characterisation data and mandates multiple static timing analyses to ensure convergence. Also, this method helps in computing the delay changes for a given voltage drop target, and does not necessarily ensure that the actual voltages seen are comprehending in the timing analysis. We propose a frame-work where conventional static timing analysis solutions can be used to analyse a timing path in the context of the instance-based time-varying voltages computed from a power-grid analysis

3 Timing Analysis

To analyse a digital design represented as a netlist of standard cells and conventional macros, we start with the assumption that the voltage variations over time are available - these are usually the result of switching scenario based design analysis for voltage drops [4]. We also assume that given a voltage value, static timing analysis capabilities exist to compute the delays of cells at the voltages specified [5]. Given these assumptions, the problem of analysing for timing impact of voltage variations is reduced to a problem of identifying an appropriate voltage value for each instance in a design at which the timing impact can be assessed, and the accuracy of such an approach is directly dependent on the accuracy of the voltages thus used. The outline of this methodology is depicted in Fig.1

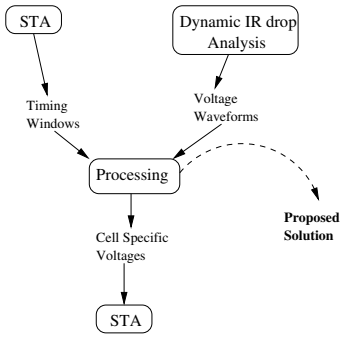


Fig.1 Methodology outline

The proposed methodology involves path based timing analysis. Some graph based timing analysis approaches for simultaneous analysis of multiple paths, with some amount of pessimism, using the graph based timing analysis are also presented.

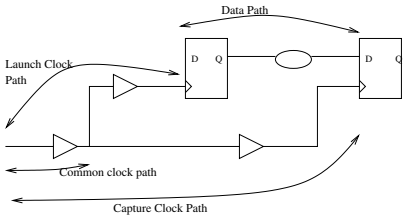


Fig.2 A standard register to register timing path

Some of the terminologies used in this paper are defined in this section. Fig.2 shows a standard timing path between 2 registers. Capture Clock Path: The path as traced by the clock from the source to the capture register.
Launch Clock Path: The path as traced by the clock from the source to the launch register.
Data Path: The path traced by the signal from the launch register to the capture register.
Clock Common Path: The part of the path common between the launch and capture clock paths.
Timing Window: It is a window between a minimum and maximum times where a transition can occur.
Max Path Delay: The maximum delay a path can incur.
Min Path Delay: The minimum delay a path can incur.
Voltage Waveform : The instantaneous difference between the power and ground voltages seen by a cell.

We now discuss the current and the proposed approaches that can be used for timing analysis.

3.1 Current Approach

A simple and straight-forward approach is to assume that the voltage waveforms through the duration of power grid analysis are directly relevant for timing computation. From the voltage waveforms of each cell, compute the *Vmin* and *Vmax* for each cell as the absolute minimum voltage and the maximum voltage seen by the cell during the whole duration of analysis. The *Vmin* hence computed will be used for max delay condition for the cell and the *Vmax* will be used for the minimum delay condition for the cell. Thus for setup analysis, the launch clock path and the data path will work on the *Vmin* voltage and the capture clock path will work on the *Vmax* voltage. The converse will happen for hold analysis. In general, the *Vmin* and *Vmax* numbers can be the absolute minimum and maximum voltages that could otherwise be assumed for the design under consideration. This is normally how designs are timed. In general, for setup timing analysis, the clock common paths should be operating at different voltages when evaluating launch and capture paths, as the IR drop numbers could differ for the same cell at different clock edges.

The advantages of this technique are that the method is straight-forward, and that the approach is independent of the design timing mode. However, this method is likely to

be pessimistic in terms of assessing performance impact as only the voltage variations around the duration of switching are likely to impact the delay through a circuit, and that the absolute minimum and maximum values only bound the voltage of relevance.

3.2 Proposed Solution

3.2.1 Using Timing Windows

One way to reduce pessimism in the voltage computation is to analyze the voltage waveforms of the cells in the backdrop of their switching scenarios. Ideally, what affects the delay of a switching circuit is the voltage the circuit sees during its switching. Knowing the path based timing window [6] of switching for a cell, we could compute the *Vmax* and *Vmin* for any cell as the maximum and minimum voltage in the cell's timing window as against the full waveform. These voltages can now be used for static timing analysis. The assumption here is that accurate voltage waveforms corresponding to the scenario of operation of the design is available.

The advantages of this technique are that this approach is computationally not intensive, while simultaneously removing a good amount of pessimism in timing analysis. However, the approach is likely to be useful only if the timing window for an instance is small enough compared to the full duration of analysis. Considering that the timing window of a cell is dependent on the fanin cone, this assumption may not be true for design cells with large fanin logic cones. Fig 3 shows the effect of large fanin cones to the timing window at a node.

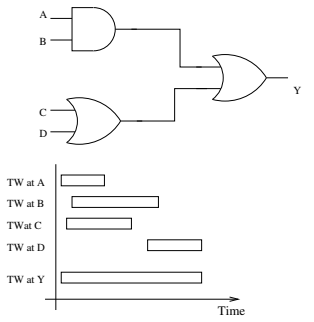


Fig.3 Effect of fanin cones on the timing window at a node.

Also, for designs with multiple clocks, this approach would necessitate consideration of the overall switching duration independent of clocks, resulting in some amount of pessimism to be retained in the analysis.

3.2.2 Shrinking the Timing Window for Voltage Computation

In this section, we look at the two options causing potential pessimism: large timing windows for some cells, and design scenarios with multiple clocks. Fig.4 shows the time-variant voltage of a cell instance and the timing window of consideration.

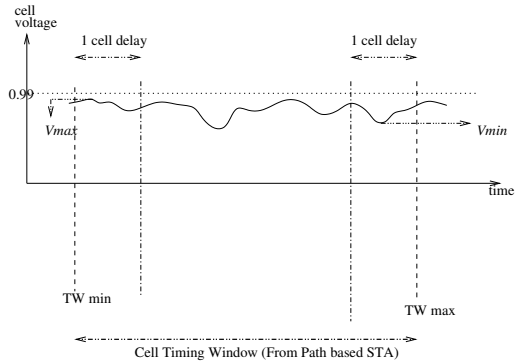


Fig.4 A sample time variant voltage for a cell and its timing window

If *TWmax* is the right edge of the timing window, then transitions happening at this *t=TWmax* on the launch clock and data path will be the most constraining for setup analysis. Similarly, transitions happening at *t=TWmin* on the capture clock path will be most constraining for setup analysis. The converse will be good for hold timing analysis. To compute a "realistic" *Vmin*, we could consider a small window around *TWmax*. The width of this small window is assumed to be 1 cell delay (*Td*) of the corresponding cell. This essentially means that there can not be a lower voltage before *t=TWmax-Td* which can cause a delay degradation of the cell beyond 100%. Thus

Vmax = minimum voltage seen by the cell b/w *t=TWmax-Td* and *t=TWmax*

Vmin = maximum voltage seen by the cell b/w *t=TWmin* and *t=TWmin+Td*.

These voltages are now annotated on the cells and static timing analysis is performed. Note that for every cell, the timing window is analyzed at the output pin of the cell. Also, only for the capture flop, the minimum voltage in the timing window corresponding to the clock pin needs to be used so as to account for the worst setup time possible.

This method assumes that the delay degradation on a cell instance cannot be more than 100% due to voltage considerations alone. This is definitely a practical assumption to make as the voltage effects in designs rarely causes more than 10-15% degradation in timing.

This definitely makes the timing window considered for assessing voltage variations realistic, while requiring additional processing of the voltage waveforms. Since we now analyse only one timing window at a time, we still have a limitation of needing separate analyses for different clock phases of interest, where the related timing windows for switching could be different. Another main drawback of this approach is that timing degradation that propagates

through the logic cone is not comprehended in the impact analysis, and hence there is a new error component that is introduced in the timing computation.

3.2.3 Performing Iterations for Convergence

We now look to address the limitation of the previous approach not comprehending progressive delay degradation on a timing path as a result of voltage fluctuations. One way to solve this problem is to do an iterative timing analysis.

After doing a static timing analysis with *Vmax* and *Vmin* computed as described above, we could recompute the timing windows of all the cells. The *Vmax* and *Vmin* are recalculated using these new timing windows and another round of static timing analysis is done. This loop can be repeated until convergence is achieved.

Convergence criteria : We show that there is a point of convergence in iterative timing analysis using the above technique. We define that the timing analysis has converged if and only if the timing numbers for every cell on the path across two consecutive iterations of analysis remain the same or if the timing delta across a given iteration comes out positive compared to the previous analysis.

We can show this to be true and valid logically.

- Assuming that across iterations there does come out a negative timing delta, we understand that the analysis has not converged.
- However, if the timing deltas turn out to be zero, then the timing windows considered for identifying voltages of interest have not changed, hence any successive iteration would only continue to indicate the same timing numbers, directly indicating convergence.
- On the other hand, if the timing deltas across iterations do come out positive, it just shows that the more pessimistic of voltages was actually comprehended in the previous iteration, this previous iteration being the point of convergence. Any further analysis would only get the timing numbers closer to the previously analysed point as the minimum/maximum voltages of interest would only tend to the numbers used in the previous iteration.

While the feedback mechanism makes the flow robust and ensures accuracy of results, this method is prohibitive in terms of compute time for realistically large designs, more so considering that the approach would still handle only one clock phase of interest in multi-clock design scenarios.

3.2.4 Progressive Timing Window Adjustment for Voltage Computation

Considering the definition of the point of convergence above, and the propagation of the delay degradation through logic cones, we now look to improve upon the definition of the actual window of timing used for assessing voltage fluctuations. We propose a heuristic approach, where we intend to find minimum/maximum voltages around the timing window edge of interest (*TWmin/TWmax*). Previously, we proposed the use of a small window on either side of these edges, we now propose that we could avoid the iterative analysis by considering an augmented window on either-side of these edges. Fig.5 shows the modified timing window as per the new approach.

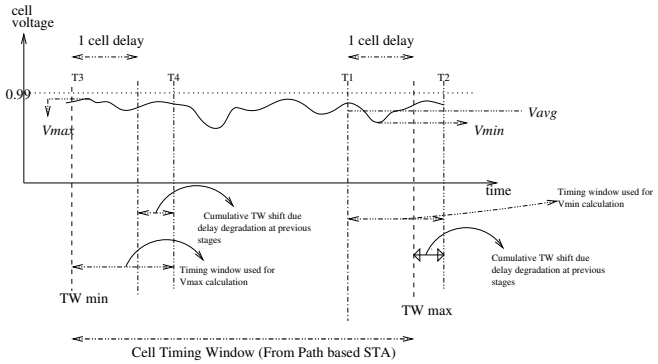


Fig.5 Cell voltage waveform and its modified timing window

For the nth cell of a path the *Vmin* and *Vmax* is computed as:

Vmin(n) = minimum voltage in the timing window between T1 and T2 where,

$$\begin{aligned} T1 &= TW_{max}(n) - T_d(n) \\ T2 &= TW_{max}(n) + T_{shift\_cum\_max}(n) \end{aligned}$$

Vmax(n) = maximum voltage in the timing window between T3 and T4 where,

$$\begin{aligned} T3 &= TW_{min}(n) \\ T4 &= TW_{min}(n) + T_d(n) + T_{shift\_cum\_min}(n) \end{aligned}$$

Where:
TW$_{max}$(n)= Max edge of the timing window of the nth cell of the path after STA.
TW$_{min}$(n)= Min edge of the timing window of the nth cell of the path after STA.
T$_d$(n)= Delay of the nth cell at the nominal voltage.
T$_{shift_cum_max}$(n)= The cumulative shift in the timing window in the maximum corner at the nth stage due to voltage degradation along the path given by the following formula.

$$\begin{aligned} T_{shift\_cum\_max}(n) &= T_{shift\_cum\_max}(n-1) + \\ &\quad \delta D(n-1) \\ T_{shift\_cum\_max}(0) &= 0 \end{aligned}$$

δD(n) = Delay delta for the nth cell due to voltage drop. STA using CCS scaling [7] techniques can compute this.

Alternatively, a worstcase voltage-delta to delay-delta scale-factor can be calculated for the library and used for the computation of δD(n). It is to be noted that the assumption of a worst-case scale-factor for delay dependence on voltage has only an indirect effect on the results. This effect can be shown to be pessimistic by assessing the timing-stage based delay degradation and ensuring that this is smaller than the timing window shift as predicted by the use of the scale-factor.

Given these newly computed voltage numbers, we can now estimate the voltage-induced timing degradation directly in a single iteration.

This technique now avoids the iterative analysis in entirety, with the only overhead of a preliminary design independent library analysis to compute the scale factor used for delay degradation estimation. In case the analysis shows that the shift in arrival times at a particular node in a timing graph is worse than what was assumed using the scale-factor, the analysis iteration can be repeated to reach convergence however, this is likely to be a remote possibility so long as the scale factors are estimated in a conservative fashion. Also, the CPU time for computing the cell voltage from the waveforms is negligible and it is linear to the number of cells on the path of consideration.

3.2.5 Enabling simultaneous analysis of multiple paths

We now try to address the problem of not being able to realistically comprehend multiple clock phases of relevance, as also trying to see if the voltages for each cell can be made independent of the paths being analysed.

Multiple paths can be analysed in the same analysis session if we can have one voltage annotation per cell that accounts for all paths passing through that cell. The following approach can be used for this purpose:

1. Get all possible timing windows at the output of every cell. This includes all clock domains and edges.
2. Using the method specified in the above technique, compute the min voltage in each of the timing windows. The minimum of these voltages can be used as Vmin for STA. A similar approach can be used for Vmax.

It is to be noted that this method will not be the most accurate but is just a means of trading off run-time cost with pessimism in the delay degradation assessment.

4 Results

The various techniques presented above were used in a large real-life design. This design was in the 90nm node, and was a multi-core chip of 202 sq.mm. area, designed in a 7-layer flip-chip process. Paths of various clock groups across the design were considered for analysis. The run time and CPU requirements depend on the size of the design being analysed. The analysis runtime was around 2hours for the design mentioned using a 64bit Linux box and 16GB of memory. Fig6 shows the slack for each of these paths, with each of the techniques described above.

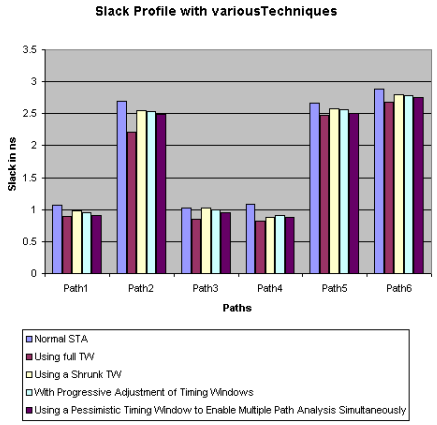


Fig.6 Slack distribution for 6 paths using various techniques.

Fig7 shows the path delay degradation due to voltage drop with various techniques as a percentage of the path delay in the normal static timing analysis.

As can be seen, the slacks for the paths for each of the techniques become more positive as we refine the techniques. Also, for the last technique described wherein multiple paths can be analysed simultaneously, the slack reduces due to the pessimism we added in the voltage comprehension. Note that the path delay degradation does not go beyond 6% which tallies with our margin assumption.

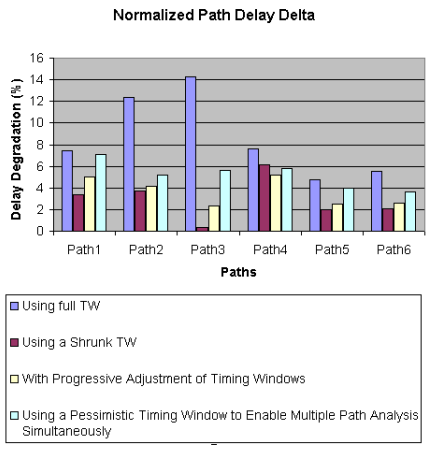


Fig.7 Profile of Path delay degradation with various techniques.

SPICE analysis was done on a few paths for correlation analysis, the results for one the paths is presented. The voltage waveform obtained from dynamic IR drop analysis was used for every cell of the path in the SPICE netlist. Sample waveform for one of the buffer cells in the path is shown in Fig8.

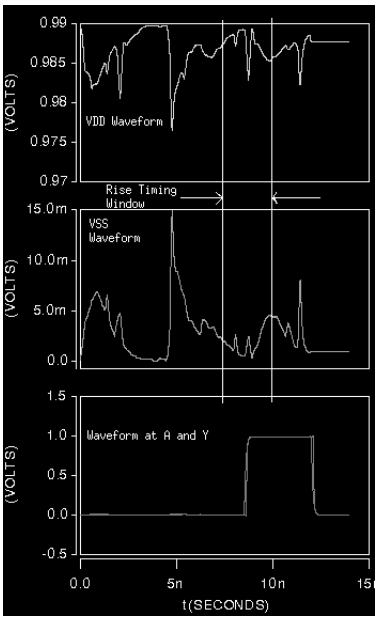


Fig.8 Voltage waveforms for a buffer on a sample path during SPICE analysis.

The path slack for the analyzed path with various techniques are summarized below.

- Path slack reported by margin based STA : 2.5678ns.
- Path slack reported by the proposed flow (Section 3.2.5) : 2.826ns
- Path slack reported by SPICE analysis : 2.9850ns.

So the proposed methodology removes significant amount of pessimism inherent in the margin based STA approach. The run time and CPU overhead of the approach on a normal static timing analysis is also minimal. The above results also highlight improvement of path slack due to lower voltage drops which could effectively be used for critical path analysis.

5 Conclusion and Future Work:

We have proposed a solution to gauge the performance impact due dynamic voltage fluctuations in the design. The above methodology enables us to directly measure the impact of power network robustness on the system performance. Also, this framework can be used to validate quality of the power network for a given performance target. The methodology enables reduce significant pessimism associated with the conventional margin based approach to account for IR drop. This should help better performance correlation between static timing analysis estimates and as observed on silicon. Also, this technique could be effectively used to analyse and signoff critical paths of the design, taking the benefit of a good quality power network if applicable.

As future work in this regard we need to correlate the performance predicted by the proposed methodology and that observed on silicon. As a first step, we plan to correlate the results with SPICE simulation on a wider range of paths. The effort for this is ongoing. The methodology further needs to be refined to handle cells with multiple rails like memories/IOs effectively. Handling crosstalk noise induced delay effects is another area of future work in this direction. The methodology does not account for the changes in the voltage waveform due to shifts in timing window/arrival times due to voltage degradation. This is something we plan to look into as an iterative approach.

References

[1] D Kouroussis, R Ahmadi, , and F.N. Najm. Voltage-aware static timing analysis. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 25:2156 – 2169, October 2006.

[2] M Graziano, C Forzan, and D Pandini. Including power supply variations into static timing analysis: Methodology and flow. *SOC Conference*, pages 229 – 232, September 2005.

[3] S Pant and D Blaauw. Static timing analysis considering power supply variations. *International Conference on Computer Aided Design*, pages 365–371, November 2005.

[4] S Lid, M Nagata, K Shimazak, K Satod, M Sumita, H Tsujikawa, and A.T. Yang. Full-chip vectorless dynamic power integrity analysis and verification against loouv/loops-resolution measurement. *Custom Integrated Circuits Conference*, pages 509–512, October 2004.

[5] Solvnet Article. CCS Timing. Homepage. `https://solvnet.synopsys.com/`.

[6] Aravind NV, Rajagopal KA, Ajith HS, and Suparna Das. Path based approach for crosstalk delay analysis. *VLSI Design*, pages 727–730, 2004.

[7] Solvnet Article. CCS Voltage and Temperature Scaling for Timing and Noise. `https://solvnet.synopsys.com/`.

Optimisation Quality Assessment in Large, Complex SoC Designs – Challenges and Solutions

R.Venkatraman[*], Shrikrishna Pundoor[**], Arun Koithyar[*], Madhusudan Rao[*], Jagdish C.Rao[*]
[*] *Texas Instruments India Ltd., Bangalore, INDIA*
[**] *Analog Devices India Ltd., Bangalore, INDIA*
rvenkat@ti.com, shrikrishna.pundoor@analog.com, arun.koithyar@ti.com, bgm-rao@ti.com, j-rao@ti.com

Abstract

Design optimisation of large system-on-chip (SoC) designs presents significant challenges in terms of the number of care-abouts in today's chip design scenario. While optimisation approaches have focused on key aspects like design timing, power and routability, there are several critical aspects that do not get modelled by the abstraction approaches used in current solutions, resulting in local areas of potentially bad quality of results (QoR). The definition of design quality metrics for assessing various aspects of interest thus comes out as a key requirement in designs today. In this paper, we highlight the key challenges in the design of large, complex SoCs and propose some design metrics that identify problem areas for improvement. We highlight how conventional abstraction schemes tend to mask such problems when looking at design-level averaging for the relevant cost-functions. We present the results of these metrics and also show why not resolving some of these potential bottle-necks could lead to significant challenges in overall design closure.

I. Introduction

With increasing chip sizes and shrinking technology nodes, the complexity of design optimisation continues to grow. Conventional optimisation techniques break-down when expected to optimise long interconnects and handling macro-dominated floor plans. In this paper, we analyse select practical problems in chip design closure and highlight the challenges that need be addressed with today's complex designs and propose design metrics that help identify problem areas and provide better diagnosis capabilities to address QoR gaps.

The rest of the paper is organised as follows. Section-II details the motivation behind this work, where we also highlight recent work done in the area of concurrent multiple cost-function optimisation. In section-III, we identify select problem areas and challenges and propose metrics to quantify the quality of optimisation, indicating results that were used to improve design QoR. Section-IV summarises these results, concluding the paper, with our thoughts on how optimisation approaches could be improved upon.

II. Motivation

Current-day design complexities tend to accentuate gaps in some of the optimisation algorithms used for chip design. While the focus of these algorithms has always been on multi-cost optimisation (for e.g., placement algorithms have been concurrently optimising for timing and congestion for some time now), the complexities of design scenarios like large die-sizes, significantly large macro component in the design (like the use of memories, pre-designed hard-macros etc.), the significance of the interconnect in design closure and the importance of estimating the effects correctly early in the design phase, and a large increase in the bus-widths and the associated complexities all add to concurrent and effective optimisation challenges.

Design optimisation and its various components viz. placement, buffering, global-routing and optimisation are well researched-upon topics, considering different criteria for design optimisation. [1] proposes a means of interconnect planning in a design through block buffer planning. This highlights the fact that the feasible region for buffer placement is large, even in cases of tight delay constraints. An algorithm for identifying optimal buffer locations is presented in [2]. This work highlights ways to identify optimal buffer locations while balancing overall placement densities. This helps in minimising the placement impact of subsequent optimisation techniques, so that the buffers do not move away from the optimal locations. The need for custom optimisation techniques to handle specific design scenarios in complex designs is highlighted in [3,5-12]. A refinement-based design approach for shrinking

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.52

process nodes, given the problems in conventional timing-closure approaches and design variable dependencies is highlighted in [4].

This paper focuses on assessment metrics for several practical scenarios seen in today's designs. While the authors believe that several of these are straight-forward definitions of sub-optimality, the point we would like to highlight is that several commercial automation solutions today lose sight of these problems in the abstractions of cost metrics used for the targeted holistic solutions. Hence an awareness of design issues with respect to some of these problem scenarios would help in faster design closure and in overall better QoR.

III. Optimisation Assessment & Metrics

We broadly classify the definition of metrics into four categories, according to the typical physical design flow: standard cell placement assessment, repeater synthesis assessment, optimisation quality assessment and route-quality assessment. The scenarios shown in the following sections are all from a 202 sq.mm., 90nm, multi-core flip-chip design which posed considerable timing-optimisation and routability challenges to the optimisation environment.

A. Placement assessment

Conventional placers optimise the design for timing, congestion and wire-length. This being an NP-complete problem, the heuristics used during placement optimisation do impact the quality of optimisation significantly. We propose a simple assessment of placement quality of identifying if logic in the fanout cone between two identified netlist cells actually get placed in the minimum-containing rectangle (MCR) of these two cells (Fig.1)

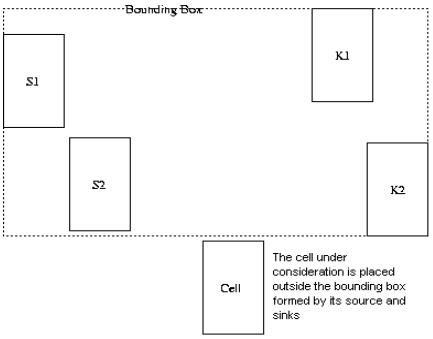


Fig.1 Sub-optimal placement. S1, S2 are source for cell under consideration, where as K1 and K2 are sinks for the same. Dotted line represents the bounding box drawn by the farthest source/sinks. The cell under consideration is placed outside the bounding box

We propose to do a placement assessment for every cell in the design by thus looking at the immediate predecessors in the fan-in cone and the immediate successors in the fan-out cone. The following pseudo-code highlights this simple procedure.

Let D = [Set of all drivers in the design]
Let D'= NULL
For every d ∈ D {
Let S = [Set of all immediate successor cells of d]
Let P = [Set of all immediate predecessor cells of d]
M = Minimum containing rectangle(P and S)
M' = M, expanded with a known tolerance δ
D' = D' U {d} if { d not placed in M'}
}
Return D'

In this, we allow a small tolerance for the placement by expanding the MCR by a factor δ. While this relaxes the goals for the check, the authors have found this efficient in comprehending practical constraints during the assessment. A distance equal to about 5 standard-cell placement rows has been found to be sufficient for this tolerance when analysing large fan-in and fan-out cones. While the factor δ itself could be made cell-count dependent (i.e. dependent on the size of union of S and P: cell-count N = |S U P|), the authors have not used such an approach in their assessment. Fig.2 shows an instance of a real design scenario of sub-optimal placement. While the reason for the sub-optimality was actually high local utilisation in the area of interest as seen from the snap-shot, conventional solutions to the long-nets created by poor placement would be to add more repeaters to the design, further increasing the design utilisation, resulting in a non-convergent design scenario or overall poor QoR.

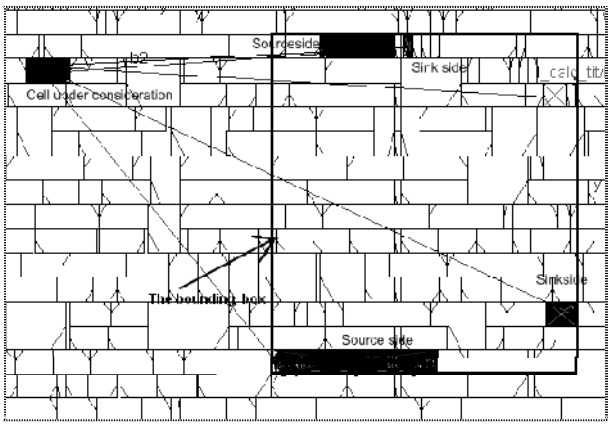


Fig.2 A practical sub-optimal placement. There are no visible standard cell blockages in the region.

The real solution to problem scenarios like these is to control cell placement to ensure that the local utilisation in regions of interest does not cross an upper-limit, which could be determined based on the number of poorly placed cells as identified by the proposed metric, and their distribution. It is to be emphasised that conventional congestion relief and wire-length minimisation techniques used as cost functions in placement may not resolve problem scenarios like the one in Fig.2.

B. Repeater synthesis assessment

(i) On-route buffering assessment. The placement metric defined in III-A can be customised to assess specific qualities of repeater trees. When analysing design critical paths or areas of design congestion, we typically find scenarios of inefficient repeater insertion, with the repeaters (or repeater-chains) not getting placed within a reasonable distance from the minimum-containing-rectangle of the driver and the sink cells under consideration, the interconnection between which was the original candidate for repeater-insertion. III-A can be customised to analyse all buffer trees, with the MCR being defined by the pre-buffeing driver and its sinks. This assessment highlights a simple but useful means to assess poor buffering topologies, which otherwise might not be noticed, but would limit design entitlement in a given floorplan.

The authors would like to point out that in most commercial solutions, design placement and repeater synthesis are two distinct design optimisation phases, which use very different algorithms and cost-functions (wire-length, congestion, timing for placement; and timing, area and design transition time constraints for buffering). Hence a placement analysis of the repeaters in the design (as proposed for the generic case in III-A) would also be an independent useful assessment.

(ii)Over-buffering metric. A standard-cell library can be pre-assessed for optimal performance and quality by analysing the drive capabilities of the repeater cells, and understanding an optimal typical buffering distance, given the routing assumptions. Given this information, and given the assessment method (i), we now can analyse whether a repeater, even when optimally placed, is actually necessary or not.

Let L =Optimal typical wire-length for repeater addition
Let D= [Set of non-buffer/inverter drivers in the design]
Overbuffered = null array
For every d ∈ D {
Let S = [Set of all non-buffer/inverter sinks of d]
Let B = [Set of all repeaters on the output of d]
M = Minimum containing rectangle(d and S)
M′≒ M, expanded with a known tolerance δ
P = semi-perimeter of M′
Expected buffer count, E = P/L
Over-buffered(d) = TRUE if (E > size(B)), else FALSE
}

While the above could be directly applied if the repeaters used are all of a given drive-strength, this may need some considerations when we have repeaters of different electrical drive capabilities. One option in the latter scenario is to make the parameter L dependent on the drive-strength, and analyse the design on a cell basis:

Let L(r) = Optimal typical wire-length for repeater r

Electrically, the capacitive loading on the repeater is what defines its optimality, and not the wire-length. Considering that the loading is dependent on the layer used for routing (as parasitic loads in DSM designs are very dependent on the metal layer used), this particular assessment could get tedious. To over-come this, the authors propose a variant of this metric to first assess the drive-strength of repeaters for optimality.

(iii) Over-driving repeaters. Knowing the optimal driving distance for each repeater, or, consequently, the maximum capacitive load for each repeater, we could modify the definition of metric-(ii) above to analyse whether given a known distance (and hence the capacitive load under known routing assumptions), the right drive-strength has been used. This metric is very useful for assessing long interconnects in the design which potentially cause significant routability and timing optimisation challenges, and are vulnerable to poor repeater QoR. Fig.3 shows a profile of a block in the design, highlighting how many over-sized repeaters exist in the design.

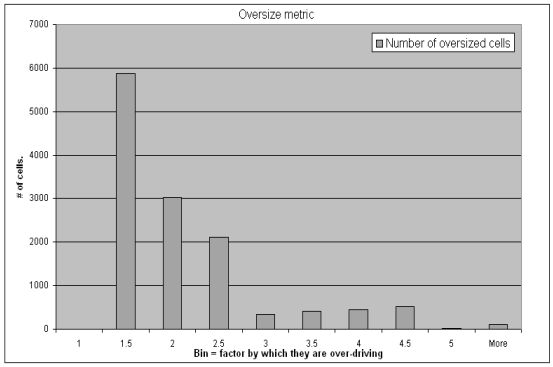


Fig.3 Drive strength histogram.

Reviewing the results of a section of repeater trees in the design, we conclude that the trees actually have several over-driven states, with potentially 50% of the

buffers over-driving by a factor of 2 or more, compared to an optimal design scenario. These results indicate potential design power dissipation issues that could be alleviated by appropriately optimising the repeater trees, with practically negligible impact to design timing.

(iv) Repeater tree assessment. We now look to combine the above metrics which individually assess connectivity and library cell choice to help identify potential issues in repeater tree construction on high-fanout nets.

Ideally, the construction of a repeater tree should result in as short an overall wire-length as possible, which is possible only if the buffers all follow the topology of the minimum Steiner tree (MST) for the net. A common issue in conventional repeater trees is an early fanout split in the tree, resulting in multiple paths from the root of the tree reaching a local area near the sinks (Fig-4).

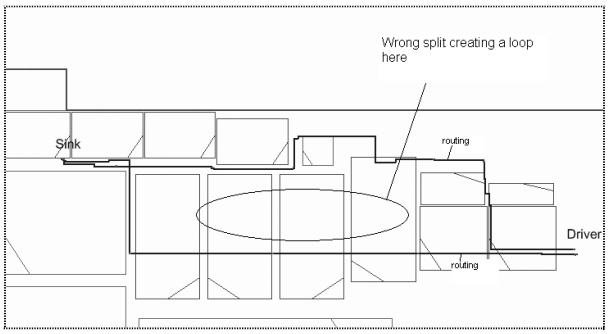


Fig.4 Early bifurcation of repeater tree

We propose to identify such issues by a sliding window technique to look for redundant nets of the same logic in the repeater tree

Let D = [Set of non-buffer/inverter drivers in the design] | (fanout > threshold)
Let W = Window size of interest
Let Bad_W = null array
For every d ∈ D {
Let S = [Set of sinks of d]
Let M = MCR(d and S)
For every window w of size W⊂M{
If (w contains duplicate wires for the same signal* in the same direction**){
Add w to Bad_W{d}
}
}
Return Bad_W

Note:
**: This assumes the same logic in any unateness through the repeater tree (inverted or non-inverted)*
***: Horizontal or vertical*

Fig.5(a) shows another real-design scenario with a larger amount of redundancy due to the loops. This unnecessarily increases the design routing congestion causing other serious design quality issues like routability and timing closure challenges. Fig.5(b) shows what the ideally intended connectivity is when the high-fanout net is routed without any repeater insertion. In an ideal repeater synthesis scenario, the repeaters need to follow the global route topology without causing loops in the tree. The authors would like to highlight that small deviations from this topology are normally expected, which is what is modelled in the placement assessment metric defined earlier as a tolerance window.

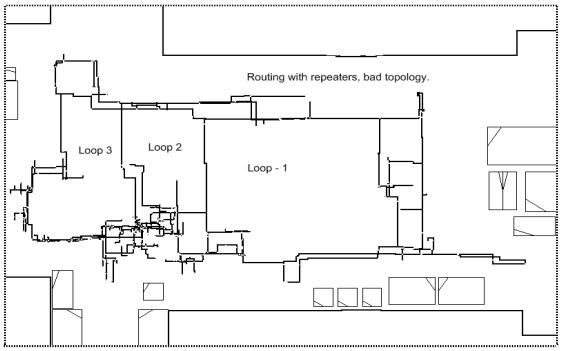


Fig.5(a) Sub-optimal repeater tree with loops. The entire tree shown here is post-repeater scenario of a single net.

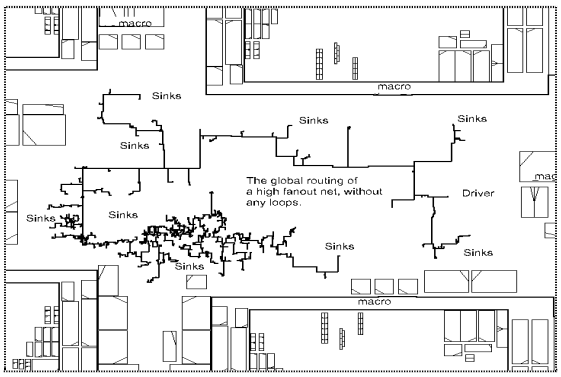


Fig.5(b) Global route of the same high fanout net, which ideally should be the routing topology to be retained even after repeater insertion.

C. Route quality assessment

(i) Routing wire length deviation. Complex floorplans dominated by many channels pose problems at different stages of routing, that is from estimates during placement, to global route and finally detailed routing. Changes from a Manhattan estimate to actual global route may impact the timing of a design drastically. Although modern routers account for many different constraints, the deviations exist in numbers large enough to affect design quality and

warrants iterations or incremental refinement. Similar deviations occur from global to detailed routing, this time however, due to requirements of complex DRC rules on the detailed router, more so in 65nm and 45nm technologies. The requirement of a wide-via, for example, may dictate the route to use a different floorplan channel compared to what was suggested by the global router. We propose a metric where we measure the wire-length deviations at these two levels to isolate the timing/congestion problem that happens only due to this mismatch in the wire-length. The deviations are reported as a histogram of error between Manhattan & global-routing estimated and also between global and detailed routes.

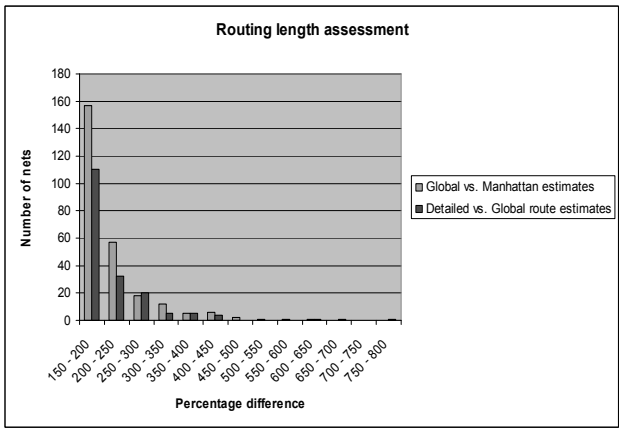


Fig.6 Wire-prediction miscorrelation through the design phase: Manhattan estimates to Global-routing estimates and further on to detailed-routed wire-lengths

The assessment results from a reasonably large congested block in the design (~147K nets) shown in Fig-6 indicates that several nets have unexpected detours and hence add to unpredictability in design closure. Only nets longer than 20um have been considered to ensure that the percentages are not skewed by short-net observations. These results clearly indicates that overall wire-length optimisation in the presence of congestion and timing challenges can potentially be sub-optimal for some nets (~3-4%) in the design.

(ii) Routing predictability of critical nets. While the previous metric does identify long detours, short jogs do tend to get masked in the assessment. In the detailed-routing phase, short jogs are created on nets when addressing layout-design rule check (DRC) errors. The DRC closure operations are usually not timing-aware in most design environments. While this has been a reasonable assumption to be made in earlier technology nodes, sub-100nm designs actually pay a penalty due to the via-resistances which are high in these nodes. We propose a metric to assess timing criticality of nets before, during and after the DRC closure, to assess the impact, and to suitably guide the DRC-closure algorithms to not impact critical nets of interest. Fig. 7 shows an analysis of about 1000 stages (cells + nets) of critical nets in a design, and the corresponding delay (in picoseconds) change through DRC-closure.

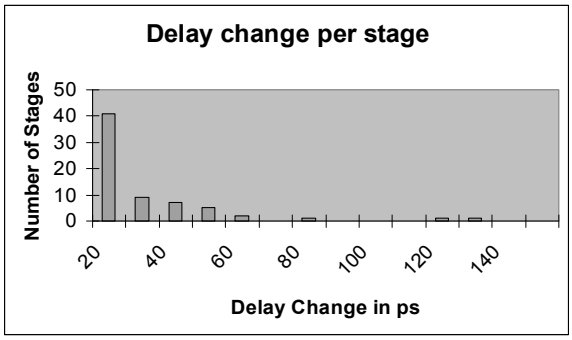


Fig-7 Critical net routing predictability with DRC closure

D. Optimisation Quality Assessment

We now propose additional metrics to profile the quality of optimisation in a design.

(i) Wire length Vs Slack metric. We propose to identify outliers in the plot of a path slack against the wire-length of a particular net under consideration. A bad placement of cells in the critical path could result in long wires, and unnecessary repeater requirements or timing-closure challenges. Critical nets should remain short enough to be shielded from layer-assignment discrepancies at global/detailed routing and any potential congestion-relief mechanisms which could result in detours. Fig 8 shows a snap-shot of the design through the optimisation phase. The encircled points are the areas of concern where the optimisation schemes need to ensure that relevant care-abouts are comprehended for these nets.

(ii) Slew Vs Slack metric. We propose to review slew plots against the corresponding slack at a given node in the design as an aid to identify additional optimisation issues and help find solutions. All nodes in the design need to meet a maximum transition time constraint, but timing critical nodes should ideally see the best slews possible while non-timing-critical nodes should be close to the worst transition-time constraint as specified, to help optimise on design area and wire-length. [12] proposes a power-optimisation technique using such an analysis. Fig.9 shows a profile of this assessment on a block in the design, which can also help fine-tune critical-path optimisation.

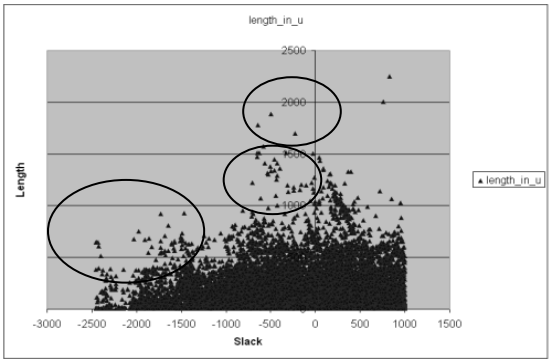


Fig.8 Wire length Vs Slack histogram

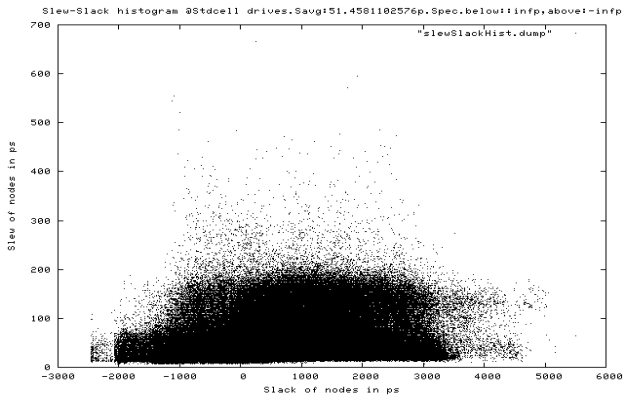


Fig.9 Slew-slack histogram. It is to be noted that though the design global target was 400ps, nodes with even +2ns slack have very sharp slews.

IV. Conclusion and Future Work

The authors have used these methods to assess the quality of optimisation in a complex system-on-chip (SoC) design in a 90nm, 7LM design process. This design was a large (202 sq.mm.), multi-core CPU system, with both timing and power targets being critical. The design was considerably macro-dominated, the macros being of various sizes, thus creating challenges of high complexity to conventional commercial placement optimisation solutions in handling the floorplan. All the techniques highlighted in this paper have been automated to help generate relevant metrics several times through the design optimisation phase, and have been suitably integrated into the design environments used. The run-times for generating the profile data was only a small percentage of the overall design time (typically <2% of the overall optimisation cycle-time). These analyses aided the design closure process immensely, both in terms of identifying the right problems to fix, while also highlighting the best optimisation strategies for use with commercial optimisation tools.

In this paper, the authors have highlighted and presented design assessment techniques which help identify issues in design optimisation which are not otherwise modelled accurately by the abstracted cost-functions used during various phases of optimisation. These metrics have helped the authors guide the optimiser to focus on root-causes of potential problems in the design-closure a complex SoC.

While the authors note that some of these metrics are simplistic, we would also like to highlight that there is a need to augment currently available commercial optimisation solutions to help address some of these potentially basic limitations to help improve overall QoR of a design. A typical case in point would be in terms of handling a mix of macros and standard-cells in design optimisation, which usually is the main source of several of the sub-optimalities highlighted in this paper. With focus on further improvement in optimisation that is potentially likely, the authors are also looking to define additional metrics to help assess congestion and clock tree synthesis and routing under various view-points.

References

[1] Jason Cong et.al., "Block buffer planning for interconnect planning and prediction", *IEEE Trans. VLSI Sys*., pp: 929-937, Dec 2001

[2] Charles J. Alpert et.al., "A Fast algorithm for identifying good buffer insertion candidate locations", *Proc. Int. Symp. Phy. Des*, *2004*

[3] Peter J. Osler et.al., "Placement Driven Synthesis Case Studies on Two Sets of Two Chips: Hierarchical and Flat", *Proc. Int. Symp. Phy. Des 2004*

[4] Olivier Coudert et.al., "Timing and Design Closure in Physical Design Flows" *Proc. Int. Symp. Qual. in Elect. Des., 2002*

[5] Tim Chan et.al, "hallenges of CAD Development for Datapath Design", *Intel Technology Journal, Q1, 1999*

[6] X. Tang et.al., "Planning buffer locations by network flows," in *Proc. Int. Symp. Physical Design, 2000*

[7] P.Sarkar et.al., "Routability-driven repeater block planning for interconnect-centric floorplanning," in *Proc. Int. Symp. Physical Design*, *2000*.

[8] C. J. Alpert et.al.,"Buffered Steiner Trees for Difficult Instances", *IEEE Trans.on Computer-Aided Design*, 21 (1), 2002, pp. 3-14.

[9] J. Lillis et.al., "Simultaneous Routing and Buffer Insertion for High Performance Interconnect", *Sixth Great Lakes Symposium on VLSI, 1996*.

[10] T. Okamoto et.al., "Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization", *IEEE/ACM Intl Conf on Computer-Aided Design, 1996*

[11] P. Saxena et.al., "The Scaling Challenge: Can Correct-by-Construction Design Help?", *Proc. Intl. Symposium on Physical Design, 2003*

[12] Shrikrishna Pundoor et.al., An efficient method of leakage optimization in timing critical designs, *IEEE Elect. Des. Proc. Symp., 2007*

Unified Challenges in Nano-CMOS High-Level Synthesis

Abstract:

The challenges in nano-CMOS circuit design include the following: variability, leakage, power, thermals, reliability, and yield. This talk will focus on interdependent consideration of these challenges during high-level (aka architectural or behavioral) synthesis. The majority of the existing design techniques related to these challenges are at the device or logic level of circuit abstraction and few are at the architectural level, however, the research is full swing in this direction. At the architecture level, there are balanced degrees of freedom to vary design parameters and take fast and correct design decisions at an early phase of the design cycle without propagating the design errors to lower levels of circuit abstraction, where it is costly to correct them. In addition, designing at higher levels of abstraction is an efficient way to cope with complexity, facilitate design verification, and increase design reuse.

For maximizing yield of circuit design in the presence of variability the designers can rely on pre-silicon or post-silicon techniques. The pre-silicon techniques are statistical optimization approaches of design phases that use statistical power, leakage, and timing analysis for design space exploration and maximize the parametric yield. A variety of approaches for scheduling, resource sharing, and module selection techniques have been proposed in current literature in this respect. The post-silicon techniques are approaches like adaptive body biasing and adaptive supply voltage which are used to tune the fabricated chips such that the circuit yield can be optimized. This talk will discuss all these techniques proposed in the context of HLS.

Speaker Bio:

Saraju P. Mohanty is an assistant professor in the Dept. of Computer Science and Engineering at the University of North Texas, USA. He obtained his Ph.D. in Computer Science and Engineering from University of South Florida, USA, in 2003. His research is in Design and CAD for Nanoscale Digital and Analog/Mixed-Signal Circuits. He researches power, leakage, and timing models, incorporates them in Design/CAD flow through optimization methodology, and demonstrates them through computationally intensive multimedia applications. He is the author of 75 peer reviewed journal and conference publications and 1 book and the inventor of 2 (pending) patents. He is a senior member of IEEE.

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.124

Session 8C

Processor Design and Scheduling

Exploring the Limits of Port Reduction in Centralized Register Files

Sandeep Sirsi and Aneesh Aggarwal
Electrical and Computer Engineering Department
Binghamton University, Binghamton, NY
{aneesh@binghamton.edu}

Abstract

Register file access falls on the critical path of a microprocessor because large heavily ported register files are used to exploit more parallelism. In this paper, we focus on reducing register file complexity by reducing the number of register file read ports. The goal of this paper is to explore the limits of read port reduction in a centralized integer register file i.e. how few read ports can be provided to a centralized integer register file, while still maintaining performance? A naïve port reduction may result in significant performance degradation and does not give a true measure of the limits, while clever techniques may be able to further reduce the number of ports. Hence, in this paper, we drastically reduce the number of ports and then investigate techniques to improve the performance of the reduced-ported register file. Our experiments show that the techniques allow further port reduction by improving the performance from reduced-ported RFs. For instance, with our experimental parameters, the naïve port reduction method requires at least five read ports to maintain a performance impact of less than 5%, whereas, our techniques require only three ports.

Keywords: Complexity-effective design, Register file, Port reduction, Instructions per cycle

1 Introduction

Modern microprocessors use heavily ported large register files (RFs) for exploiting instruction level parallelism. Since the register file lies in the critical path of dependent instruction execution, heavily ported large register files have significant clock cycle time and energy dissipation implications in microprocessor design [1, 3, 4]. In fact, a study [5] has shown that for current out-of-order superscalar processor designs such as MIPS R10000 [7] and Alpha 21264 [6], RF consumes the largest fraction of the total chip power consumption.

One method to reduce the RF complexity is to have multiple RF banks. In this method, a single centralized RF is constructed from the multiple interleaved register banks [1, 13], or the banks are used in a decentralized fashion for clustered processors [6, 21]. Our work differs from multi-banked register files, as we consider a centralized RF. For a centralized RF, there are two possible design options to reduce RF complexity – reduce the number of registers in the RF and/or reduce the number of RF ports. Both the options can significantly reduce the amount of instruction level parallelism that can be exploited. However, the relative performance degradation between the two options depends on whether the registers or the RF ports are in demand.

To be able to achieve good performance from the reduced complexity RFs, it is important to investigate techniques that increase the parallelism exploited when using them. Several techniques [8, 9, 10] are proposed to improve the parallelism from an RF with fewer registers. Unfortunately, there has been only one other effort to investigate the performance of a centralized reduced-ported RF [12]. The goal of this paper is explore the limits of RF read port reduction (i.e. the lowest number of RF read ports that can be provided) in a centralized integer register file, while still maintaining good performance. A brute-force method of port reduction may result in significant performance degradation, thus restricting the extent of port reduction. Clever port reduction techniques, on the other hand, may be able to further reduce the number of ports. Hence, in this paper, we drastically reduce the number of read ports and then propose and investigate techniques to efficiently utilize the reduced read ports provided in a centralized RF. In this paper, we focus only on the integer register file.

To alleviate the performance impact of a reduced-ported RF, our techniques (i) limit the drop in the number of operands read from the forwarding path, and (ii) increase the effective number of read ports in the reduced-ported RF. Our experiments show that our techniques significantly improve the parallelism that is exploited from reduced-ported RFs, thus allowing further reduction in the number of RF read ports. For instance, the brute force method incurs a performance impact of 25% for a two-ported RF, where our techniques incur a performance impact of only about 9% for the two-ported RF.

2 Register File Design

2.1 Brute-Force Reduced-Ported Register File (*TraRP—RF*) design

Fewer RF read ports can be provided because not all instructions have both the register operands and many operands are read from the forwarding path. The scheduler can be modified so that only those instructions that can be serviced with the reduced number of ports are issued (for instance, using bypass hints as discussed in [12]). This implementation can have a significant impact on the already complex dynamic scheduler design [28]. We use an alternative approach that performs port arbitration on the issued instructions, and requires little modifications to the scheduler. For a full-ported RF, the issued instructions read their operands in the next cycle. However, with the basic implementation of reduced number of ports, the issued instructions arbitrate for ports, and only those instructions that acquire the required number of ports proceed to the next stage. Port arbitration logic (PAL) involves port requirement analysis followed by the actual port arbitration. The PAL ensures that only the required number of ports is requested and that a consumer instruction is not executed before the producer. The instructions that acquire the

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.29

ports proceed to the following stages, whereas those that do not acquire the required ports stall and again arbitrate for the ports in the next cycle. We do not discuss detailed implementation of PAL to conserve space.

2.2 Limitations with the brute-force approach

In the basic implementation of reduced-ported RFs, at least two ports are required because instructions may read both their operands from the RF. In this implementation, the stalled instructions result in more register file reads. For instance, consider an instruction that requires one operand from the RF whereas the other operand is available from the forwarding logic. By the time the instruction acquires an RF read port, the other operand's value is written into the RF and is no longer available from the forwarding path. The stalled instructions also reduce the effective issue width of the processor (the issue slots where the instructions are stalled are frozen), delaying the issue of dependent instructions.

The brute-force approach also results in wastage of read ports, which are a precious commodity in reduced-ported RFs. Wastage of read ports results because an instruction cannot proceed till it has all the required ports. For instance, if there is just one free read port available and all the instructions arbitrating for the read ports require two operands from the RF, then the read port is wasted because it cannot be allocated to any of the instructions.

2.3 Split-instruction Reduced-Ported Register File (*SiRP—RF*) Design

As discussed in the previous section, operands of the stalled instructions can be dropped from the forwarding path by the time these instructions acquire the required RF read ports. These dropped operands are then read from the register file increasing the RF read port requirements. In this technique, we propose splitting the issued instructions between the operands whose values are available and those whose values have not yet been read from the RF. Hence, for issued instructions, the register operands for which RF ports are allocated and those that are available from the forwarding path are forwarded to the next pipe stage. In this approach, it may happen that only one register operand (out of the two) for an instruction is forwarded, splitting the instruction among the two pipeline stages. The register operands that are forwarded to the next stage obtain the values using the ports or off the forwarding path in the next cycle, irrespective of whether the instruction is split or not, and move to the following stage. The register operand that is left behind again arbitrates for a port in the next cycle. In the following cycles, the remaining operands of a "split" instruction are given priority for RF ports. When all the operands of an instruction are available, the instruction executes on the corresponding functional unit. *This technique alleviates the limitations of the TraRP—RF approach.*

Figure 1 illustrates an example of this RF design. In Figure 1, instruction *X* requires both the operands from the RF and instruction *Y* requires one operand from the RF and the other from the forwarding path. The RF is provided with a single read port. In the next cycle, *X* acquires the one port and hence *op1* of *X* and *op2* (obtained from the forwarding path) of *Y* proceed to the allocated port latches. In the next cycle, these two operands get their values, which are written into the corresponding ready-to-execute latches, and *X* again acquires the one read port and its *op2* also moves forward. In addition, another issued instruction *Z* takes the place of *X*. In the next cycle, both *Z* and *Y* again arbitrate for the ports.

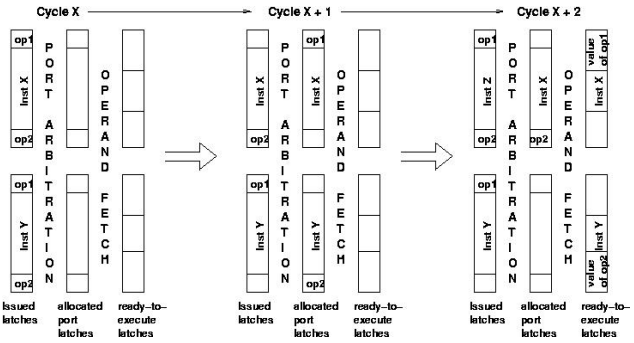


Figure 1: Example Illustrating the Working of the SiRP—RF design

2.4 Split Ported Reduced-Ported Register File (*SpoRP—RF*) Design

This technique is proposed to increase the effective number of ports in a reduced-ported RF and is implemented on top of the *SiRP—RF* design. This design exploits the observation that many of the operands used by instructions, and read from the RF, are narrow in size (require fewer bits for representation) [14]. Our experiments showed that almost about 50% of the operands read from the RF are narrow. A value is defined as narrow if it has 16 or more (out of a maximum of 32) leading bits as zeros. In this design, we partition each available RF read port into two portions. The portion that can read the upper bits of a register is called the *upper port* and the other portion is called the *lower port*. For a 32-bit register, the *upper port* reads the upper 16 bits and the *lower port* reads the lower 16 bits. Of course, this technique requires one decoder per narrow port. In this design, the instructions specifically arbitrate for ½ or 1 or 1½ or 2 ports, depending on the size of the operands. The sizes of the values written into the RF are recorded using a *size bit* for each register. *Note that it is not required to determine the size of the operands read from the forwarding path.*

This RF design will still result in the *lower ports* being utilized more often and the *upper ports* being idle. For instance, if two issued instructions both want to read a narrow operand from the RF, they both will want to acquire the lower port. Hence we replicate a narrow value in the register so that a narrow value can be read by either one of the *upper* and the *lower ports*. For instance, a value 0x000067f3 will be stored in the register as 0x67f367f3. The replication operation is performed in the dummy memory pipeline stage for non-memory instructions and in parallel to the cache output multiplexers for memory instructions.

In this design, upper or lower portions of ports may still be wasted. For instance, for a single-ported RF, if the *lower port* is already acquired by an instruction and another instruction wants to read a normal-sized operand, then that instruction can acquire both the *lower* and the *upper ports* only in the next

cycle. So, in this cycle, the *upper port* is wasted. To resolve this issue, we further split the operands between the several pipeline stages.

Figure 2(a) shows the schematic organization of a conventional register file (RF) with four read ports dedicated to the two functional units (FUs). Figure 2(b) shows the *TraRP—RF* organization with two ports for the two FUs. In this case, the output of the port is supplied to both the operands of the two FUs. Figure 2(c) shows the *SpoRP—RF* organization with one partitioned read port for the two FUs. In *SpoRP—RF*, the operand values read from the register file can either be "0L" or "0U" or "UL", where "0" specifies 16 zero bits, "L" specifies the 16 bits read using the lower port and "U" specifies the 16 bits read using the upper port. *Note that this multiplexer implementation will have minimal affect on latency because the control signals for the multiplexers are set in parallel to reading the registers*.

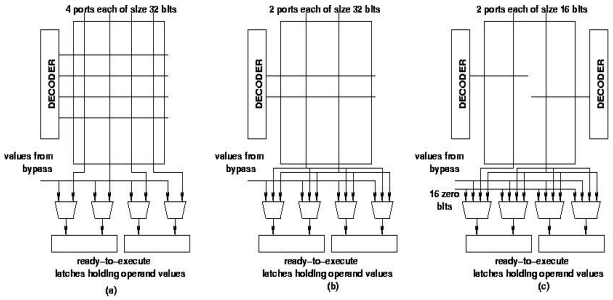


Figure 2: (a) Conventional RF with a dedicated port for each operand; (b) *TraRP—RF* with two ports; and (c) *SpoRP—RF* with one partitioned port

2.5 Arbitration of Ports

The performance impact of reduced read ports also depends on the arbitration policy used for allocating the ports. We experimented with four different arbitration policies for read port allocation. Latch position based is the simplest of the allocating policies, where priority is given to an instruction in the topmost pipeline latch, and then to the instruction in the following latch and so on. Load preference based policy is still latch position based, however, higher priority is given to load instructions, followed by ALU instructions and then by store instructions. In age based policy, an older instruction (in terms of its position in the program order) is given a higher priority during allocation of ports. The implementation of this policy will be more complex than the latch position based. We observed that the port-requirement based policy, in which the *PAL* gives the highest priority to the instruction requiring the fewest ports, is the best performing one. A tie is broken by using a latch position based priority.

3 Experimental Results

3.1 Experimental Setup

Table 1 gives experimental parameters for our baseline superscalar processor without a reduction in RF ports. For this study, we experiment with a relatively wider machine because the port requirements are anyways low for a narrow machine. In this paper, the schemes are only applied to the Integer RF accessed by the Integer FUs and the write ports are kept intact. To measure IPCs, we modify the SimpleScalar simulator [18], simulating a 32-bit PISA architecture. For benchmarks, we use a collection of 13 SPEC2000 integer and floating point benchmarks. The statistics are collected for 200M instructions after skipping the first 1B instructions. Our base pipeline has 8 frontend stages. An additional stage is introduced after issue for the *PAL*. We assume a single pipe stage for register access and two pipe stages for data memory access.

Parameter	Value	Parameter	Value
Fetch/Commit Width	8 instructions	*Floating-point FUs*	3 ALU, 1 Mul/Div
Unified Register File	128 Int/128 FP	*Integer FUs*	4 ALU, 2 AGUs, 1 Mul/Div
Integer RF	10 read / 5 write ports	*FP RF*	6 read / 3 write ports
Issue Width	5 Int/ 3 FP	*Issue Queue Size*	96 Int/64 FP
Branch Predictor	4K Gshare	*BTB Size*	4K entries, 2-way
L1 Icache	32K direct mapped 2 cycle lat	*L1 Dcache*	32K, 4-way, 2 cycle lat
Memory Latency	100 cycles first word 2 cycle inter-word	*L2 cache*	Unified 512K, 8-way, 10 cycle lat.
ROB Size	256 instructions	*Load/store buffer*	64 entries

Table 1: Default Experimental Parameters for the Base Superscalar Processor

3.2 Experimental Result

Our experiments showed that on an average more than 50% of the integer operands are obtained from the forwarding path and about 40-50% operands read from the RF are narrow in size. Interestingly, we observed that even though the percentage of narrow-sized register values produced was significantly higher than 20%, most of the narrow values produced are consumed off the forwarding path, resulting in only about 20% being read from the RF.

Next, we evaluate the performance of the different RFs in terms of instructions per cycle (IPC). We only present the results with the best performing port requirement based port allocation policy to conserve space. We observed an average of about 25% IPC drop for the two-ported *TraRP—RF* configuration with respect to the base superscalar with a 10-ported RF. Figure 3 compares the IPCs of single- and two-ported *SiRP—RF* and *SpoRP—RF* configurations with that of a two-ported *TraRP—RF*. Figure 3 shows that the parallelism exploited increases in *SiRP—RF* and *SpoRP—RF*. On an average, the two-ported *SiRP—RF* and *SpoRP—RF*

configurations perform about 5% and 10% better than *TraRP—RF*. However, the single-ported *SiRP—RF* and *SpoRP—RF* configurations perform, on an average, about 30% and 17% worse than *TraRP—RF*. The best performing configuration – two-ported SiRP—RF – still performs about 16% worse than the base 10-ported RF. This suggests that further techniques are required to reduce the performance impact, or more read ports have to be provided.

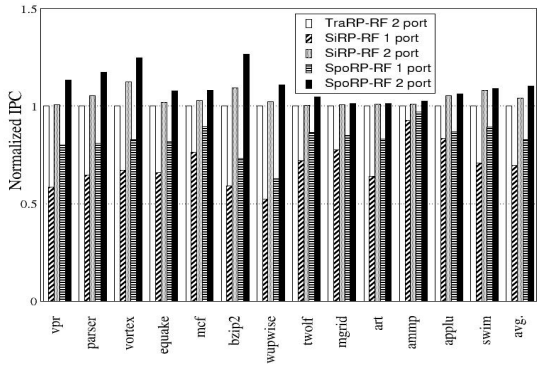


Figure 3: Normalized Performance of *SiRP—RF and SpoRP—RF* with respect to the *TraRP--RF*

4 Enhancements to Improve Performance

In the *SpoRP—RF* design, the provided ports are fully utilized, i.e. none of the *lower* or *upper ports* are idle if there are issued instructions that want to read a value from the RF. However, even the *SpoRP—RF* design still incurs delays in the issue of dependent instructions because of the waiting instructions. Delays in the issue of instructions reduce the effective issue width and further increase the RF read port requirements. Figure 4 compares the percentage of operands for integer instructions obtained from the forwarding path for one- and two-ported *SpoRP—RF* and two-ported *TraRP—RF*. Figure 4 shows that the percentage of operands obtained from the forwarding path reduces by about 15% for two-ported *SpoRP—RF* and by about 18% for two-ported *TraRP—RF*. The reduction for one-ported *SpoRP—RF* is about 20%. The enhancements proposed in this section further improve the parallelism exploited from the *SpoRP—RF* by (i) further reducing the port requirement by making more operands available from the forwarding path, and (ii) further increasing the effective number of ports available for the instructions. We discuss the results of all the enhancements in Section 4.4.

4.1 Latching the Forwarded Values

In a traditional back-end pipeline stages, a result value can be forwarded from several pipe stages. Similarly, different pipe stages may be forwarding different results. Hence, to determine the register whose value is present on a particular forwarding path, a buffer is typically used to store the register identifiers of the values on the forwarding path. As the register values pass through the backend pipeline stages, their register identifiers are written into the *RID* buffer. Once a value is available from a register, the *valid bit* for the register is set. From then onwards, all the instructions that require that value read it from the register.

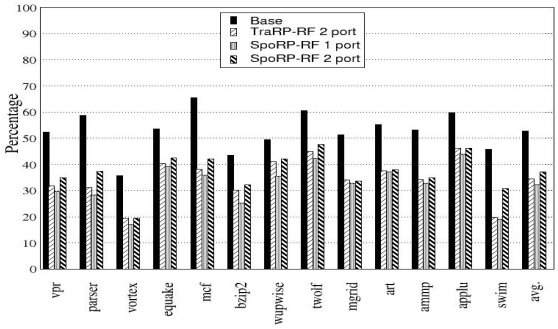


Figure 4: Percentage of operands read from the forwarding path for the integer instructions

In the *SpoRP—RF* design, many of the functional units (FUs) may not be utilized every cycle because the instructions issued to those FUs are waiting for RF read ports to read their operands. Hence, no new values are produced at the outputs of these FUs. Also instructions such as branch instructions do not produce any result. We propose a technique where the values produced by FUs are latched onto the forwarding paths till new valid values are present. Note that the latched values are still driven on the forwarding path every cycle. With this enhancement, a value is available from the forwarding path if its register identifier is still present in the *RID* buffer. In this technique, even the delayed dependent instructions may be able to read their operands from the forwarding path, increasing the number of operands read from the forwarding path. For instance, this enhancement increased the operands obtained from forwarding path from about 38% to about 44% for two-ported *SpoRP—RF*.

4.2 Servicing Multiple Instructions with a Single Port

This technique also attempts to increase the effective number of read ports. With multiple instructions vying for RF read ports, it may happen that in the same cycle, multiple instructions require a value from the same register. In all our techniques so far, different RF read ports are used to read the same register for these instructions. So with limited read ports, one instruction may have to wait for a read port even though its operand value is read from the RF. We also experimented with a technique in which the value read from the RF using a single port (which could be the lower or the upper port or the entire port) is also passed on to all instructions that require that value. To implement this technique, in parallel to vying for the RF read ports, an issued instruction also compares its source operands' register identifiers with all the other issued instructions. If an instruction (lets say *X*) does not acquire an RF read port, but the register identifier of its source operand (*op1*) matches with that of an instruction (*Y*) that has acquired a port, then *op1* of instruction *X* also proceeds to the following stages. In this case, the value read from the RF is forwarded to both the instructions.

4.3 Additional RF read port for few registers

We also experiment with a technique that increases the overall effective number of RF read ports by providing just one additional read port to only a few registers. We observed in our experiments that about 63% of the normal-sized operands

from the register file are read for the load/store instructions, especially for address computation. This is because an address value cannot be narrow. Hence, we propose a technique where a few registers are provided with an additional *unpartitioned* read port. For a centralized RF, this technique can be implemented by providing the additional port for the bottom registers. For instance, for a 128-entry RF, the registers numbered 118 to 127 can be provided with the additional port. We call such registers as *addport registers*. The *addport registers* can also be read using the other (*non-additional*) ports. The *addport registers* are allocated to instructions whose results can be used for address computation in load and store instructions. For this technique, we provide an additional bit for each instruction in the instruction cache. This bit is dynamically set for the instructions whose result can be used in address computation. Such instructions are identified when they execute for the first time. *Note that the output of this additional port is dedicated only to the AGUs used for address computation of load/store instructions*.

In this technique, when an instruction that produces a result used in address computation of load/store instructions is renamed, it is allocated an *addport register*. If no such registers are free, then it is allocated any other register. In addition, the *addport registers* can also be allocated to other instructions (whose results are not used in load/store address computation) if the remaining registers are all allocated. *However, only the load/store instructions that need to read the addport registers can vie for the additional port*. We implemented this technique by providing the additional *unpartitioned* port for 16 registers (out of the 128 registers).

4.4 Results

Figure 5 presents the performance results for the *SpoRP—RF* organization when the techniques – latching the forwarded values (*LFV*), providing an additional read port for a few registers (*ARP*), and servicing multiple instructions with a single port (*SMP*) – are applied individually, and when all the enhancements are applied together (*ALL*). As compared to two-ported *TraRP—RF*, the *ALL* enhancements improve IPC (parallelism) by about 21% for two-ported *SpoRP—RF* (a maximum of more than 50% for *bzip2*) and about 10% for one-ported *SpoRP—RF* (a maximum of more than 30% for *vortex*). All the enhancements achieve reasonable performance improvement even when they are applied individually. Note that the *ALL* enhancements improve the IPC of one-ported *SpoRP—RF* from almost 20% less than to about 10% more than the two-ported *TraRP—RF* configuration. The two-ported *SpoRP—RF* organization with ALL enhancements still performs about 9 % less than full-ported RF, but down from 25% for the two-ported TraRP—RF.

Next, we investigate the performance of the *SpoRP—RF* configuration with *ALL* enhancements as the number of ports is increased. Figure 6 presents the normalized IPC of the one-, two- and three-ported *SpoRP—RF* configuration with *ALL* enhancements and three-ported *TraRP—RF* as compared to two-ported *TraRP—RF*. As can be seen, the two-ported *SpoRP—RF* configuration performs better than three-ported *TraRP—RF*. The three-ported *SpoRP—RF* configuration is about 6% better than two-ported *SpoRP—RF*, about 11% better than three-ported *TraRP—RF,* and only about 4% less than the full-ported RF. The three-ported TraRP—RF is about 16% lower than the full-ported RF. To maintain a performance impact of less than 5%, we observed that the brute-force port reduction method requires at least five read ports.

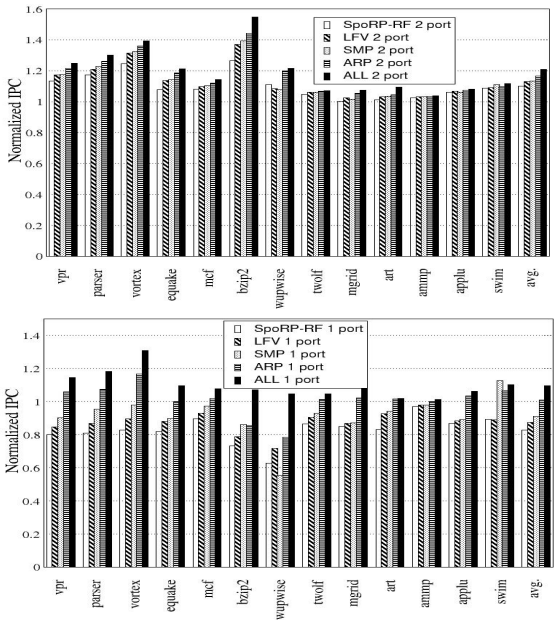


Figure 5: Normalized Performance of one- and two-ported *SpoRP—RF* (with respect to two-ported *TraRP—RF*) with and without the enhancements; using the *port requirement* based policy

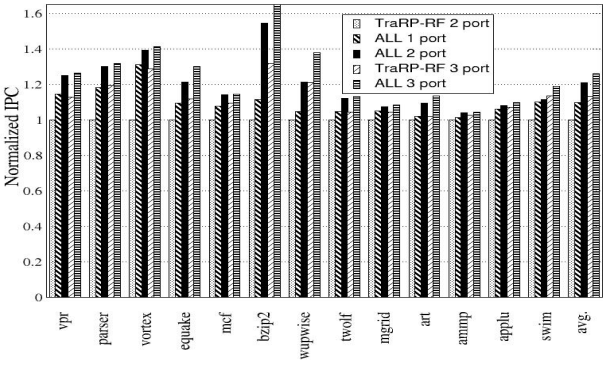


Figure 6: Normalized IPC with respect to two-ported TraRP—RF for varying number of ports

5 Related Work

Most of the low-complexity RF schemes propose banking or partitioning a register file or reducing the number of registers. The only other technique that has been proposed to reduce the number of ports a centralized RF also uses bypass hints to reduce the RF port requirements [12]. The bypass hint mechanism uses the wakeup tag search to determine bypassibility. This can significantly increase the complexity of the dynamic scheduler. In our schemes, we avoid modifications to the complex dynamic scheduler hardware.

We also propose many more schemes to further improve the parallelism with reduced RF ports.

Software controlled two-level RFs have been proposed in [15, 16, 17]. Cruz, et al [3] proposes a two-level hierarchical RF. They use a fully associative upper level RF, and run-time caching and prefetching to store the critical register values in the upper level RF, as was also proposed in [19]. The authors in [1] propose a two-level RF, which uses a *Usage Table* to record information for every physical register. The L1 RF is the conventional RF, to which a L2 RF is added. Registers are written to the L2 RF only when the number of physical registers falls below a pre-set threshold.

Partitioning or replication of monolithic register files has been proposed in the context of clustered processors [20, 21, 6, 22]. These organizations reduce the porting requirements per cluster while adding inter-cluster communication. Partitioned register files have also been proposed in [23] for a VLIW processor. Balasubramonian, et al [1] also propose reducing the ports per bank by modifying the scheduler.

Researchers have exploited the inefficiencies in register usage to reduce the number of registers in three major ways. One set of solutions delays the actual allocation of physical registers until the time that the result is written back [e.g. 24, 25]. The second set of solutions reduces the number of registers through the use of register sharing [e.g. 26]. The third set of techniques aim at reducing the register file pressure by using the early deallocation of physical registers [e.g. 8].

6 Summary and Conclusions

Heavily ported large register files are provided to achieve good performance for modern processors with large issue queue sizes and issue widths. Such a large register file has a significant impact on the processor clock cycle time and power consumption. The register file size can be reduced by reducing the number of read ports that are provided. However, a naïve reduction of the register file read ports results in significant performance degradation, thus limiting the extent of port reduction. In this paper, we propose and investigate various techniques to improve the parallelism that can be exploited from a reduced-ported RF, thus enabling further reduction in ports.

We propose a unique RF where the operands are split among various pipeline stages and the ports are partitioned so that a single port can read two narrow operands from different registers. These techniques reduce the number of operands dropped from the forwarding path and increase the effective number of available read ports. We proposed further enhancements to this technique, such as latching the values in the forwarding paths, forwarding a register value read from the register file to all the instructions requiring that value, and providing an additional read port for a few registers. With these techniques, the parallelism exploited from a reduced-ported RF is significantly increased. These techniques allow reducing the number of integer RF read ports from 10 to three, while maintaining an average performance impact of less than 5%.

References

[1] R. Balasubramonian, et. al., Reducing the Complexity of the Register File in Dynamic Superscalar Processors, Proc. Micro-34, 2001.

[2] K. Farkas, et. al., Register File Considerations in Dynamically Scheduled Processors, Proc. HPCA, 1996.

[3] J.-L. Cruz, et. al., Multiple-Banked Register File Architectures, Proc. ISCA-27, 2000.

[4] D. M. Tullsen et. al., Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor, Proc. ISCA, 1996.

[5] V. Zyuban and P. Kogge, Optimization of High-Performance Superscalar Architectures for Energy Efficiency,
Proc. ISLPED, 2000.

[6] R. Kessler, The Alpha 21264 Microprocessor, IEEE Micro, March/April 1999.

[7] K. C. Yeager, The MIPS R10000 Superscalar Microprocessor, IEEE Micro, April 1996.

[8] Martinez, J., Renau, J., Huang, M., Prvulovich, M., Torrellas, J., "Cherry: Checkpointed Early Resource Recycling in Out-of-order Microprocessors", in *Proc. of MICRO-35*, 2002.

[9] Monreal, et.al., "Delaying Register Allocation Through Virtual-Physical Registers", in *Proc. of MICRO-32*, 1999.

[10] Ergin O., et.al., "Register Packing: Exploiting Narrow-Width Operands for Reducing Register File Pressure", in Proc. Of .*MICRO*, 2004.

[11] P. Shivakumar and N. Jouppi, CACTI 3.0: An Integrated Cache timing, Power, and Area Model, *Technical Report, DEC Western Lab*, 2002.

[12] Park, I., Powell, M., Vijaykumar, T., "Reducing Register Ports for Higher Speed and Lower Energy", in *Proc. of MICRO-35*, 2002.

[13] Tseng, J., Asanovic, K., "Banked Multiported Register Files for High Frequency Superscalar Microprocessors", in *Proc. of ISCA-30*, 2003.

[14] D. Brooks and M. Martonosi, Dynamically Exploiting Narrow Width Operands to Improve Processor Power and Performance, Proc. HPCA, 1999.

[15] J. Zalamea, et. al.,Two-Level Hierarchical Register File Organization for VLIW Processors, Proc. Micro-33, 2000.

[16] R. Russell, The Cray-1 Computer System, Readings in Computer Architecture, 2000.

[17] J. Swensen and Y. Patt, Hierarchical Registers for Scientific Computers, Proc. ICS, 1988.

[18] D. Burger and T. M. Austin, The SimpleScalar Tool Set, Version 2.0, Computer Arch. News, June 1997.

[19] R. Yung and N. Wilhelm, Caching Processor General Regsiters, Proc. ICCD, 1995.

[20] A. Capitanio, et. al., Partitioned Register Files for VLIWs: A Preliminary Analysis of Trade-offs, Proc. Micro-25, 1992.

[21] K. Farkas, et. al., The Multicluster Architecture: Reducing Cycle Time Through Partitioning, Proc. ISCA-24, 1997.

[22] S. Rixner, et. al., Register Organization for Media Processing, Proc. HPCA, 2000.

[23] J. Janssen and H. Corporaal, Partitioned Register File for TTAs, Proc. Micro-28, 1995.

[24] Gonzalez, A., Gonzalez, J., Valero, M., "Virtual-Physical Registers", in *Proc. of HPCA-4*, 1998.

[25] Wallase, S., Bagherzadeh, N., "A Scalable Register File Architecture for Dynamically Scheduled Processors", in *Proc. PACT-5, 1996*.

[26] Srinivasan S., et.al., "Continual Flow Pipelines", in *Proc. of ASPLOS* 2004.

[27] G. Loh, "Exploiting data-width locality to increase superscalar execution bandwidth," *Proc. Micro-35*, 2002.

[28] S. Palacharla, et al., "Complexity-Effective Superscalar Processors," *Proc. ISCA*, 1997.

2009 22nd International Conference on VLSI Design

Temperature Aware Scheduling for Embedded Processors

Ramkumar Jayaseelan Tulika Mitra
Department of Computer Science
National University of Singapore
{ramkumar,tulika}@comp.nus.edu.sg

Abstract

Power density has been increasing at an alarming rate in recent processor generations resulting in high on-chip temperature. Higher temperature results in poor reliability and increased leakage current. In this paper, we propose a temperature aware scheduling technique in the context of embedded multi-tasking systems. We observe that there is a high variability in the thermal properties of different embedded applications. We design temperature-aware scheduling (TAS) scheme that exploits this variability to maintain the system temperature below a desired level while satisfying a number of requirements such as throughput, fairness and real time constraints. Moreover, TAS enables exploration of the tradeoffs between throughput and fairness in temperature-constrained systems. Compared against standard schedulers with reactive hardware-level thermal management, TAS provides better throughput with negligible impact on fairness.

1 Introduction

Decreasing feature sizes and increased complexity have resulted in very high power density in modern processors. The power dissipated is converted into heat and the processors are pushing the limits of packaging and cooling solutions [2]. The problem is prominent in the embedded domain where mobility and size constraints do not warrant for elaborate cooling mechanisms such as fan and heat sink. Increased operating temperature affects reliability. Moreover, leakage power increases exponentially with operating temperature. Increasing leakage power can further raise the temperature resulting in a thermal runaway [14]. Hence, there is a need to control temperature at all levels of system design.

Designing the thermal package for the worst-case power dissipation has become prohibitively expensive. Instead, packages are designed for worst typical behavior and rely on Dynamic Thermal Management (DTM) techniques to control the temperature. Many hardware and software-based DTM techniques have been proposed recently [3, 6, 7, 12, 13, 15]. Most of them, with the exception of [15], are reactive in nature. They invoke appropriate mechanism to cool down the system when the temperature reaches a threshold. Cooling down typically involves techniques such as global clock gating or dynamic voltage scaling that degrade the system throughput. As such, reactive techniques do not have control over the system behavior that leads to the threshold temperature. In contrast, predictive techniques [15] anticipate the future thermal behavior and take appropriate measures to avoid thermal emergencies.

Recently there has been significant interest in thermal management in embedded systems. Majority of the thermal management techniques proposed in the context of embedded systems employ static or design-time approaches [18, 8]. Recently, there have been a huge proliferation of multimedia-dominated personal embedded devices, such as PDAs, cell phones, portable audio/video players etc. These high-performance devices provide multiple functionalities, include operating system, and support concurrency through multi-tasking. For example, it is quite common for a user to run a audio clip decoder on his/her PDA, while at the same time browse the web or perform word processing.

In this scenario, where the embedded device runs a mix of soft real-time applications (e.g., audio decider) and best effort tasks (e.g., word processing) chosen at runtime by the user, the static or design-time thermal management techniques are no longer effective. Hence we propose a dynamic predictive temperature-aware scheduling (TAS) strategy for embedded multi-tasking systems consisting of a mix of soft real-time applications and best-effort applications. Our scheme exploits the variation between the temperature profiles [14, 15] of different applications to maintain the temperature below the threshold while maintaining high throughput. The basic idea of the scheduler is to differentiate among the tasks based on their predicted thermal profiles and penalize the hot tasks if necessary to maintain high system throughput. Moreover, we show that a simple parameter can control the tradeoffs between throughput and fairness of the TAS scheme. We evaluate our scheduling scheme on a ARM cortex A8 like embedded processor model. We observe that TAS can achieve higher throughput while maintaining QoS guarantees for soft real-time tasks with marginal loss in fairness among the best-effort tasks

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.42

compared to traditional schedulers employed in conjunction with DTM techniques.

2 Related Work

Dynamic Thermal Management (DTM) mechanisms can be hardware or software-based. In both hardware and software based schemes, the temperature sensors on the processor are continuously monitored and when this temperature exceeds a predefined threshold, appropriate mechanisms are invoked to reduce the temperature. Different thermal management schemes differ in the mechanisms that they employ to maintain the temperature below the threshold. Hardware-based DTM mechanisms include chip wide mechanisms such as dynamic voltage scaling [14], global clock gating [6] and ILP-based techniques [3, 15, 7].

Recently there has been widespread interest in software and system level thermal management schemes. Rohou et al. [12] propose a software-based technique where hot tasks are not allocated processing time when the system reaches a threshold. Kumar et al. [9] propose a similar software-based technique that examines the interplay between hardware and software thermal management. Reactive thermal management for hard real time systems has been discussed in [16]. While the above mentioned schemes are reactive in nature, predictive schemes for thermal management have also been proposed. Predictive techniques anticipate the future thermal behavior and take appropriate measures to avoid thermal emergencies. A predictive scheme for multimedia applications has been proposed in [15]. It exploits the frame-based nature of media applications to predict the temperature of the next frame based on the offline profiles of execution of similar frames. It is not immediately clear how this technique can be extended to predict the temperature of arbitrary applications.

In this paper we present a temperature aware scheduling strategy in the context of high performance embedded multi-tasking systems. Unlike previous reactive and predictive thermal management schemes, our strategy can optimize the system performance while maintaining other requirements such as real time constraints and fairness. In the next section we present our temperature aware scheduling framework and the temperature model.

3 Temperature Aware Scheduling Framework and Thermal Model

The goal of our thermal management framework is to maintain the temperature below the threshold while satisfying a variety of system level scheduling requirements such as throughput, fairness and real time constraints. Power consumption and hence the the thermal profiles vary between different tasks. Given a set of tasks with varying thermal profiles, the temperature can be controlled by varying the relative amount of time for which hot tasks and cold tasks execute. Our thermal aware scheduling framework exploits this observation in conjunction with voltage scaling to maintain the processor temperature below the maximum specified temperature.

The framework consists of a predictive thermal model and the temperature aware scheduler. The predictive thermal model is used to characterize the thermal properties of each task and also predict the change in temperature when a task executes starting from an initial temperature. The temperature aware scheduler uses the thermal properties of the tasks from the model and the task execution time requirements to determine the time for which each task executes. Our framework consists of both real time and best effort tasks. The scheduler ensures that real time tasks meet the deadline and for best effort tasks, the goal is to maximize the throughput while maintaining a user supplied level of fairness. For maintaining fairness as well as to maintain real time constraints, our scheduler exploits dynamic voltage and frequency scaling. Next we present our thermal model and Section 4 presents our temperature aware scheduler.

3.1 Thermal Model

In this section, we present a thermal model to predict the processor temperature at any point during the execution of a specific application. We use a predictive thermal model which models the temperature profile of a given application as an exponential function of the form [18]

$$T(t) = T_s - (T_s - T_{init}) \times e^{-Kt} \tag{1}$$

where T_s is the steady state temperature of the application which is defined as the temperature the processor would reach if the application executes indefinitely, $T(t)$ is the temperature of the processor after the application executes for t time units, T_{init} is the initial temperature, and K is a processor specific and application independent constant.

The value of the application independent processor specific constant K can be determined by fitting the observed temperature profiles for different applications into the exponential function. This process is done offline and the computed value of K is used in the predictive thermal model. For our processor model[1] we compute the value of $K = 0.00472$.

The steady state temperature of an application can be determined online by observing the temperature change over a period of time when the application executes and rearranging Equation 1.

$$T_s = \frac{T_c - T_{init} \times e^{-Kc}}{(1 - e^{-Kc})} \tag{2}$$

where T_s is the steady state temperature of the application, T_c is the temperature after the application executes for c

[1]The details of the processor model is presented in the experimental section

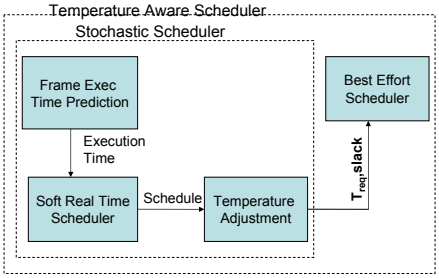


Figure 1. Temperature Aware Scheduling Policy

time units, T_{init} is the initial temperature before the application starts execution, and K is the application independent processor specific constant that is computed offline.

Once the steady state temperature of the application is known, Equation 1 can be used to predict the change in temperature when this application executes starting from any initial temperature.

Accuracy of the Prediction Model In order to check the accuracy of the predictive thermal model, we run a set of embedded benchmarks on a ARM Cortex A8 [1] like embedded processor model and observe the temperature profiles of the processor. We compare the observed temperature from these runs with the temperatures predicted from the model. For each application, we obtained the temperature curve from HotSpot with a sampling frequency of 1 millisecond. We also applied our model to predict the temperature variations and compared the temperature curves from the model and from HotSpot. For each benchmark, we measured the peak error that this the temperature difference at the point at which the predicted and the observed curves diverge the most. The maximum peak error is 0.6°C and the average peak error is 0.14°C across all the benchmarks. Hence our model provides sufficient accuracy for software based thermal management. In the next section we present our temperature aware scheduler.

4 Temperature Aware Scheduling

An overview of our thermal aware scheduler is shown in Figure 1. Our system consists of soft real-time (multimedia) and best-effort tasks. Our soft real time tasks comprise of periodic multimedia tasks that release a job per period, e.g., decoding a video frame every 30 ms. We employ a hierarchical scheduling structure typically used in multi-media systems [5, 10, 11].

The thermal aware scheduler consists of two subschedulers to handle soft real time tasks and best effort tasks. The execution requirements for the next frame is predicted using a frame execution time predictor. We employ the histogram based method for execution time rediction proposed in [17] for its accuracy and ease of implementation. The predicted frame execution times are given as input to the soft real time scheduler which schedules the soft real time tasks. We employ a simple static priority soft real time scheduler in our scheme where the audio decoding task has a higher priority than the video decoding task.

Our thermal aware scheduler has an additional thermal adjustment phase. This phase takes the soft real time schedule and the predicted frame execution time requirements as input and has two main parts (i) Ensure that the soft real time task remains below the threshold frequency/voltage scaling the soft real time tasks if necessary (ii) Compute the starting temperature (T_{req}) for the next period so that the temperature of the soft real time tasks remain below the threshold. The slack and the required temperature (T_{req}) is provided as input to the best effort task scheduler. We employ a modified version of a round robin scheduler as our best effort scheduler. Our best effort scheduler classifies tasks into hot and cold tasks and controls temperature by changing the execution time provided to the hot and cold tasks. Next we present the thermal adjustment phase employed in conjunction with the soft real time scheduler.

4.1 Thermal Adjustment Phase

The thermal adjustment phase takes the frame execution time prediction and soft real time schedule as input and performs the following tasks (i) Compute T_{req} the starting temperature for the next set of soft real time task such that temperature during the next invocation remains below the threshold (ii) Ensure that current set of soft real time tasks maintain the temperature below the threshold , voltage scaling/ dropping frames if necessary. We explain a case with two real time tasks R1 and R2 in the remainder of this discussion but the scheme can be extended to multiple soft real time tasks

4.1.1 Computing T_{req}

T_{req} is defined as the maximum initial temperature such that the execution of the next real-time task(s) is guaranteed not to exceed T_{max}. As the execution time and period of real-time tasks are known, it is easy to compute T_{req}. For example, suppose the system will execute two soft real-time tasks for $t1$ and $t2$ time units in the near future with steady state temperatures T_{s1} and T_{s2}. Then T_{req} can be determined by using Equation 1 as

$$T_{req} = T_{s1} - (T_{s1} - T_{s2})e^{Kt1} + (T_{s2} - T_{max})e^{K(t1+t2)} \quad (3)$$

This can be easily extended to multiple soft real-time tasks.

4.1.2 Voltage Scaling Soft Real Time Tasks

This phase also checks if the execution of the soft real time tasks maintains the temperature below the threshold using the model. For instance if the present temperature of the system is T_{init} and there are two soft real-time tasks for $t1$

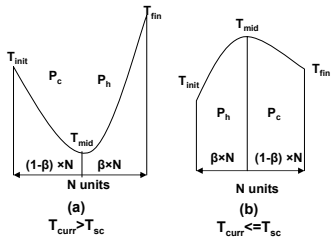


Figure 2. CPU Share between Hot and Cold Tasks

and $t2$ time units in the near future with steady state temperatures T_{s1} and T_{s2}. The temperature at the end of Task 1 and Task 2 are given by

$$T_1 = T_{s1} - (T_{s1} - T_{init})e^{-Kt1} \quad (4)$$

$$T_2 = T_{s2} - (T_{s2} - T_1)e^{-Kt2} \quad (5)$$

If $T_1 < T_{max}$ and $T_2 < T_{max}$ then the phase computes the slack and presents the slack and T_{req} to the best effort scheduler. If either one of tasks exceed the threshold then the then the corresponding tasks's frequency is lowered to the next lower frequency level. After lowering the frequency the temperature and deadline constraints are verified.If either constraints are not met then the frame is dropped. At the end of the temperature adjustment phase, a feasible soft real time schedule with frequency levels for each task as well as the corresponding slack and T_{req} values are computed. The slack and T_{req} values are sent to the best effort scheduler which uses it for scheduling the best effort tasks. Next we present our best effort scheduler.

4.2 Best Effort Scheduler

We first categorize the best-effort tasks into **hot tasks** and **cold tasks**. P_c is a cold task if its steady state temperature is below T_{req} as it would cool down the system. Similarly, P_h is a hot task if its steady state temperature is above T_{req} as it may heat up the system beyond T_{req}.

We observe that a schedule alternating between hot and cold tasks provides a good solution. The scheduler considers a pair of tasks (one hot and one cold) at a time. The problem now boils down to dividing up CPU share between these two tasks so as to keep the temperature below T_{max} and the temperature at the end of the schedule is below T_{req}. If the set of best effort tasks consists of only hot tasks then our best effort scheduler uses a voltage scaled version of the hot task as the cold task.

4.3 CPU Share between a Hot and Cold Task

Given (1) current temperature T_{curr}, (2) a hot task (P_h) with steady state temperature T_{sh}, and (3) a cold task (P_c) with steady state temperature T_{sc}, the goal of the scheduler is to allocate N time units between P_h and P_c so as to maintain the system temperature below T_{max} and the temperature at the end of the schedule is T_{req}. In particular, we determine the maximum share $0 \leq \beta \leq 1$ that can be allocated to the hot task (i.e., it executes for βN) while maintaining the system temperature below T_{max} and temperature at the end of the schedule is less than T_{req}.

Case 1: $T_{curr} \geq T_{sc}$ In this case, the cold task should be scheduled first to cool down the system and maximize the share for the hot task. Figure 2(a) shows the temperature curve over N time units. T_{mid} is the temperature after executing the cold task and T_{fin} is the final temperature.

$$T_{fin} = T_{sh} - (T_{sh} - T_{mid}) \times e^{-K\beta N} \quad (6)$$

$$T_{mid} = T_{sc} - (T_{sc} - T_{curr}) \times e^{-K(1-\beta)N} \quad (7)$$

Clearly, the temperature constraints are satisfied if $T_{fin} < T_{req}$. Hence, the maximum value of β can be obtained by substituting $T_{fin} = T_{req}$ and solving for β

$$\beta = \frac{\ln(\frac{C2}{C1+C3\times e^{-KN}})}{K \times N} \quad (8)$$

where $C1 = T_{sh} - T_{req}$; $C2 = T_{sh} - T_{sc}$; $C3 = T_{curr} - T_{sc}$

Case 2: $T_{curr} < T_{sc}$ In this case the maximum share for the hot task is obtained when it is scheduled first. This scenario is shown in Figure 2(b). Here the temperature is guaranteed to be below T_{max} if $T_{mid} <= T_{max}$ and the final temperature constraint is satisfied if $T_{fin} <= T_{req}$. So the value of β can be obtained from

$$T_{mid} = T_{sh} - (T_{sh} - T_{curr}) \times e^{-K\beta N} \quad (9)$$

$$T_{fin} = T_{sc} - (T_{sc} - T_{mid}) \times e^{-K(1-\beta)N} \quad (10)$$

Substituting $T_{mid} = T_{max}$ and solving for β

$$\beta_1 = \frac{\ln(\frac{T_{sh}-T_{curr}}{T_{sh}-T_{max}})}{kN} \quad (11)$$

Using $T_{fin} = T_{req}$ we get

$$\beta_2 = \frac{\ln(\frac{(T_{req}-T_{sc})+(T_{sh}-T_{curr})}{T_{sh}-T_{sc}})}{kN} \quad (12)$$

$$\beta = min(\beta_1, \beta_2) \quad (13)$$

Best Effort Scheduling Policy The run queue consisting of the ready tasks is split into two queues corresponding to the hot and cold tasks, respectively. The scheduler also keeps track of the CPU share given to each task so far. Whenever the scheduler is invoked, it selects the task with least share in the hot queue (P_h) and the task with the least share in the cold queue (P_c). Let N be the scheduling unit. The maximum share, β, that can be allocated to P_h in the next $2N$ time units is determined using Eqn 8 or Eqn 13. Our best effort scheduler examines the CPU share allocated to both the tasks and tries to maintain fairness while ensuring that the hot task gets no more than $\beta \times 2N$ time units.

Enforcing Fairness The scheduling scheme discussed earlier cannot ensure fairness as it gives higher preference to cold tasks in trying to keep the system temperature below T_{max}. To obtain a tradeoff between throughput and fairness, we employ selective voltage scaling for the hot tasks in conjunction with our thermal-aware scheduler. We assume that the processor supports two voltage levels V_{min} and V_{max} with corresponding frequencies f_{min} and f_{max}. To ensure fairness, we define **minimum share** s_{min} for any task. If the current share of a hot task is below s_{min}, then its voltage scaled version is transferred to the cold queue. The parameter s_{min} represents the tradeoff between fairness and throughput. Given an aggressive value of s_{min}, the system spends most of its time in voltage scaled mode thus reducing throughput. A smaller value of s_{min}, in contrast, may lead to unfairness towards the hot tasks.

5 Experimental Evaluation

Setup We use SimpleScalar 3.0 architectural simulator with configurations similar to a ARM Cortex A8 Embedded Processor [1]. The processor model is a in-order dual issue processor with 32 KB instruction/data caches, 512 entry branch miss-prediction buffer and a 13 entry branch miss-prediction pipeline. The temperature values are obtained using HotSpot-3.0 [14], an architecture-level thermal simulator working in conjunction with Wattch [4], an architecture-level power simulator. We use Wattch's linear scaling to obtain the power consumption at 1.2 V and 1.5 Ghz [1] and for voltage scaling we use a lower operating frequency of 800 Mhz which we find sufficient to remove all temperature violations [13]. We use a thermal resistance of $1.83^oC/W$ and a thermal capacitance of $112.4mJ/^oC$ and an ambient temperature of 40^oC. We assume that the temperature should not exceed 80°C based on the cooling solution [9]. The benchmarks selected from MiBench, MediaBench and EEMBC benchmark suites have steady state temperatures in the range $63.65^oC - 88.5^oC$. We have tasks with low (`patricia`, `gs`, `apcm`), medium (`jpeg`, `mpeg`, `mp3`, `blowfish`,`crc`,) and high (`rijndael`, `sha`, `susan`) thermal profile. We create eight task sets using different combinations of these benchmarks as shown in Table 1. Each task set contains applications with varying thermal characteristics. Four of these task sets have soft real-time applications (`mpeg` and `mp3`) while the other four have only best-effort applications. We assume a frame rate of 30 ms for `mpeg` and 26 ms for `mp3`. Tasks sets containing real-time applications are simulated till 450 video or audio frames complete decoding. Tasks sets containing only best effort applications are simulated for a total of 500 time slices (each time slice = 20ms). Of these task sets, S8 consists of only hot applications and hence in this case our scheduler performs scheduling by using a voltage scaled version of available hot tasks as cold tasks.

	Soft Real-Time	Best Effort
S1	`mpeg, mp3`	`sha, jpeg, adpcm, crc`
S2	`mpeg, mp3`	`rijndael, susan, patricia, gs`
S3	`mpeg, mp3`	`susan, jpeg, blowfish, gs`
S4	`mpeg, mp3`	`rijndael, sha, adpcm, patricia`
S5		`jpeg, susan, crc, gs`
S6		`sha, rijndael, crc, gs`
S7		`sha, susan, jpeg, patricia`
S8		`susan, rijndael, jpeg, blowfish`

Table 1. Composition of Task Sets

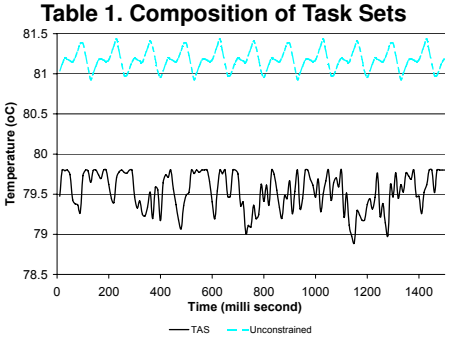


Figure 3. Temperature Profile for TAS

Traditional Thermal Management: Our temperature-aware scheduler (**TAS**) maintains the temperature below the threshold by appropriately scheduling the best-effort tasks. We compare it against a standard round robin (**RR**) scheduler for the best-effort tasks with a time slice of 20ms. The RR scheduler cannot guarantee that the temperature will not exceed the threshold and hence dynamic thermal management (DTM) techniques need to be engaged. We employ two popular DTM techniques in conjunction with RR scheduler: dynamic voltage scaling (**DVS**) and global clock gating (**CG**). *DVS* lowers the voltage/frequency of the processor whenever the system hits the threshold temperature. In case of *CG*, the processor global clock is gated (i.e., the processor remains idle). Once a DTM mechanism is engaged, the system begins to cool down. The normal operating voltage and frequency are resumed once the system temperature goes sufficiently below the maximum temperature. We implement a binary DVS scheme [13] that is shown to be performing as well as multi-level DVS schemes.

Maintaining Temperature : Figure 3 shows the temperature profiles for our thermal aware scheduling scheme as well as the temperature profile if no DTM scheme is present. We observe that *TAS* is able to keep the temperature below T_{max} for all the task sets by changing either the share of processing time given to hot and cold tasks or by appropriately scaling the operating frequency.

Throughput: Let t_{min} and t_{max} be the time the system spends in lower (800 MHz) and higher frequency (1.5GHz) for *DVS* and *TAS*. Note that *TAS* does not use low voltage as a consequence of hitting the threshold temperature. Instead, it selectively lowers the voltage of hot tasks to ensure fairness. For both cases, the throughput is defined as

$$Throughput = \frac{t_{max} + 0.53 \times t_{min}}{t_{max} + t_{min}} \quad (14)$$

Task Set	Throughput				Fairness			
	TAS(0)	TAS(0.2)	DVS	CG	TAS(0)	TAS(0.2)	DVS	CG
S1	1.0	0.96	0.91	0.82	0.94	0.97	0.98	0.94
S2	1.0	0.92	0.89	0.84	0.89	0.96	0.98	0.92
S3	1.0	0.89	0.86	0.82	0.84	0.94	0.98	0.94
S4	1.0	0.98	0.94	0.88	0.91	0.96	0.97	0.90
S5	1.0	1.0	0.95	0.87	0.97	0.97	0.98	0.91
S6	1.0	0.93	0.92	0.89	0.88	0.96	0.98	0.96
S7	1.0	0.96	0.90	0.81	0.89	0.92	0.98	0.90
S8	0.82	0.82	0.84	0.73	0.99	0.99	0.99	0.96

Table 2. Throughput and Fairness of Thermal-aware Scheduler (TAS) with $s_{min} = 0$, $s_{min} = 0.2$ and DTM Schemes

Let t_{idle} be the idle time under global clock gating. Then

$$Throughput_{CG} = \frac{t_{max}}{t_{max} + t_{idle}} \quad (15)$$

Table 2 shows the throughput for *TAS*, *DVS* and *CG*. We use two versions of *TAS* scheme with different values of s_{min} that controls fairness (see Section 4.3). The first version ($s_{min} = 0$) maximizes the throughput while ignoring fairness. The second version ($s_{min} = 0.2$) introduces a voltage-scaled version of a hot task when its share drops below 0.2.

The results shows that *TAS* performs better than clock gating in all cases. It performs better than *DVS* in cases where there is at least one cold task (S1–S7). In the case where only hot tasks are available (S8), *TAS* gives almost the same throughput as *DVS*. As expected, the throughput of *TAS* drops as the minimum expected share (s_{min}) is increased from 0 to 0.2.

Fairness: The share of a best effort task P_i is given by

$$s(i) = \frac{t_{max}(i) + 0.53 \times t_{min}(i)}{\Sigma_{i=1}^{Q} t_{max}(i) + 0.53 \times \Sigma_{i=1}^{Q} t_{min}(i)} \quad (16)$$

$$s_{CG}(i) = \frac{t_{max}(i)}{\Sigma_{i=1}^{Q} t_{max}(i)} \quad (17)$$

where $t_{max}(i)$ and $t_{min}(i)$ are the amount of time task P_i spends at maximum and minimum voltage and Q is the number of best-effort tasks. Based on the share given to each best effort application, our metric for fairness is

$$F = 1 - \frac{\Sigma_{i=1}^{Q} |s_{fair} - s(i)|}{Q} \quad (18)$$

where s_{fair} is the expected fair share.

Table 2 shows the fairness metric for *TAS*, *CG*, and *DVS*. *TAS* with $s_{min} = 0$ maximizes the throughput with lower fairness than DVS. However, if we use $s_{min} = 0.2$, then the fairness improves and becomes comparable to *DVS* with higher throughput than DVS . As the scheduler tries to be more fair (higher s_{min}), the throughput drops. Thus our predictive scheme can provide a tradeoff between system throughput and fairness when operating under thermal constraint. A reactive DVS scheme, on the other hand, only operates at one of these points.

6 Conclusion

In this paper, we have proposed a simple thermal model to predict the temperature when an arbitrary application starts execution from an initial temperature. Based on our model, we have designed and experimentally evaluated a thermal-aware scheduling strategy for multi-tasking embedded systems. Our thermal-aware scheduler can provide better throughput in comparison to DTM employed in conjunction with conventional schedulers.

7 Acknowledgements

This work is partially supported by NUS research project R-252- 000-292-112.

References

[1] ARM Cortex A8 Processor. http://www.arm.com/products/CPUs/ARM_Cortex-A8.html.

[2] S. Borkar. Design Challenges of Technology Scaling. *IEEE Micro*, 19(4), 1999.

[3] D. Brooks and M. Martonosi. Dynamic Thermal Management for High-Performance Microprocessors. In *HPCA*, 2001.

[4] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-level Power Analysis and Optimizations. In *ISCA*, 2000.

[5] P. Goyal, X. Guo, and H. M. Vin. A Hierarchical CPU Scheduler for Multimedia Operating Systems. In *OSDI*, 1996.

[6] S. Gunther et al. Managing the Impact of Increasing Microprocessor Power Consumption. *Intel Technology Journal*, 2001.

[7] S. Heo, K. Barr, and K. Asanovic. Reducing Power Density through Activity Migration. In *ISLPED*, 2003.

[8] W-L. Hung et al. Thermal-aware task allocation and scheduling for embedded systems. In *DATE*, 2005.

[9] A. Kumar et al. HybDTM: A Coordinated Hardware-Software Approach for Dynamic Thermal Management. In *DAC*, 2006.

[10] J. Nieh and M. S. Lam. A smart scheduler for multimedia applications. *ACM Trans. Comput. Syst.*, 2003.

[11] J. Regehr and J. A. Stankovic. Hls: A framework for composing soft real-time schedulers. In *RTSS*, 2001.

[12] E. Rohou and M. Smith. Dynamically Managing Processor Temperature and Power. In *Workshop on Feedback-Directed Optimization*, 1999.

[13] K. Skadron. Hybrid Architectural Dynamic Thermal Management. In *DATE*, 2004.

[14] K. Skadron et al. Temperature-aware Microarchitecture: Modeling and Implementation. *ACM TACO*, 1(1), 2004.

[15] J. Srinivasan and S. V. Adve. Predictive Dynamic Thermal Management for Multimedia Applications. In *ICS*, 2003.

[16] S. Wang and R. Bettati. Reactive Speed Control in Temperature-Constrained Real-Time Systems. In *ECRTS*, 2006.

[17] W. Yuan and K. Nahrstedt. Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems. In *SOSP*, 2003.

[18] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *ICCAD*, 2007.

SACR: Scheduling-Aware Cache Reconfiguration for Real-Time Embedded Systems

Weixun Wang and Prabhat Mishra
Department of Computer and Information Science and Engineering
University of Florida, Gainesville, FL
wewang@cise.ufl.edu, prabhat@cise.ufl.edu

Ann Gordon-Ross
Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL
ann@ece.ufl.edu

Abstract

Dynamic reconfiguration techniques are widely used for efficient system optimization. Dynamic cache reconfiguration is a promising approach for reducing energy consumption as well as for improving overall system performance. It is a major challenge to introduce cache reconfiguration into real-time embedded systems since dynamic analysis may adversely affect tasks with real-time constraints. This paper presents a novel approach for implementing cache reconfiguration in soft real-time systems by efficiently leveraging static analysis during execution to both minimize energy and maximize performance. To the best of our knowledge, this is the first attempt to integrate dynamic cache reconfiguration in real-time scheduling techniques. Our experimental results using a wide variety of applications have demonstrated that our approach can significantly (up to 74%) reduce the overall energy consumption of the cache hierarchy in soft real-time systems.

1. Introduction

Design and optimization of real-time embedded systems have been widely studied over the last few decades. These systems require unique design considerations due to time constraints placed on the tasks. Hard real-time system tasks have deadlines and tasks must complete execution by their deadlines in order to ensure correct system behavior. Due to these stringent constraints, real-time scheduling algorithms must perform task *schedulability analysis* based on task attributes such as priorities, periods, and deadlines [4][12]. A task set is considered *schedulable* if there exists a schedule that satisfies all timing constraints. As embedded systems become ubiquitous, real-time systems with soft timing constraints (missing certain deadlines are acceptable) are gaining widespread acceptance. Soft real-time systems can be found everywhere including gaming, multimedia, and housekeeping devices. Tasks in these systems remain effective even if their deadlines are not guaranteed to be met. Minor deadline misses may result in temporary service or quality degradation, but will not lead to incorrect behavior.

One of the most important optimizations in real-time embedded systems is energy consumption reduction since most of these systems are battery-operated devices. Processor idle time (also known as slack time) provides a unique opportunity to reduce the overall energy consumption by putting the system into sleep mode using Dynamic Power Management (DPM) techniques [2]. Alternatively, Dynamic Voltage Scaling (DVS) [8] methods can be used to reduce the clock frequency such that the tasks execute slowly but do not violate their deadlines [10][16].

In recent years, reconfigurable computing provides the unique ability to *tune* the system during runtime (*dynamically reconfigure*) to meet optimization goals by changing *tunable* system parameters. The primary aspect of reconfigurable computing research emphasizes tuning algorithms, which determine *how* and *when* to dynamically reconfigure tunable parameters to achieve higher performance, lower energy consumption, and/or balance overall system behavior. One such tunable component is the cache hierarchy. An efficient reconfigurable cache framework and tuning algorithms are proposed in [7].

Although reconfigurable caches are highly beneficial in desktop and embedded systems, currently, reconfigurable caches have not been considered in real-time systems due to several fundamental challenges. For example, how to employ and make efficient use of reconfigurable caches in real-time systems remains unsolved. Determining the appropriate cache configuration typically requires some amount of runtime evaluation of different candidates. Furthermore, any change in cache configuration on-the-fly may alter task execution time. In hard real-time systems, the benefit of reconfiguration is limited since both of these facts can make scheduling decisions difficult and eventually may lead to unpredictable system behavior. On the other hand, soft real-time systems offer much more flexibility, which can be exploited to achieve considerable energy savings at the cost of very minor impacts to user experiences. Our proposed research focuses on soft real-time systems.

To the best of our knowledge, this is the first approach in exploiting dynamic reconfigurable caches in real-time systems. This paper presents a novel methodology for using reconfigurable caches in real-time systems with preemptive tasks. Our proposed methodology, Scheduling-Aware Cache Reconfiguration (SACR), provides an efficient and near optimal cache tuning strategy based on static program profiling for both statically and dynamically scheduled real-time systems. The goal is to optimize energy consumption with performance considerations via reconfigurable cache tuning while ensuring that the majority of task deadlines are met.

The rest of the paper is organized as follows. Section 2 surveys the background literature addressing both dynamic cache reconfiguration and real-time scheduling techniques. Section 3 describes our proposed research on scheduling-aware cache reconfiguration in soft real-time systems. Section 4 presents our experimental results. Finally, Section 5 concludes the paper.

2. Related Work

There are no prior works in the area of dynamic cache reconfiguration in real-time systems. Our proposed research is the first attempt in this direction. This section surveys the background literature in the following three related domains.

2.1 Real-Time Scheduling Techniques

Based on task properties and associated systems, scheduling algorithms can be classified into various types [12]. Earliest Deadline First (EDF) scheduling [4] and Rate Monotonic (RM) scheduling [12] are the most frequently referenced fundamental scheduling algorithms in the real-time systems community. Periodic tasks, which usually have known worst case execution

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.66

time (WCET), period, and deadline are scheduled using such methods. Sporadic tasks are accepted into the system only if the task passes an acceptance test when it arrives. Since sporadic tasks normally have hard time constraints, all accepted tasks are guaranteed to meet their deadlines, and are thus treated as periodic tasks. Aperiodic tasks are scheduled whenever enough slack time is available. Hence, aperiodic tasks normally have soft deadlines and can only be scheduled as soon as possible. In reality, these three kinds of tasks may exist simultaneously. In this work, we use EDF as the scheduling algorithm for tasks with only soft real-time constraints. However, RM is also applicable with minor changes in our approach.

2.2 Caches in Real-Time Systems

Incorporating caches into real-time embedded systems faces certain difficulties due to the unpredictability imposed on the system. Scheduling algorithms have difficultly calculating WCET for tasks since data access time cannot be predetermined in the presence of caches. A great deal of research efforts are directed at employing caches in real-time systems either by proving schedulability through WCET analysis and/or avoiding hazardous compulsory miss uncertainty altogether. WCET analysis is a static, design time analysis of tasks in the presence of caches to predict cache impact on task execution times [14]. Cache locking [15] is a technique in which useful cache lines are "locked" in the cache when a task is preempted so that these blocks will not be evicted to accommodate the new incoming task. Through cache line locking, the WCET and cache behavior becomes more predictable since the major delay from data replacement and access is avoided. Cache partitioning [19] is a similar but more aggressive approach where the cache is partitioned into reserved regions, each of which can only cache data associated with a dedicated task. However, a potential drawback to both cache locking and cache partitioning is per-task reduction of cache resources. To alleviate this limitation, cache-related preemption delay analysis [18] features tight delay estimation so that prediction accuracy is higher than in traditional WCET analysis. This improved accuracy can in turn result in a durable task schedule. Our approach is applicable to real-time systems that employ caches.

2.3 Reconfigurable Cache Architectures

In power constrained embedded systems, nearly half of the overall power consumption is attributed to the cache subsystem [13]. Fortunately, since applications require vastly different cache requirements in terms of cache size, line size, and associativity, research shows that specializing the cache to an application's needs can reduce energy consumption by 62% on average [6].

There exists much work in dynamic cache reconfiguration [1] [7]. The reconfigurable cache architecture proposed by Zhang et al. [20] imposes no overhead to the critical path, thus cache access time does not increase. Furthermore, the cache tuner consists of small custom hardware or a lightweight process running on a co-processor, which can alter the cache configuration via hardware or software configuration registers. The underlying cache architecture consists of four separate banks, each of which acts as a separate way. Way concatenation, which logically concatenates ways together, enables configurable associativity. Way shutdown effectively shuts down ways to vary cache size. Configurable line size, or block size, is achieved by setting a unit-length base line size and then fetching subsequent lines if the line size increases.

Given a runtime reconfigurable cache, determining the best cache configuration is a difficult process. Dynamic and static analyses are two possible techniques. With dynamic analysis, cache configurations are evaluated in system during runtime to determine the best configuration. However, it is inappropriate for real-time systems as it either imposes unpredictable performance or significant energy overhead, both due to the exploration of suboptimal cache configurations. During static analysis, various cache alternatives are explored and the best cache configuration is selected for each application in its entirety (application-based tuning) [7] or for each phase of execution within an application (phase-based tuning) [17]. Regardless of the tuning method, the predetermined best cache configuration (based on design requirements) may be stored in a look-up table or encoded into specialized instructions. The static analysis approach is most appropriate for real-time systems due to its non-intrusive nature. However, previous methods focus solely on energy savings or Pareto-optimal points trading off energy consumption and performance. However, none of these methods consider task deadlines, which are imperative in real-time systems.

3. Scheduling-Aware Cache Reconfiguration

A major challenge for cache reconfiguration in real-time systems is that tasks are constrained by their deadlines. Even in soft real-time systems, task execution time cannot be unpredictable or prolonged arbitrarily. Our goal is to realize maximum energy savings while ensuring the system only faces an innocuous amount of deadline violations (if any). Our proposed methodology, Scheduling-Aware Cache Reconfiguration (SACR), provides an efficient and near optimal strategy for cache tuning based on static program profiling for both statically and dynamically scheduled real-time systems. Our approach statically executes, profiles, and analyzes each task intended to run in the system. The information obtained in the profiling process is fully utilized to make reconfiguration decisions dynamically. The remainder of this section is organized as follows. First, we present an overview of our approach using simple illustrative examples. Next, we present our static analysis technique for optimal cache configuration selection. Finally, we describe how the static analysis results are used during runtime for statically- and dynamically-scheduled real-time systems.

3.1 Overview

This section presents a simple demonstrative example to show how reconfigurable caches benefit real-time embedded systems. This example assumes a system with two tasks, *T1* and *T2*. Traditionally if a reconfigurable cache technique is not applied, the system will use a *base cache*[1] configuration $\text{Cache}_{\text{base}}$ throughout all task executions. In the presence of a reconfigurable cache, as shown in Figure 1, different optimal cache configurations are determined for every *phase* of each task. For ease of illustration, we divide each task into two phases. $Phase_1$ starts from the beginning to the end, and $phase_2$ starts from the half position of the dynamic instruction flow (midpoint) to the end. The terms $\text{Cache}^1_{\text{T1}}$, $\text{Cache}^2_{\text{T1}}$, $\text{Cache}^1_{\text{T2}}$, and $\text{Cache}^2_{\text{T2}}$ represent the optimal cache configurations for $phase_1$ and $phase_2$ of task *T1* and task *T2*, respectively. These

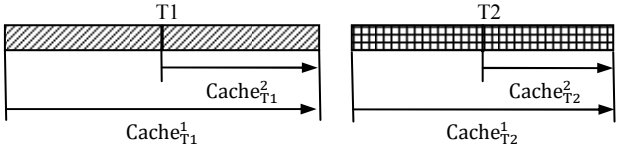


Figure 1: Cache configurations selected based on phases

[1] In this paper, we use the term "*base cache*" to refer to the cache used in typical real-time systems. Caches in such systems, as discussed in Section 2.2, are chosen to ensure durable task schedules.

configurations are chosen statically to be more energy efficient (with same or better performance), in their specific phases, than the global base cache, $\text{Cache}_{\text{base}}$.

Figure 2 illustrates how energy consumption can be reduced by using our approach in real-time systems. Figure 2(a) depicts a traditional system and Figure 2(b) depicts a system with a reconfigurable cache (our approach). In this example, *T2* arrives (at time *P1*) and preempts *T1*. In a traditional approach, the system executes using $\text{Cache}_{\text{base}}$ exclusively. With a reconfigurable cache, the first part of *T1* executes using $\text{Cache}^1_{\text{T1}}$. Similarly, $\text{Cache}^1_{\text{T2}}$ is used for execution of *T2*. Note that the actual preemption point of *T1* is not exactly at the same place where we pre-computed the optimal cache configuration (midpoint). When *T1* resumes at time point *P2*, the cache is tuned to $\text{Cache}^2_{\text{T1}}$ since the actual preemption point is closer to the midpoint compared to the starting point.

The overall energy consumed using a reconfigurable cache results from the energy savings due to use of different energy optimal caches for each phase of task execution compared to using one global base cache in the traditional system. Our experimental results suggest that the proposed approach can reduce energy consumption up to 74% without introducing any performance penalty.

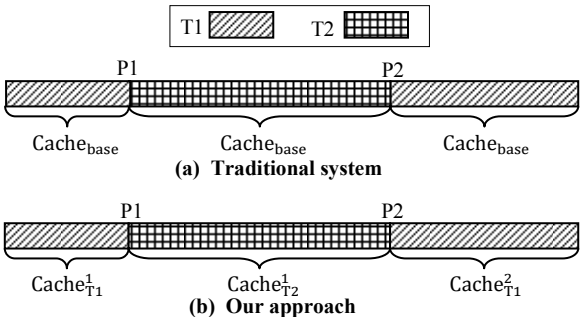


Figure 2: Dynamic cache reconfigurations for tasks T1 and T2.

3.2 Phase-based Optimal Cache Selection

This section describes our static analysis approach to determine the optimal cache configurations for various task phases. In a preemptive system, tasks may be interrupted and resumed at any point in time. Each time a task resumes, cache performance for the remainder of task execution will differ from the cache performance for the entire application due to its own distinguishing behaviors as well as cold-start compulsory cache misses. Thus, the optimal cache configuration for the remainder of the task execution may be different. Figure 3 depicts the general case where a task is divided by *n* potential preemption points (0, P_1, P_2 … P_{n-1}). We define a static profiled *phase* as the period of time between a predefined *potential preemption point* (also called *partition points*) and task completion. Here, C_1, C_2 … C_n represent the optimal cache configuration (either energy or performance) for each phase, respectively. Again, the potential preemption points, which define phases, are decided during the static profiling stage and are not necessarily the same as actual preemption points observed during system execution.

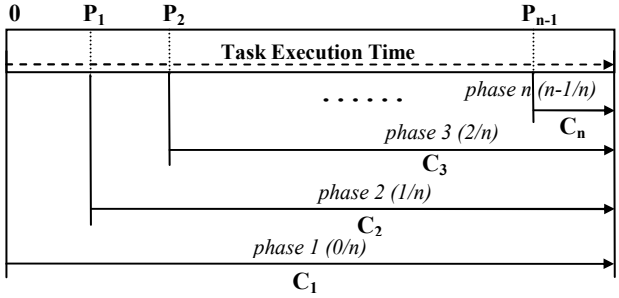


Figure 3: Task partitioning at *n* potential preemption points (P_i) resulting in *n* phases. Each phase comprises execution from the invocation/resumption point to task completion. C_i denotes the cache configuration used in each phase

During static profiling, a *partition factor* is chosen that determines the number of potential preemption points and resulting phases. Partition granularity is defined as the number of dynamic instructions between partition points. The partition granularity is determined by dividing the total number of dynamically executed instructions by the partition factor. Smaller granularities result in finer grained configuration, and potentially greater energy savings. However, making granularity too fine would result in a prohibitively large look-up table which would not be feasible due to area constraints. Thus, a trade-off should be made to determine a reasonable partition factor based on energy-savings potential and acceptable overheads.

An important question is whether a larger partition factor (finer granularity) reveals more energy savings. Our experimental results show that once the partition factor is larger than a certain threshold for a task, more and more neighboring partitions share the same optimal cache configuration. This is evident due to the well-established 90/10 rule of execution – 90% of the execution time is spent in only 10% of the code – in which the 90% of the time is typically spent executing small loops. For each loop iteration, except the first and last, execution behavior is typically similar, thus resulting in the same optimal cache configuration for all iterations. For a loop with N iterations, the partition factor need only be large enough to capture all dynamic instructions of iterations 2 through N – 1, as any smaller granularity would capture a subset of iterations, each of which have the same optimal configuration. Clearly, if there is no variation, no energy savings is possible. Even if variations can be observed, according to our experiments, they only happen with very limited ranges, which means a minor energy saving is possible only when preemption/resumption takes place in these ranges (8% of the dynamic instruction flow on average). Thus, the goal of a system designer is to find a partition factor which leads to maximized energy reduction and minimizes the number of partition points that need to be stored. Based on our experience, a partition factor ranging from four to seven is sufficient to generate a static profile table that SACR can utilize efficiently.

The *profile table* is the output of static analysis that stores the potential preemption points and the corresponding optimal cache configurations for each task. Section 3.3 and 3.4 describe how this profile table is used during runtime of statically- as well as dynamically-scheduled systems.

3.3 Statically Scheduled Systems

With static scheduling, arrival times, execution times, and deadlines are known a priori for each task and this information serves as scheduler input. The scheduler then provides a schedule detailing all actions taken during system execution. According to this schedule, we can statically execute and record the energy-optimal cache configurations that do not violate any task's deadline (in hard real-time systems) for every execution phase of each task. For soft real-time systems, global (system-wide) energy-optimal

configurations can be selected as long as the configuration performance does not severely affect system behavior. After this profiling step, the profile table is integrated with the scheduler so that the cache reconfiguration hardware can tune the cache accordingly for each scheduling decision.

3.4 Dynamically Scheduled Systems

With dynamic scheduling (online scheduling), scheduling decisions are made during runtime. In this scenario, task preemption points are unknown since new tasks may enter the system at any time with any feasible time constraint. In this section, we present two versions of our technique based on the nature of the target system.

3.4.1 SACR - Conservative Approach

In some soft real-time systems where time constraints are pressing, only an extremely small number of violations are tolerable. The SACR conservative approach could ensure that given a carefully chosen partition factor, almost every task could meet their deadlines with only few exceptions. To ensure the largest task schedulability, any reconfiguration decision will only change the cache into a lowest energy configuration whose execution time is not longer than that of the base cache. In other words, to maintain a high quality of service, only cache configurations with equal or higher performance than the base cache are chosen for each task phase. Note that the chosen energy-optimal configuration may not be the global lowest energy configuration but is the one with lowest energy consumption given the time constraint. We denote them as deadline-aware energy-optimal cache configurations.

The scheduler chooses the appropriate cache configuration from the generated profile table that contains the deadline-aware energy-optimal cache configurations for each task phase. Table 1 (a) shows the profile table for task *i* with a partition factor *p*. $EO_i(n/p)$ represents the deadline-aware energy-optimal cache configuration for partition phase *n/p* (which means the execution of this phase is from the partition point of *n/p* until the end of the task) in task *i*. Here, *n/p* represents the *n*'th phase in the set of *p* phases. The total dynamic instruction count (*TIN*) refers to the number of dynamic instructions executed in a single run of that task.

During system execution, the scheduler maintains a task list keeping track of all existing tasks as shown in Table 1(b). In addition to the static profile table records from Table 1(a), runtime information such as arrival time (A_i), deadline (D_i), and remaining number of dynamic instructions (*RIN*) is recorded. This information is stored not only for the scheduler, but also for the cache tuner. When a newly arrived task begins execution, the deadline-aware energy-optimal cache configuration ($EO_i(0/p)$) is obtained from the task list entry, and the cache tuner adjusts the cache appropriately.

As indicated in Section 3.2.1, potential preemption points are pre-decided during the profile table generation process. However, it is highly unlikely that the actual preemptions will occur precisely on these potential preemption points. Hence, a *nearest-neighbor technique* is used to determine which cache configuration should be used. Essentially, if the preemption point falls between partition points *n/p* and *(n+1)/p*, the nearest point will be selected as the current cache configuration. As our experimental result shows, conservative SACR obtains significant energy savings with little or no impact on quality of service.

3.4.2 SACR - Aggressive Approach

In a soft real-time system with less pressing time constraints, a more aggressive version of SACR can reveal additional energy savings at the cost of possibly violating several low priority future task deadlines, while remaining in an acceptable range.

Similar to the conservative approach, a profile table is associated with every task in the system; however this profile table contains the performance-optimal cache configuration (whose execution time is the shortest) in addition to the energy-optimal configuration (the one with lowest energy consumption among all candidates) cache for every task phase. In order to assist dynamic scheduling, the profile table also includes the corresponding phase's approximate execution time (in cycles) for each configuration. Table 2(a) shows the profile table for task *i* with a partition factor of *p*. The terms *EO*, *EOT*, *PO*, and *POT* stand for the energy-optimal cache configuration, the energy-optimal cache configuration's execution time, the performance-optimal cache configuration, and the performance-optimal cache configuration's execution time, respectively.

Table 2(b) shows the task list entry for the aggressive

Table 1: (a) Static profile table and (b) Task list entry for task *i* for the conservative approach.

Task ID: i	Partition Factor: p
Total Instruction Number (TIN)	
$EO_i(0/p)$	
$EO_i(1/p)$	
$EO_i(2/p)$	
.....	
$EO_i(p-1/p)$	

(a)

Task ID: i	Partition Factor: p
Arrival time (A_i)	Deadline (D_i)
Total Instruction Number (TIN)	Remaining Instruction Number (RIN)
$EO_i(0/p)$	
$EO_i(1/p)$	
$EO_i(2/p)$	
.....	
$EO_i(p-1/p)$	

(b)

Table 2: (a) Static profile table and (b) task list entry for task *i* in the aggressive approach.

Task ID: i		Partition Factor: p	
Total Instruction Number (TIN)			
$EO_i(0/p)$	$EOT_i(0/p)$	$PO_i(0/p)$	$POT_i(0/p)$
$EO_i(1/p)$	$EOT_i(1/p)$	$PO_i(1/p)$	$POT_i(1/p)$
$EO_i(2/p)$	$EOT_i(2/p)$	$PO_i(2/p)$	$POT_i(2/p)$
.....			
$EO_i(p-1/p)$	$EOT_i(p-1/p)$	$PO_i(p-1/p)$	$POT_i(p-1/p)$

(a)

Task ID: i		Partition Factor: p	
Arrival time (A_i)		Deadline$_i$ (D_i)	
Total Instruction Number (TIN)		Remaining Instruction Number (RIN)	
Current Phase (CP)			
$EO_i(0/p)$	$EOT_i(0/p)$	$PO_i(0/p)$	$POT_i(0/p)$
$EO_i(1/p)$	$EOT_i(1/p)$	$PO_i(1/p)$	$POT_i(1/p)$
$EO_i(2/p)$	$EOT_i(2/p)$	$PO_i(2/p)$	$POT_i(2/p)$
.....			
$EO_i(p-1/p)$	$EOT_i(p-1/p)$	$PO_i(p-1/p)$	$POT_i(p-1/p)$

(b)

approach. The difference from the conservative approach (shown in Table 1(b)) is that every task list entry also holds a Current Phase (CP_i) identifier. CP_i denotes the partition point that this task's execution just passed and is useful for cache reconfiguration upon task resumption. In addition to the task list, the scheduler also maintains another runtime data structure called the ready task list (*RTL*), which contains an identifier representing each task currently ready to execute in the system.

To explain the SACR aggressive approach, we use an illustrative example in which there are three tasks, *T1*, *T2*, and *T3*, with deadlines D_{T1}, D_{T2}, and D_{T3}, where $D_{T2} < D_{T1} < D_{T3}$. According to EDF, the priority sequence is simply the opposite of the deadlines, which is $Pri_2 > Pri_1 > Pri_3$. Figure 4 shows a schedule for these tasks. Note that *P0*, *P1*, *P2,* and *P3* represent the time instances when any event (arrival, completion, etc.) occurs. At time point *P0*, *T1* arrives and the scheduler generates the task list entry for *T1* and adds *T1* to the *RTL*. Since *T1* is currently the only task in the system, the scheduler instructs the cache tuner to configure the cache to $EO_{T1}(0/p)$ if and only if $P0 + EOT_{T1}(0/p) < D_{T1}$, otherwise the cache will be tuned to $PO_{T1}(0/p)$, which ensures that *T1*'s deadline will be met. At time point *P1*, *T2* arrives with priority higher than the currently active task *T1*. The scheduler calculates *T1*'s current phase CP_{T1} and updates *T1*'s task list entry. Note that *T1's* deadline may be violated if the following inequality holds:

$$P1 + POT_{T1}((CP_{T1}+1)/p) + POT_{T2}(0/p) > D_{T1} \qquad (1)$$

This is obviously an underestimation of the execution time that *T2* and the remaining portion of *T1* will take, thus more aggressive, but it favors tasks with higher priority. However, if we use $POT_{T1}(CP_{T1}/p)$ in Equation 1, *T2* may have a lower chance of being accepted, but *T1* would likely meet its deadline.

If Equation 1 does not hold, the scheduler determines *T2*'s cache configuration C_{T2} as follows (assuming $POT_i(0/p) < D_i$ for all tasks *i* otherwise task *i* is not schedulable in any situation):

if $(P1 + EOT_{T2}(0/p) > D_{T2})$
  then $C_{T2} = PO_{T2}(0/p)$
else if $(P1 + EOT_{T2}(0/p) + POT_{T1}((CP_{T1} + 1)/p) < D_{T1})$
  then $C_{T2} = EO_{T2}(0/p)$
else if $(P1 + EOT_{T2}(0/p) + POT_{T1}((CP_{T1} + 1)/p) > D_{T1})$
  then $C_{T2} = PO_{T2}(0/p)$

At time point *P2*, *T2* completes and *T1* resumes since it is the only ready task. The scheduler utilizes CP_{T1} to determine the appropriate partition to choose a cache configuration. This technique is similar in principle to the nearest neighbor approach used in Section 3.4.1, except that a decision should be made whether to use the energy-optimal or performance-optimal configuration based on the remaining time budget. At some point during *T1*'s execution, *T3* arrives but since *T3* has a lower priority than *T1*, *T3* begins execution after *T1* completes execution. By this time, *T3* is the only task and its cache configuration decision is made using the same method as task *T1* at time *P0*.

4. Experiments

4.1 Experimental Setup

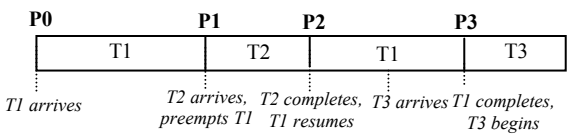


Figure 4: Task set and sample scheduling.

To quantify energy savings using SACR, we examined selected benchmarks from the MediaBench [11] and EEMBC Automotive [5] benchmark suites, representing typical tasks that might be present in a real-time system. All applications were executed with the default input sets provided with the benchmarks suites.

We utilized the configurable cache architecture developed by Zhang et al [20] with a four-bank cache of base size 4 KB, which offers sizes of 1 KB, 2 KB, and 4 KB, line sizes ranging from 16 bytes to 64 bytes, and associativity of 1-way, 2-way, and 4-way. For comparison purposes, we define the *base cache* configuration to be a 4 KB, 2-way set associative cache with a 32-byte line size, a reasonably common configuration that meets the needs of the benchmarks studied.

To obtain cache hit and miss statistics, we used the SimpleScalar toolset [3] to simulate the applications. Our energy model, adopted from the one used in [20], calculates both dynamic and static energy consumption, memory latency, CPU stall energy, and main memory fetch energy. We updated the dynamic energy consumption for each cache configuration using CACTI 4.2 [9].

To populate the static profile tables for each benchmark, we utilize SimpleScalar's external I/O trace files (eio file), checkpointing, and fastforwarding capabilities. This method allows for every benchmark phase to be individually profiled via fastforwarding execution to each potential preemption point. In our experiments, we examined partition factors ranging from two to seven potential preemption points. Driven by Perl scripts, the design space of 18 cache configurations is exhaustively explored during static analysis to determine the energy-, performance-, and deadline-aware energy-optimal cache configurations for each phase of each benchmark.

4.2 Results

To model sample real-time embedded systems with multiple executing tasks, we created four different task sets as shown in Table 3. In each task set, the three selected benchmarks have comparable dynamic instruction counts in order to avoid behavioral domination by one relatively large task. For system simulation, task deadlines and priorities are as described in Section 3.4.2. We examine a varying set of preempting points and average these values so that our results represent a generic degree of scheduling decisions since task *T2* may preempt task *T1* at any point in time.

We compare the energy consumption for each task set using different schemes: a fixed base cache configuration, the SACR conservative approach, and the SACR aggressive approach. Energy consumption is normalized to the fixed base cache configuration such that value of 1 represents our baseline. Figure 5 and Figure 6 present energy savings for the instruction and data cache subsystems, respectively. Energy savings in the instruction cache subsystem ranges from 22% to 36% for the SACR conservative approach, while it reaches as high as 74% for the SACR aggressive approach. Energy savings average 28% and 51% for the SACR conservative and aggressive approaches, respectively. In the data cache subsystem, energy saving is generally less than that of the

Table 3: Benchmark task sets.

	Task 1	Task 2	Task 3
Task Set 1	epic*	pegwit*	rawcaudio*
Task Set 2	cjpeg*	toast*	mpeg2*
Task Set 3	A2TIME01**	AIFFTR01**	AIFIRF01**
Task Set 4	BITMNP01**	IDCTRN01**	RSPEED01**

** MediaBench **EEMBC*

instruction cache subsystem due to less variation in cache configuration requirements. In the data cache subsystem, energy savings range from 15% to 47% for the SACR conservative approach, while it reaches as high as 64% for the SACR aggressive approach, and average 17% and 22% for the SACR conservative and aggressive approaches, respectively.

The remainder of this section describes the overhead of implementing the profile table in hardware. The profile table is stored in SRAM and accessed by the cache tuner to fetch the cache configuration information. The size of the table depends on the number of tasks in the system and the partition factor used. The table entry consists of five bits since the configurable cache architecture used in this study offers 18 possible cache configurations. We have implemented the profile table using Verilog HDL and synthesized using Synopsis Design Compiler with TSMC 0.18 cell library. Table 4 illustrates our results. Each row in the table indicates the area, dynamic power, leakage power, and critical path length for profile table with different sizes. We assume a table lookup frequency of one million nanoseconds during dynamic power computation, which means there is a table lookup every five hundred thousand cycles (a reasonably frequency based on normal task sizes) using a 500MHz CPU. We observed that on average for each task set, the energy overhead of our approach only account for less than 0.02% (450 nJ comparing to 2825563 nJ) of the total energy savings. Therefore, the energy overhead of implementing the profile table is negligible compared to the energy savings produced by our approach.

Table 4: Overhead of different lookup tables

Table size (# of entries)	Area (μm^2)	Dynamic Power (nW)	Leakage Power (nW)	Critical Path Length (ns)
64	61416	134.40	114.37	0.91
128	121200	266.22	224.90	0.91
256	244520	544.73	461.30	1.08
512	483416	994.20	904.70	1.20

5. Conclusions

Dynamic reconfiguration techniques are widely used in designing efficient embedded systems. Dynamic cache reconfiguration is a promising approach to improve both energy consumption and overall performance. The contribution of this paper is a novel scheduling aware dynamic cache reconfiguration technique for soft real-time embedded systems. To the best of our knowledge, this is the first approach integrating dynamic cache reconfigurations into real-time embedded systems. Our methodology employs an ideal combination of static analysis and dynamic tuning of cache parameters with minor or no impact on timing constraints. Our experiments demonstrated a 50% reduction on average in the overall energy consumption of the cache subsystem in soft real-time embedded systems.

References

[1] D. H. Albonesi, "Selective cache ways: on demand cache resource allocation", *Journal of Instruction Level Parallelism, May 2002*.

[2] L. Benini, G. De Micheli, "A survey of design techniques for system-level dynamic power management", *TVLSI, 8(3):299-316, June 2000.*

[3] D. Burger, T. Austin, S. Bennet, "Evaluating future microprocessors: the simplescalar toolset", *CS-TR-1308, University of Wisconsin, 2000.*

[4] G. Buttazzo, Hard Real-Time Computing Systems. *Kluwer 1995.*

[5] EEMBC, http://www.eembc.org.

[6] A. Gordon-Ross, F. Vahid, N. Dutt, "Automatic Tuning of Two-Level Caches to Embedded Applications", *DATE,* page 10208, *2004.*

[7] A. Gordon-Ross, F. Vahid, N. Dutt, "Fast configurable-cache tuning with a unified second level cache, *ISLPED, pages 323-326, 2005.*

[8] I. Hong et al., "Power optimization of variable voltage core-based systems", IEEE *TCAD, 18(12):1702-1714, December 1998.*

[9] HP Labs, CACTI 4.2, http://www.hpl.hp.com/

[10] R. Jejurikar, R. Gupta, "Energy-Aware Task Scheduling With Task Synchronization for Embedded Real-Time Systems", IEEE *TCAD, 25(6):1024-103, June 2006.*

[11] C. Lee et al. "Mediabench: A tool for evaluating and synthesizing multimedia and communication systems", *MICRO, 1997.*

[12] J. Liu, Real-Time Systems. Upper Saddle River, Prentice-Hall 2000.

[13] A. Malik et al., "A low power unified cache architecture providing power and performance flexibility", *ISLPED, pages 241-243, 2000.*

[14] I. Puant, "Cache analysis vs static cache locking for schedulability analysis in multitasking real-time systems", *ECRTS, 2002.*

[15] I. Puant et al., "Low-Complexity Algorithms for Static Cache Locking in Multitasking Hard Real-Time Systems", *RTSS, 114-125, 2002.*

[16] G. Quan, X. S. Hu, "Energy Efficient DVS Schedule for Fixed-Priority Real-Time Systems", *ACM TECS,6(4),article 29, 2007.*

[17] T. Sherwood et al., "Discovering and exploiting program phases", *IEEE Micro, December 2003.*

[18] Y. Tan, V. Mooney, "Timing Analysis for Preemptive Multitasking Real-Time Systems with Caches", *ACM TECS, (6)1, article 7, 2007.*

[19] A. Wolfe, "Software-Based Cache Partitioning for Real-time Applications", *IWRCS, 1993.*

[20] C. Zhang, F. Vahid, W. Najjar, "A Highly Configurable Cache for Low Energy Embedded Systems", *ACM TECS, 6(4):362-387, 2005.*

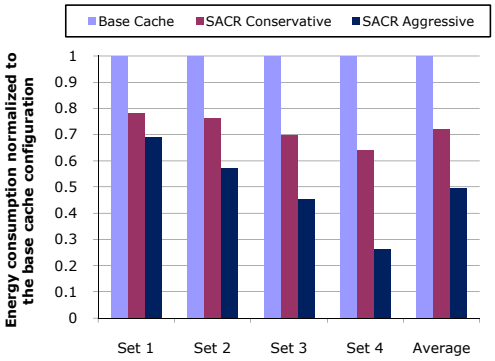


Figure 5: Instruction cache subsystem energy consumption normalized to the base cache configuration for each task set

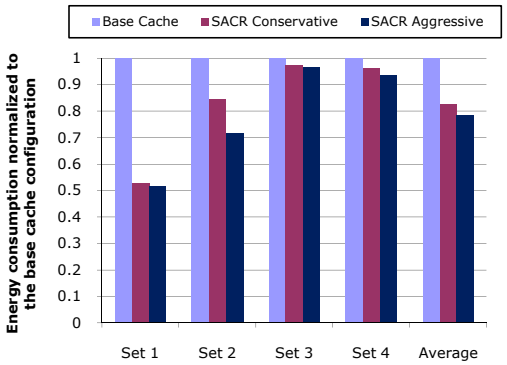


Figure 6: Data cache subsystem energy consumption normalized to the base cache configuration for each task set

2009 22nd International Conference on VLSI Design

H-NMRU: A Low Area, High Performance Cache Replacement Policy for Embedded Processors

Sourav Roy
Freescale Semiconductors, India Design Center
sourav.roy@freescale.com

Abstract

We propose a low area, high performance cache replacement policy for embedded processors called Hierarchical Non-Most-Recently-Used (H-NMRU). The H-NMRU is a parameterizable policy where we can tradeoff performance with area. We extended the Dinero cache simulator [1] with the H-NMRU policy and performed architectural exploration with a set of cellular and multimedia benchmarks. On a 16 way cache, a two level H-NMRU policy where the first and second levels have 8 and 2 branches respectively, performs as good as the Pseudo-LRU (PLRU) policy with storage area saving of 27%. Compared to true LRU, H-NMRU on a 16 way cache saves huge amount of area (82%) with marginal increase of cache misses (3%). Similar result was also noticed on other cache like structures like branch target buffers. Therefore the two level H-NMRU cache replacement policy (with associativity/2 and 2 branches on the two levels) is a very attractive option for caches on embedded processors with associativities greater than 4.

1 Introduction

Older generation embedded processors would consist of a large internal SRAM and an external memory interface connecting to a DRAM outside the chip. Most applications would fit in the internal SRAM comfortably. Caches were not preferred due to real-time variability in memory access latencies. As embedded systems became more complex, applications became very large and caches became inevitable. In the last decade, cache architectures have penetrated the world of embedded processors. At present embedded processors have cache architectures as complex as general purpose processors. The architecture of embedded processor caches, especially on mobile devices, is complicated by the fact that all three metrics of performance, power and area have to be satisfied simultaneously within the given constraints. This is different from general purpose processors, where performance is the clear priority, followed by power (in recent times). Today in the multi-core era, there is a trend to move to embedded systems with multiple processors, each of low area and frequency connected with high throughput interconnect fabric. Hence there is a need to reduce the complexity, area and power of individual processor components like caches, with minimal performance reduction.

Cache replacement policies determine which line to replace when there is a miss. In a set-associative cache, a miss occurs when the accessed set is full. There are three replacement policies that are widely used : LRU (least recently used), FIFO (first in first out) and RAND (random). Among these LRU is widely accepted to be the most preferred cache replacement policy. LRU most closely corresponds to the concept of temporal locality. Typically, on general purpose benchmarks FIFO and RAND amplifies the LRU miss rate by 12% and 20% respectively [2] [3]. More recently there has been some work on cache performance evaluation on SPEC2K benchmarks [4] [5]. The LRU policy is implemented as a circular stack as shown in fig 1. During a hit the most recently accessed line is placed at the top of stack. During a miss, the line at the bottom of the stack is replaced and the newly inserted line is placed at the stack top. Though LRU is the preferred policy from a performance angle, it is also the most complicated to implement. Each set comes with its own LRU history bits and they are updated with each memory reference to that set. For a m-way set-associative memory $(m-1) * log_2 m$ bits are required per directory entry of a set [2]. This is a big overhead in embedded processors where area is extremely important. To overcome this problem, approximations to LRU which occupy much lesser area have been suggested. The most prominent and widely used among them is Pseudo LRU (PLRU). The PLRU is extensively used in the PowerPC series

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.32

of microprocessors [6]. The PLRU is implemented a binary tree, whose leaf nodes are the lines in the set, as shown in fig. 1. During a hit, the path to the referenced leaf node is traversed and on the way the node values are flipped to point to the other branch. During a miss the tree is traversed according to the node values to find the line for replacement. For a m-way set-associative cache, PLRU requires $(m-1)$ bits per directory entry of a set. The PLRU while occupying much lesser area has been reported to perform as well as the LRU policy on general purpose benchmarks [4]. Most embedded processors use LRU or PLRU as their cache replacement policies. Processors using LRU replacement policy typically avoid associativities higher than 4 due to extra area overhead of LRU history bits. PLRU can be also used for higher associativities like 8 and 16, and in other highly or fully associative cache like structures like branch-target-buffers (BTB).

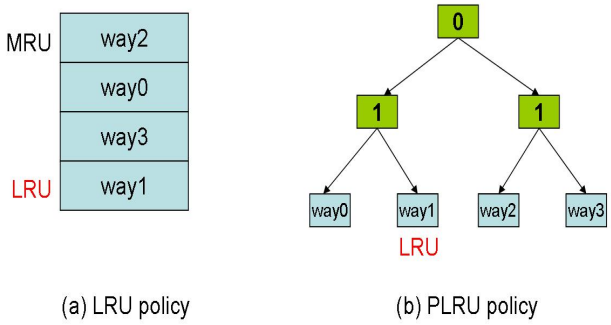


Figure 1. LRU and PLRU cache replacement policies

Cache replacement policies have been explored thoroughly over the last two decades. For embedded processors, the known cache replacement policies are efficient enough to provide high performance. However in today's multi-core era there is an acute requirement to reduce the complexity and area of caches, with minimal performance impact. In this work we have proposed a new cache policy H-NMRU to reduce area without sacrificing non-negligible performance over the LRU policy. The rest of this paper is organized as follows. Section 2 introduces the H-NMRU cache replacement policy and its computational and space complexity. Section 3 describes the results of using H-NMRU cache replacement policy on a set of cellular and multimedia benchmarks, vis-a-vis other popular cache replacement policies. Section 4 gives further results of applying H-NMRU replacement policy in BTBs on a UMTS voice call trace. Section 5 concludes with the advantage of using H-NMRU over LRU and PLRU policies.

2 The H-NMRU Cache Replacement Policy

The H-NMRU cache replacement policy can be explained using a multi-way tree, where the leaf nodes represent the lines in the set. Each intermediate node stores the value of its most-recently-used (MRU) child. Let us take an example of a 16-way set-associative cache as shown in fig. 2. In this example, the root has 4 branches. Each child in turn has 2 branches, followed by another 2 branches in the next level. This multi-way tree has 3 levels. It is represented using H-NMRU(4,2,2) where the numbers represent the number of branches at each level of the tree. The nodes of the tree store the value of the most recently traversed branch.

During a cache hit, the H-NMRU tree is traversed to reach the accessed line at the leaf node. On the way, the value of the nodes are updated to point to the path of traversal. In this way, the most recently used branches are stored at each node of the tree. On a cache miss, the tree is traversed selecting a random value different from the MRU value stored in the node. From each level a non-MRU path is selected. In this way, this algorithm points to a leaf node which has not been accessed in recent times. For example in fig. 2, the most recently used way is 11, which can be determined by a simple tree traversal. The dotted lines represent the current set of branches that can be traversed randomly for way replacement. Hence the H-NMRU policy selects one from way 3, way 5, or way 14 for replacement. The random replacement is done in hardware with simple linear feedback shift registers (LFSR). The random selection is done only for the accessed set in the cache. Hence it occupies negligible hardware area.

Since this is a parameterizable policy, the key is to find the correct number of levels and the number of branches for each level. Let us define a generic H-NMRU policy for a cache with associativity N:

$$\text{Cache Policy} : \text{H-NMRU}(i_0, i_1, \ldots, i_{n-1}) \quad \text{with} \prod_{j=0}^{n-1} i_j = N$$

The PLRU can then be defined as a special case of H-NMRU where each of the $i_j = 2$ and $n = log_2 N$.

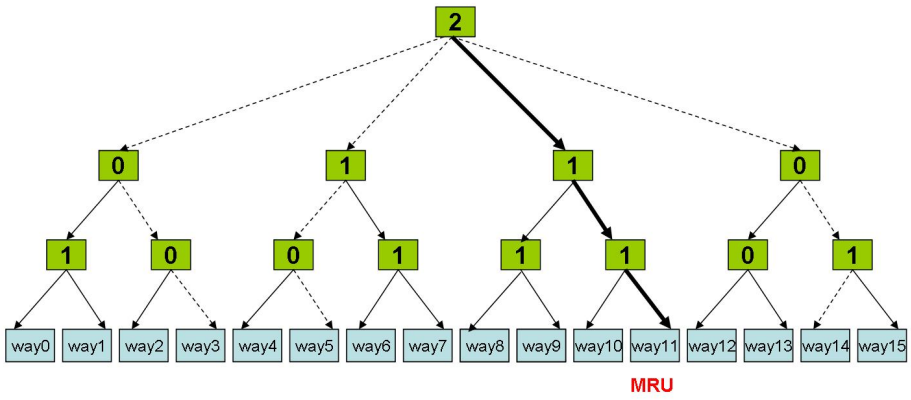


Figure 2. H-NMRU(4,2,2)

Similarly NMRU is also a special case of H-NMRU.

$$\text{PLRU} = \text{H-NMRU}(2, 2, \ldots, 2)$$
$$\text{NMRU} = \text{H-NMRU}(\text{N})$$

The computational complexity of H-NMRU is directly related to the number of levels in the multi-way tree i.e., n. Let us now find the space complexity of the H-NMRU policy. A node that has m branches can be represented using $log_2 m$ bits. Hence the number of history bits per set required for the H-NMRU policy is given by,

$$\text{No. of bits per set} = \sum_{j=0}^{n-1} (\prod_{k=0}^{j-1} i_k) * log_2(i_j) \quad (1)$$

The randomizer hardware needs to select the non-MRU branch at every level. Hence it can select one out of $(i_j - 1)$ branches. For any level with greater than 2 branches, the LFSR needs to have $log_2(i_j)$ bits. In a simple implementation of the randomizer hardware we create a pseudo-random sequence of $(i_j - 1)$ states excluding the all-zero state. When this state is exclusive-OR ed with the MRU state value, we get the random non-MRU state.

Next we explore the design space of the H-NMRU policy by evaluating the performance for a 16 way, 16 KB data cache with different parameters. Fig. 3 plots the average misses for different H-NMRU configurations. The misses are normalized with respect to that of PLRU or H-NMRU(2,2,2,2). The misses are measured by simulating a large number of cellular and multimedia benchmarks on the Dinero cache simulator which has been extended to support the configurable H-NMRU replacement policy. As seen from the plot, three other H-NMRU configurations have very similar performance (within 2% degradation) to that of PLRU but with much lower area requirement. The most attractive is the H-NMRU(8,2) configuration which uses only 11 bits per set entry instead of 15 as in PLRU, yet deliver a performance with only 2% degradation. As we reduce the number of bits per set, the miss rate of the H-NMRU policy increases. For the same number of bits, H-NMRU(8,2) performs better than H-NMRU(2,2,4).

3 Performance and Area of H-NMRU policy

In this section we will discuss the performance of the H-NMRU cache replacement policy. We extended the Dinero cache simulator to support the H-NMRU cache replacement policy. Then we applied several real-life traces on it to measure the cache performance. The benchmarks included in the paper are cellular and multimedia applications - Adaptive MultiRate (AMR) encoder, G.729 speech codec, MPEG4 decoder, Session Initiation Protocol (SIP), protocol stack of 3G voice call with HSDPA, and MP3 decoder. All the traces consist of more than 100 million instruction and data accesses. This is to ensure that we do not get misleading results from small traces. Level 1 instruction and data caches with 16 ways and 32B line size have been simulated using Dinero. Among the entire set of H-NMRU configurations, we have chosen H-NMRU(8,2) and H-NMRU(4,2,2). The former is the most attractive option from an area vs. performance trade off, whereas the latter is the best performance obtained from a non-PLRU H-NMRU policy. The results of individual applications are shown in fig. 4 and the av-

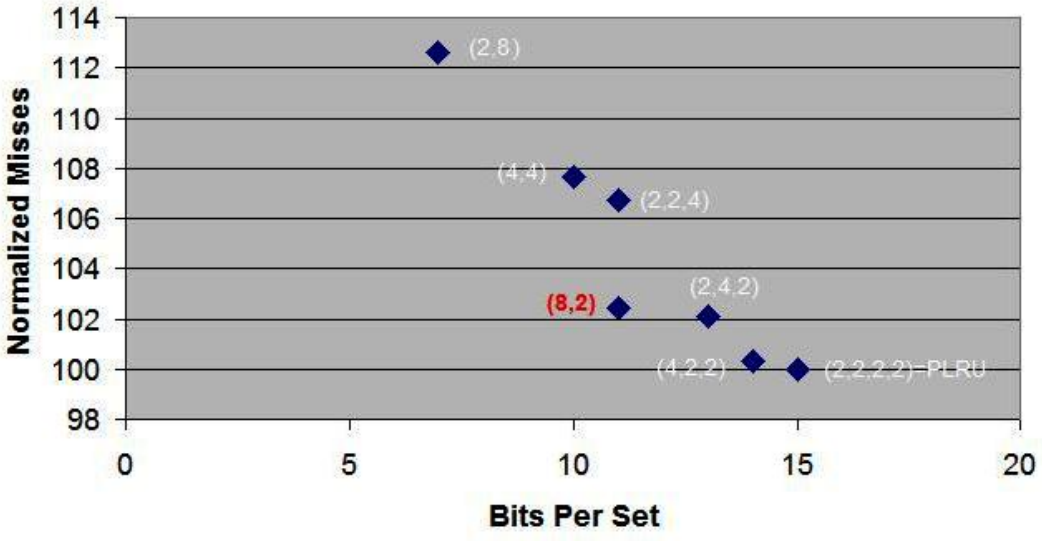


Figure 3. Design Space Exploration of H-NMRU policy on 16KB Data Cache

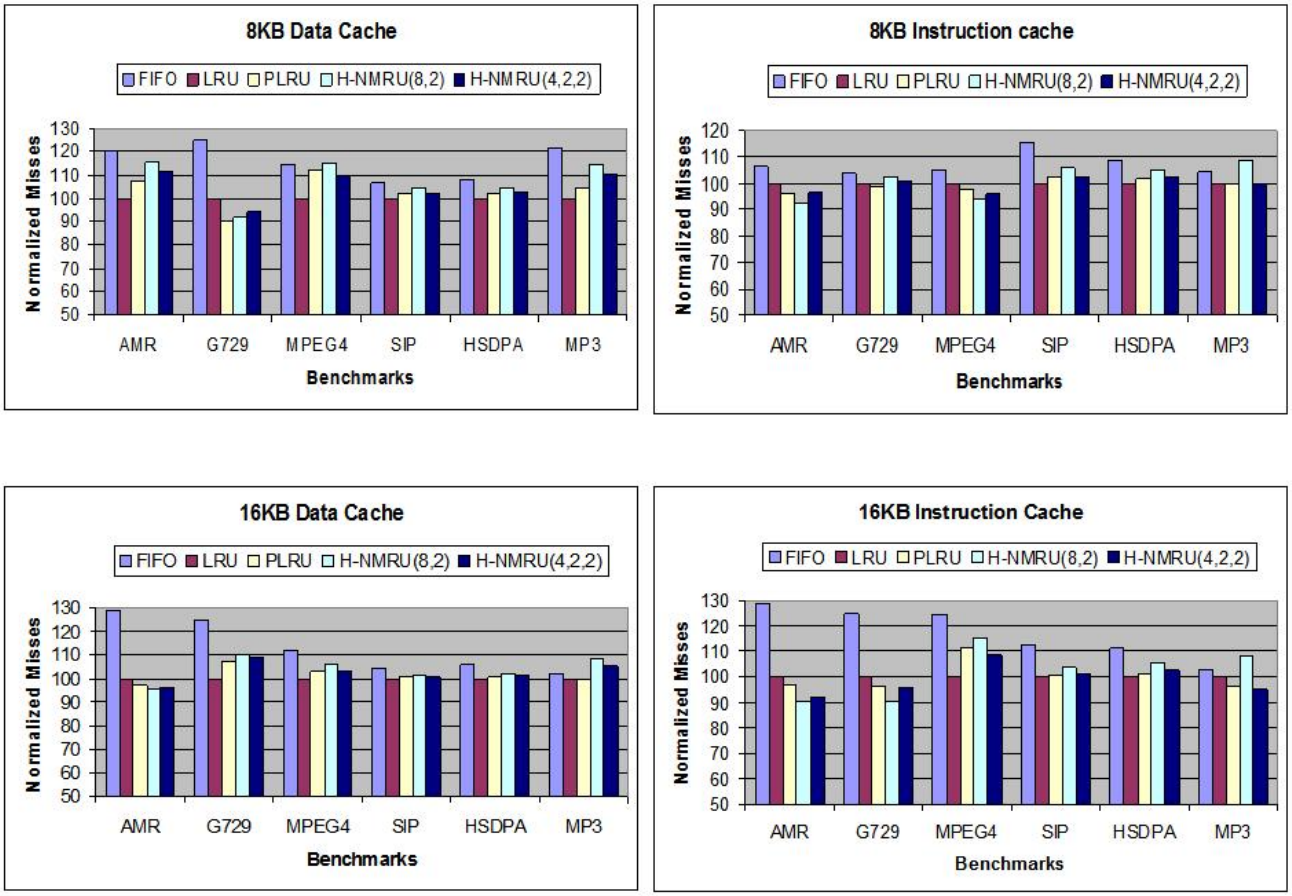


Figure 4. Comparison of various cache replacement policies

erage over all the applications is shown in fig. 5. We conclude that the miss rate for H-NMRU(8,2) is only 3% higher than LRU and 1.5-2.0% higher than PLRU. Other cache metrics like traffic depend on the miss rate, and so we have only analyzed the cache misses. Though we have considered only level 1 caches here, similar relative miss rates can be observed on level 2 caches as well. This directly follows from the hypothesis that if the level 1 cache is small compared to the level 2 cache, the performance of the latter can be computed by assuming that no level 1 cache is present [7].

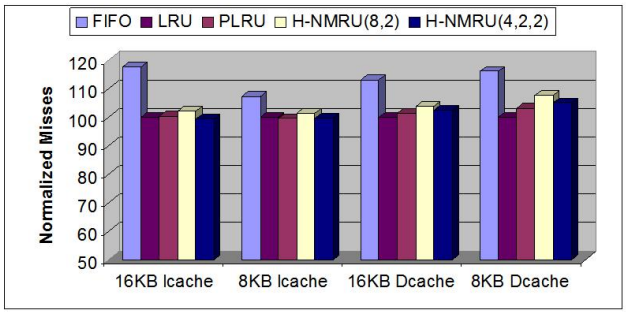


Figure 5. Average Performance

The H-NMRU(N/2,2) option is the best option in terms of area savings with very marginal performance loss. For an associativity N, PLRU bits per set is given by $(N-1)$ whereas H-NMRU(N/2,2) bits per set is given by $log_2(N/2) + N/2$. As N increases, the area savings of H-NMRU is higher. For large N, $N/2 >> log_2(N/2)$, hence H-NMRU bits $\approx N/2$ and PLRU bits $\approx N$. Therefore theoretically H-NMRU(N/2,2) takes half the storage bits of PLRU for large associativities. For typical associativities of 16 and 32, H-NMRU reduces storage requirement by 27% and 35% over PLRU. Compared to LRU, the H-NMRU savings are 82% and 87% respectively. Fig. 6 compares PLRU and H-NMRU(N/2,2) storage bits across cache sizes and associativities. It is important to note that it is not only the storage bits that are reduced but also the associated logic around the storage elements.

From an absolute area perspective, L1 caches typically implement replacement storage bits using registers to avoid frequency reduction. Approximate savings of H-NMRU(N/2,2) over PLRU for 16 way, 16KB and 32KB L1 caches will be 20,000 and 50,000 um^2 in 65nm technology. For a typical 256KB L2 cache, the approximate saving of H-NMRU over PLRU is 2.5 mm^2 in 65nm technology. Sometimes L2 cache replacement bits are stored in SRAMs as they are not timing critical. Then the approximate area savings would reduce to 0.2 mm^2. Hence H-NMRU is practically more suited for higher associativities and also for large caches as found in L2 cache. It is to be noted here that real silicon area depends on the timing of the particular chip and also on the technology.

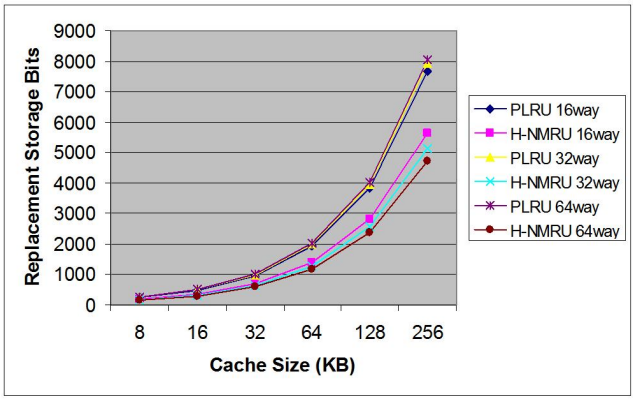


Figure 6. Area Comparison of PLRU and H-NMRU(N/2,2)

4 H-NMRU replacement for BTB

In this section we explore the performance of the H-NMRU replacement policy for another processor component called Branch Target Buffer (BTB). The BTB is a cache like structure that is used to reduce the performance penalty of branches by predicting the path of the branch. Each entry in the BTB typically contains the branch address, the target address and the prediction bits. When the processor fetches the instruction, its address is compared with the entries in the BTB. If there is a match with any entry, the prediction bits are used to predict if the branch will be taken or not. In case the prediction is branch taken, then the target address is used as the next instruction fetch address. This saves the branch penalty of the pipeline. There are two aspects of the BTB that are important for effective operation viz., BTB prediction accuracy and the BTB hit ratio. We will deal with the latter here in connection with replacement algorithm. If there is a BTB miss, it is advantageous to predict that program flow should go in line. Once there is a BTB miss and the program takes a branch, the BTB needs to be updated with the new branch entry. In this case an entry needs to be evicted to make space for the new entry. Typically, the LRU policy is used as the replacement policy. In this section we explore the performance of both PLRU and H-NMRU policies for BTB.

Several embedded processors with deeper pipelines

have started using BTB for branch performance improvement. Popular examples are ARM11 [8] and StarCore 3400 [9]. The challenge in embedded processors is to improve branch performance with limited area overhead. Typically modest size BTBs in embedded processors have high associativity, even fully associative or only very few (2-4) sets. This increases the area of LRU bits. To reduce the area overhead of history bits for cache replacement, the H-NMRU replacement policy can be used. In fig. 7 we plot the BTB misses for LRU, PLRU and H-NMRU(size/2,2) policy. The BTB explored here has full associativity and the application is a 3GPP UMTS voice call application. Few billion instructions have been simulated to study the BTB miss rates. As we see the H-NMRU policy performs very close (within 2-3%) to the LRU and PLRU policies while saving substantial area. For example the PLRU policy requires 31, 63, 127, 255 bits whereas the H-NMRU(size/2,2) would require 20, 37, 70, 135 bits respectively. Since the H-NMRU policy reduces storage bits and has only two levels in the tree, the timing also considerably improves.

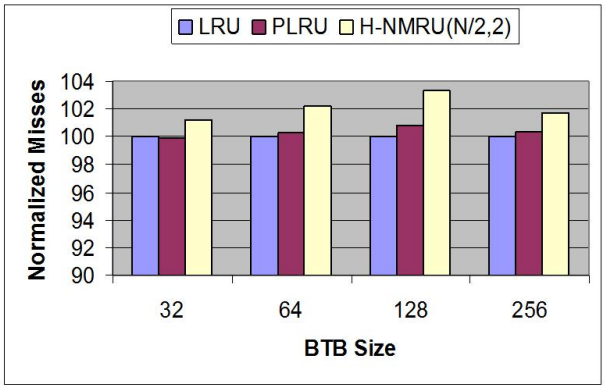


Figure 7. BTB Misses for UMTS voice call

5 Conclusion

In this paper we presented a new cache replacement policy called H-NMRU that saves substantial area while providing similar performance to LRU and PLRU. The H-NMRU is a parameterizable policy and among them an attractive option is H-NMRU(way/2,2). On a 16 way cache, it saves 27% area over PLRU with minimal loss of performance(1.5%). The absolute area saving is encouraging for highly associative and level 2 caches. This policy can also be used for BTB entry replacement, where typically associativity is high. Timing also improves considerably due to lower number of bits and only two levels in the tree. Hence H-NMRU(N/2,2) policy can help to reduce complexity, area and standby power of cache structures in present-day multi-core mobile platforms.

Here we have explored H-NMRU cache replacement policy in relation with embedded processors as area is a high priority for them. However we would also like to encourage architects in the general purpose computing domain to explore H-NMRU.

6 Acknowledgments

We would like to thank Brace Randall and Itay Peled at Freescale Semiconductors for the application traces that we used for cache performance evaluation.

References

[1] J. Elder and M. Hill. (1997) Dinero IV trace-driven uniprocessor cache simulator. [Online]. Available: http://www.cs.wisc.edu/~markhill/DineroIV

[2] A. J. Smith, "Cache memories," *ACM Computing Surveys*, vol. 14(3), pp. 473–530, Sep. 1982.

[3] J. D. Gee, M. D. Hill, and A. J. Smith, "Cache performance of the SPEC benchmark suite," University of California at Berkeley, CA, USA, Tech. Rep. CSD-91-648, 1991.

[4] H. Al-Zoubi, A. Milenkovic, and M. Milenkovic, "Performance evaluation of cache replacement policies for the SPEC CPU2000 benchmark suite," in *Proceedings of the 42nd annual ACM Southeast regional conference*, Huntsville, Alabama, 2004.

[5] J. F. Cantin and M. D. Hill, "Cache performance for selected SPEC CPU2000 benchmarks," *ACM SIGARCH Computer Architecture News*, vol. 29(4), pp. 13–18, Sep. 2001.

[6] A. Kennedy, M. Alexander, E. Fiene, J. Lyon, B. Kuttanna, R. Patel, M. Pham, M. Putrino, C. Croxton, S. Litch, and B. Burgess, "A G3 PowerPC superscalar low-power microprocessor," in *Proceedings of COMPCON 97*, 1997.

[7] M. J. Flynn, *Computer Architecture - Pipelined and Parallel Processor Design*. Jones and Bartlett Publishers Inc., 1998.

[8] *ARM1136JF-S and ARM1136J-S, Technical Reference Manual*, ARM Limited, 2002.

[9] *MSC8144 Quad Core Digital Signal Processor*, Freescale Semiconductors, Apr. 2008.

Session 9A

VLSI Education

Infrastructures for Education, Research and Industry in Microelectronics - a look worldwide and a look at India

B. Courtois, K. Torki, S. Dumont, S. Eyraud, J-F Paillotin, G. di Pendina
CMP - 46 Avenue Felix Viallet, 38031 GRENOBLE Cedex, FRANCE
email: Bernard.Courtois@imag.fr

Abstract

Infrastructures to provide access to custom integrated hardware manufacturing facilities are important because they allow Students and Researchers to access professional facilities at a reasonable cost, and they allow Companies to access small volume production, otherwise difficult to obtain directly from manufacturers. This paper is reviewing the most recent developments at CMP like the introduction of a CMOS 45nm process, the cooperation between the major infrastructures services available worldwide and recent developments w.r.t. India. The conclusion is addressing technical developments as well as considerations like globalization and excellence.

1. The need for infrastructures

Infrastructures to provide access to custom integrated hardware manufacturing facilities are important for several reasons:

- they allow Students and Researchers to access professional facilities at a reasonable cost,
- they allow Companies to access small volume production, otherwise difficult to obtain directly from manufacturers.

The needs of Universities, Research Laboratories and Companies can be summarized as follows:

- Universities need to have access to technology for teaching their students. Those students will be in the industry. Therefore they have to be trained at least on the actual state of the art technology processes.
- Research Laboratories usually need to have high performance technologies to validate new concepts. The quality of the research results depends mostly on the quality of the technologies. Accessing to up to date technologies is a necessity.
- Industrial users also need to access to the state of the art of the offered technologies. This is vital for industrial users. The development of a product is usually long (more than 1 or 2 years). It is necessary that an industrial user has access to an up to date process, giving guaranty on product life.

Offering manufacturing Services has to be governed by the following basic principles:

- Industrial quality process lines should be used (University process lines cannot offer a stable yield),
- Design kits to link CAD and MPC/MPW facilities should be offered to ease the design.

It is also to be noted that almost every country or continent is a high cost country or continent to another one at the time of global markets. It is thus important to keep students, researchers, industrialists ahead of others in order to stay in business. To stay ahead means to train, research, use, advanced design methods and tools and to design on advanced processes.

2. Developments at CMP

A review of early national efforts can be found in [1], and a review of first cooperative initiatives may be found in [2]. Since 1981, several periods may be distinguished at CMP:

Development at CMP

Several periods may be distinguished.

1981–1982	:	launching CMP with NMOS
1983–1984	:	development of NMOS, launching CMOS
1984–1986	:	development of CMOS
1987–1989	:	abandon NMOS, increase the frequency of CMOS runs
1990–1994	:	launching Bipolar, BiCMOS, GaAs MESFET, GaAs HEMT, advanced CMOS (.5 µ TLM)
1995–1997	:	launching CMOS and GaAs compatible
MEMS,		DOEs, deep-submicron CMOS (.25µ 6LM)
1998	:	launching silicon surface micromachining, abandon MESFET GaAs
1999	:	launching SiGe, deep submicron CMOS (.18µ 6LM), SOI/SOS CMOS (.5µ)
2000	:	launching SiGe BiCMOS (.35µ 5LM)
2001	:	launching very deep submicron CMOS (.12µ 6LM)
2002	:	launching Inp HBT process
2003	:	launching 0.35µ CMOS-Opto
2004	:	launching very deep submicron CMOS (90nm, 7LM), HBT Sige:C BiCMOS 0.25µ
2006	:	launching CMOS 65nm (7LM)
2008	:	launching CMOS 45nm

Processes available

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.17

Presently the processes available for ICs and MEMS manufacturing are depicted in Table 1.

Austriamicrosystems	0.35µ CMOS C35B4C3
	0.35µ CMOS C35B4M3
	0.35µ CMOS-Opto C35B4O1
	0.35µ CMOS Flash C35B4E3
	0.35µ SiGe BiCMOS S35D4M5
	0.35µ HV-CMOS H35B4D3
STMicroelectronics	45nm CMOS CMOS045
	65nm SOI
	65nm CMOS CMOS065
	90nm CMOS CMOS090
	0.12µ CMOS HCMOS9GP
	0.12µ SOI
	0.25µ SiGe:C BiCMOS7RF
OMMIC	0.2µ HEMT GaAs ED02AH
SANDIA	SUMMiT
MEMSCAP	PolyMUMPs SOI MUMPs Metal MUMPs

TABLE 1: IC AND MEMS PROCESSES AVAILABLE

ICs design kits and CAD software

Design kits and libraries are distributed by CMP for most of the processes and most commonly used CAD tools. CMP sometimes develop design kits, in cooperation with the manufacturers and the CAD vendors. CMP also offers special CAD software conditions from a few CAD vendors. As a focal point, CMP also distributes information on configuration files, converters, etc. About 40 design kits are available for each process and the main CAD tools.

Other services

Packaging and testing services are also offered. Various types of packages are supported, including DIL, SOIC, CQFP, JLCC, PGA, etc. Test of prototypes is usually done by the final user. On request, especially for low volume production, CMP may take over testing together with manufacturing.

Key figures

Since 1981, CMP has served more than 1000 Institutions from 66 countries in various processes. Support to Industry started in 1993 for small volume production. CMP is ISO 9002-1994 certified from 2000 to 2003. CMP is working on the certification ISO 9002-2000.

Recent developments

Recent developments have been the move to very deep submicron processes: 120nm CMOS, 90nm CMOS, 65nm CMOS and 65nm SOI, 45nm CMOS, .35µ HBT SiGe BiCMOS, .25µ Sige:C HTB BiCMOS from STMicroelectronics and the exploration of new MEMS fabrication offerings.

The move to very deep submicron processes.

CMP introduced 120nm CMOS as early as 2001. A total of 175 circuits were fabricated from 2001 to June 2007. CMP introduced 90nm CMOS in 2004 and nearly 100 circuits have been fabricated up to now. Finally 65nm CMOS was launched in 2006 and a ten of circuits have been fabricated already. This means a total of nearly 300 circuits coming from about 50 Research Laboratories and Industrial Companies. These processes have been very well received. Let's detail what happened with the CMOS 90nm. The 90nm CMOS has been announced in 2004, first DRMs and design kits have been shipped to designers in 2004. The list of Institutions who have used the 90nm CMOS to date is depicted in Table 2. One can notice a number of top level Universities in Europe and North America mostly. All Canadian Universities are using the 90nm CMOS process. The move to 65nm has started. The 65nm CMOS has been announced in 2006. The Table 3 depicts the list of Institutions who have received the DRMs and design kits up to now. Again there are many top level Universities in Europe and in North America who are moving to 65nm CMOS. About 30 Institutions have received the 45nm design rules.

Institution	Town	Country
U. of Calgary	Calgary	CANADA
U. of Waterloo	Waterloo	CANADA
Carleton U.	Carleton	CANADA
U. of British Columbia	Vancouver	CANADA
U. of Toronto	Toronto	CANADA
Ecole Polytechnique de Montréal	Montréal	CANADA
McGill U.	Montréal	CANADA
CMC Microsystems	Kingston	CANADA
U. of Alberta	Edmonton	CANADA
Queen's U.	Kingston	CANADA
Techn. U. of Denmark	Lyngby	DENMARK
VTT	Espoo	FINLAND
IMEP	Grenoble	FRANCE
ISEN	Lille	FRANCE
U. Stuttgart	Stuttgart	GERMANY
U. of Modena	Modena	ITALY
INFN	Pavia	ITALY
U. degli studi di Pisa	Pisa	ITALY
Novelda AS	Kviteseid	NORWAY
U. of Oslo	Oslo	NORWAY
Norwegian U. of Sc & Technol.	Trondheim	NORWAY
Instituto Microelectronica	Sevilla	SPAIN
Linkoping U.	Linkoping	SWEDEN
U. of Neuchatel IMT	Neuchatel	SWITZERLAND
ETH Zentrum IIS	Zurich	SWITZERLAND
U. of Michigan	Ann Arbor	USA
UC Berkeley - BWRC	Berkeley	USA
MIT	Cambridge	USA
U. of Virginia - DECE	Charlottesville	USA
Georgia Electronic Design Center	Atlanta	USA
UCLA - El. Eng. Dept.	Los Angeles	USA
Sun Microsystems Inc.	Mountain View	USA
U. of Washington	Seattle	USA
Stanford U.	Stanford	USA
TOTAL	**34 Institutions from 11 countries**	

TABLE 2: INSTITUTIONS HAVING SUBMITTED CIRCUITS 90NM CMOS

AUSTRALIA	:	1
BELGIUM	:	4
BRAZIL	:	2

CANADA	:	16
DENMARK	:	3
FINLAND	:	2
FRANCE	:	27
GERMANY	:	7
ITALY	:	15
JAPAN	:	2
KOREA	:	1
NETHERLANDS	:	2
NORWAY	:	2
SINGAPORE	:	1
SOUTH AFRICA	:	1
SPAIN	:	7
SWEDEN	:	3
SWITZERLAND	:	4
TUNISIA	:	1
UNITED ARAB EMIRATES	:	1
U.K	:	6
USA	:	20
TOTAL: 128 Institutions from 22 countries		

TABLE 3: INSTITUTIONS HAVING RECEIVED THE 65NM DRMS & DESIGN KITS

Advanced MEMS processes.

CMP offered manufacturing for MEMS as early as 1995, being the first service to offer MEMS. It is important that MEMS can be obtained from the same service delivering ICs, so that integration (either at the packaging level or at the die level) is made easier.

For many years, CMP has been offering the MUMPS processes from MEMSCAP: PolyMUMPS, MetalMUMPS, SOIMEMS. These MUMPS services enjoy a 10 years MEMS experience, obtained through more than 70 runs, more than 200 designs. Recently CMP has been introducing SUMMITV from SANDIA and to a MEMS process based on the CMU post process capabilities.

The SUMMIT (Sandia Ultra-planar Multi-level MEMS Technology) fabrication process is a five-layer polycrystalline silicon surface micromachining process (one ground plane/electrical interconnect layer and four mechanical layers). The MEMS structures made possible by this five-layer planarized surface micromachining process are extremely diversified.

CMU post process is a post CMOS processing from Carnegie Mellon University. This post process, applied to an advanced process proposed by CMP (0.35 SiGe BiCMOS from STMicroelectronics), allows to combine on the same chip MEMS structures (resonnators, cantilevers, accelerometers, etc.) and microelectronics structures of an advanced BiCMOS process.

CMP also introduced long ago low cost bulk post process micromachining based on a CMOS processes.

3. Other major infrastructures

There are 7 major infrastructure services today: CIC in Taiwan, CMC in Canada, CMP in France, ICC in China, IDEC in Korea, MOSIS in the USA and VDEC in Japan. They are briefly described below. Three of them, CMC, CMP and MOSIS have decided in 2002 to cooperate. It might happen that further cooperations will be developed later on. The following depicts these 7 services as per their inputs to the 2007 CMP Annual report.

CIC

National Chip Implementation Center (CIC) Project was initiated by the National Science Council in 1992. This project aims to pave the way for a national research and service center for IC/System design.

In 2007, the process technologies provided by CIC are listed below

- UMC 90nm MS CMOS,
- TSMC 0.13µm MS/RF CMOS,
- TSMC 0.13µm Logic/MS CMOS,
- TSMC 0.18µm 1P6M CMOS,
- TSMC 0.35µm 2P4M CMOS,
- TSMC 0.35µm SiGe BiCMOS,
- WIN 0.15µm PHEMT GaAs,
- TSMC 0.35µm CMOS MEMS Post Process,
- TSMC 0.18µm CMOS MEMS Post Process.

Totally, there were 1721 prototyped ICs being successfully fabricated in 2007. Up to now, CIC has provided MPC services for over eighty universities in Taiwan with 10,029 prototyped ICs including 9,044 ICs from the academics (universities and polytechnics) and 985 ICs from the research institutes as well as industrial sectors. Furthermore, CIC also develop Sip module and CMOS MEMS technologies for academia.

CMC

CMC Microsystems (www.cmc.ca) provides national infrastructure for microsystems research and technology development.

As of 2007, CMC's services include:

- 65-nanometre CMOS (STMicroelectronics through CMP)
- 90-nanometre CMOS (STMicroelectronics through CMP)
- 0.13µ CMOS (IBM through MOSIS)
- 0.18µ CMOS (TSMC through MOSIS)
- 0.35µ CMOS (TSMC through MOSIS)
- 0.8µ CMOS in three process flavors: high-voltage--up to 300V, mid-voltage range--+/-20V, and standard-voltage--2.7V to 5.5V (DALSA Semiconductor).
- 2.5 GHz Bipolar linear array (Gennum Corporation)
- PolyMUMPs surface micromachining process (through partnership with CMP and MEMSCAP)
- MetalMUMPs (through partnership with CMP and MEMSCAP)
- Micragem SOI-based micromachining process (Micralyne Generalized MEMS process)
- Protolyne for semi-custom microfluidics devices (Micralyne)

- Photonics/optoelectronics: InP, GaAs, EPI-only InP/GaAs, Silica/Si and Silicon-on-Insulator based technologies (through Canadian Photonics Fabrication Centre)
- On an exploratory basis: Microfluidic process with metallization (through Micronit)

A total of 378 designs were fabricated in 2007 using the technologies listed above.

ICC

Founded in 2000 by Science and Technology Commission of Shanghai Municipality, Shanghai Research Center for Integrated Circuit Design (so-called ICC) is dedicated in promoting Shanghai and all China IC Design industry to realize durative rapid development. The services ICC provides include Multi-Project Wafer service, SoC design platform, testing service, training and evaluation, information service, etc. From 1996 to 2000, Shanghai MPW Service (SMS), operated by Fudan University, was mainly open to academic users, with totally 116 designs fabricated. From 2001, ICC began to operate SMS, expanded the service to industrial sectors and became the China National MPW Center. Totally 877 designs from more than 250 design houses, universities and research institutes were prototyped on MPW runs and low volume production since 2001.

The following technologies were available in SMS in 2007:

- CSMC 0.6um CMOS
- Chartered 0.35um CMOS
- Chartered 0.25um CMOS
- Chartered 0.35um SiGe
- TSMC 0.13um CMOS
- TSMC 0.18um CMOS
- TSMC 0.25um CMOS
- TSMC 0.35um CMOS
- SMIC 0.13um CMOS
- SMIC 0.18um CMOS
- SMIC 0.35um EEPROM
- HJTC 0.18um CMOS
- HJTC 0.25um CMOS
- HJTC 0.25um EEPROM

There are totally 22 runs in the year of 2007. 158 chips from 75 customers were successfully fabricated. In 2007, ICC provided an SoC design support platform with the cores from ARM, ZSP, Synopsys and etc, such as ARM7TDMI, ARM926EJ, NEO, ZSP200, ZSP400.

IDEC

IDEC (Integrated Circuit Design Education Center) was launched in 1995 with the support of the Ministry of Commerce, Industry and Energy and major semiconductor industries for the purpose of educating designers in the non-memory IC field.

Currently, IDEC provides MPW services for 62 WGs (Working Groups) in Korea. As of January 2008, a total of 1,814 IC chips have been successfully fabricated through the IDEC MPW (Multi-Project Wafer) program. The technologies provided in 2007 are listed below:

• CMOS 0.35 μ, 1-poly 4-metal, Samsung Electronics
• CMOS 0.18 μ, 1-poly 4-metal, Samsung Electronics
• CMOS 0.35 μ, 2-poly 4-metal, Magnachip/Hynix
• CMOS 0.18 μ, 1-poly 6-metal, Magnachip/Hynix
• CMOS 0.18 μ, 1-poly 6-metal, Dongbu Electronics
• InGaP HBT, Knowledge-on

MOSIS

MOSIS is a low-cost prototyping and small volume production service for VLSI circuit development with a worldwide customer base. Since 1981, the service has fabricated more than 50,000 integrated circuit designs for use by commercial firms, government agencies and universities and has served as the model for similar operations throughout the world. It is a not-for-profit organization started in 1980 by DARPA (Defense Advanced Research Projects Agency of the U.S. Department of Defense) at the Information Sciences Institute to provide their research community with access to advanced IC fabrication lines in a cost effective manner. Fast-turnaround prototype and low-volume fabrication of integrated circuits is available through a number of major commercial IC fabrication vendors such as Agilent/HP (now Avago) Technologies (0.5μ CMOS), AMI Semiconductor (0.35μ, 0.5μ, 0.7μ, 1.5μ CMOS), IBM (65nm, 90nm, 0.13μ, 0.18μ and 0.25μ CMOS; 0.13μ, 0.18μ, 0.25μ, 0.35μ and 0.5μ SiGe BiCMOS) and TSMC (0.13μ, 0.18μ, 0.25μ, 0.35μ CMOS). CMOS-compatible MEMs technologies are also available. Other technologies such as austriamicrosystems (0.35μ CMOS, 0.35μ HV CMOS, 0.35μ SiGe BiCMOS) are available through a partnership with CMP in France.

VDEC

VLSI Design and Education Center (VDEC), which is located in the University of Tokyo, has been utilized by academic users in Japan since its foundation in May, 1996. As an MPC service center, VDEC aims at improvements of education on VLSI design and supports on VLSI chip fabrication for national universities, public universities, private universities and colleges in Japan. VDEC receives a lot of supports from Japan government, as well as semiconductor industries through STARC (Semiconductor Technology Academic Research Center).

Presently the following technologies are available for

chip fabrication service.

- 2-poly 2-metal CMOS 1.2 µm process from SCG Japan Ltd. (OnSemiconductor Ltd.)
- 1-poly 5-metal CMOS 0.18 µm process from Rohm Co. Ltd.
- 2-metal 0.8µm bipolar process from NEC Compound Semiconductor Devices Ltd.
- 1-poly 6-metal CMOS 90 nm from ASPLA
- VDEC-MOSIS CMOS 0.25µm /0.18µm from TSMC
- VDEC-MOSIS Si-Ge BiCMOS 0.5µm from IBM

In last VDEC fiscal year (2006.4 – 2007.3), there were totally 24 chip fabrication runs in the last year, each with a 2 to 3 months period. 103 professors and research groups from 55 universities and colleges participated chip design and fabrication through VDEC. Totally 433 chips on 5397 mm^2 silicon area were designed and fabricated.

4. Present cooperative efforts

Presently, the major cooperative effort is undertaken by CMC, CMP and MOSIS. These 3 infrastructure services announced it at DAC in June 2002. Since then, the cooperation has been steadily expanding [3]. Recently, CMP has entered an agreement with IDEC to serve Korean Universities with STMicroelectronics processes. Separately, CMP has also set up a number of bilateral cooperation with infrastructure services, with special groups in various countries, and has established distributors in several parts of the world.

5. A look at india

Up to now, India Universities/Research Centers made modest use of CMP services. A few Instructions have received the Austriamicrosystems design rules including IIT Kanpur and 2 have received the 90nm design rules: Vellore Institute of Technology in Chennai and Birla Institute of Technology and Science in Pilani. IIT Kanpur is finalizing a design for manufacturing in September 2008 in the .35µ CMOS process. Interestingly, a University of the Philippines is finalizing a circuit in the 90nm process for manufacturing in July 2008. These are positive signs of moves from India and other countries who are not yet in the Table 2.

6. Conclusion

Key issues at CMP in 2007 have been:

- More and more circuits: +25% from 2005 to 2006, +22% from 2006 to 2007
- Industrial circuits is maintained to about 20% of the total number of circuits and low volume production is provided up to tens of wafers
- A large portfolio of technologies (17 different processes from low cost processes to very advanced ones) for ICs and MEMS
- Cooperation continues with other major services in the world

Several general conclusions are addressed in the following, according to 3 broad lines:

- more Moore
- more than Moore
- more than more than Moore

and 2 considerations:

- going global
- being excellent

More Moore

It has been recognized that Students, Researchers and SME designers must be provided with the possibility to have their circuits fabricated. From its inception in 1981, CMP has been successfully pursuing this goal and experiencing a very significant growth to reach and to keep its present level. The success is partly due to the basic principles which have been governing the choices of the Service: use of industrial and advanced process lines. Advanced processes are more and more necessary because of the need for very skilled designers and because CAD industrial software is more widely available to Universities (instead of University CAD software). Since new versions of CAD software are targeted to industrial use, there is no choice but to use advanced processes. Industry makes also more and more use of the Service. During the 80s, the CMP processes were not very advanced, but they approached more and more industry state of the art during the 90s, because of CAD software reasons and because of the increasing industry use of CMP. Since then, CMP is always offering state of the art processes.

More than Moore

The quest for always larger densities may also be satisfied with 3D processes, possibly not including very advanced process dies. 3D processes lead to easier to manage interconnections and to reasonable cost. CMP will introduce soon 3D processes using TSVs (through silicon vias).

It is also recognized that complementary developments must be addressed, in order to address more diversified needs. With this respect, CMP has been a pioneer in being the first service in the world to offer MEMS processes as early as 1995. Going further, more than mechanics-electronics is to be addressed like photonics, optics, fluidics, etc. CMP will be actively promoting these developments in the future.

More than more than Moore

Going further beyond, other communities than EE and CS should be addressed, for which electronics will

offer more and more opportunities in the future. CMP has started to address the BioMed community [4]. Many kinds of BioMed applications are addressed in this paper, ranging from neurosciences to surgery aid, to endoscopy, to skin treatment. Many other kinds of applications might be devised in the future. Going further from dermatology for example, hardware devices might be designed in view of the coming market dealing with dermonutrition or nutricosmetics, depending on the way companies are coming from. Danone is offering yoghurts "nourishing the skin from inside", and L'Oreal is offering with Nestle nutritional food fighting the skin aging: nutraceuticals with cosmetic benefits (the so-called beauty pills). In both cases, the efficiency can be scientifically measured by specific devices.

What is important for the BioMed community is that Education and Research should take advantage of these infrastructures, in the same way as Education and Research in microelectronics have taken advantage of these infrastructures in the 80s. At that time, these infrastructures offered the possibility to EE and CS students, teachers, researchers, to focus on the design of complex circuits hence to focus on the applications, because these infrastructures gave them the opportunity not to be burden by the manufacturing processes, nor by the cost of their projects. Today, various CMOS and MEMS processes can allow students, teachers, and researchers to focus on BioMed applications. Not all possible applications can be reached by standard processes offered by service organizations like CMP, but many can be addressed. Other communities could take advantage of such services in the same way.

Going global

All the above requires cooperation between services like CMP, since it is difficult for one single service to offer a wide set of processes of its own. CMP had set up a cooperation with CIC (Taïwan) long ago. Recently, CMC (Canada), CMP and MOSIS (USA) announced a reinforcement of their cooperation in 2002. More recently CMP announced a cooperation with ICC (China) and IDEC (Korea). It is expected that CMP will reinforce such cooperations in the future.

On the side of the users of CMP and of other similar services, the design way is also going more and more global in the sense that more and more IP blocks may come from various sources. This is due to the ever increasing complexity of designs, including parts coming from various teams, countries, companies, etc. An initiative is being developed to go this way: the Global Education for Microelectronic Systems (GEMS) [5].

Being excellent

Globalization also requires to be excellent in order to stay ahead of others. This is important at the time of global markets, when every country or continent is a high cost country or continent to another one. Some countries or continents that were said to be "low-cost" countries or continents a few years ago already experience that other countries or continents are coming to the picture with lower costs, forcing them to outsource their own outsourcing [6]. The way to combat that is to stay ahead of the others. The way to stay ahead is to educate and research using top level electronic processes available from top level services like CMP.

7. References

[1] "MPC Services available worldwide", invited paper, APCCAS'94 IEEE Asia-Pacific Conference on Circuits and Systems, December 5-8 1994, Grand Hotel, Taipei, Taiwan.

[2] "Infrastructures for Education and Research: from National Initiatives to Worldwide Development", invited talk, Technical University of Darmstadt, M. GLESNER 60th birthday ceremony, 29 August 2003.

[3] CMC, CMP and MOSIS, "The Scale of Cooperation Increases as the Dimensions of Microchips Decrease", invited paper, 3rd International Conference on Microelectronic Systems Education, 1-2 June 2003, Anaheim, USA.

[4] COURTOIS B., CHARLOT B., DI PENDINA G., RUFER L., Infrastructures for Education, Research and Industry: CMOS and MEMS for BioMed, *Invited paper* at the 12th World Multi-Conference or Systemics, Cybernetics and Informatics (WMSCI 2008), Orlando USA, 29 June – 2 July 2008.

[5] RUCINSKI A. "Global Education for Microelectronic Systems, www.ieee-gems.edu".

[6] India outsources its own outsourcing – Pioneer nation fends off new rivals, International Herald Tribune – 25 September 2007.

Invited Paper

Phase Locked Loops

Specification driven design of Phase locked loops

Prakash Easwaran, Prasenjit Bhowmik, Rupak Ghayal

Cosmic Circuits Pvt. Ltd.

Bangalore, India

prakash@cosmiccircuits.com

prasenjit@cosmicciruits.com

rupak@cosmiccircuits.com

Abstract— The factors that impact the topology and the performance specifications for a Phase locked loop (PLL) is presented. Correct specification of the PLL is critical for optimizing the performance and power of the system. PLL specifications for different systems have been derived and the architectural tradeoffs have been discussed. Three PLL design examples have been presented for WLAN base-band PLL application, DVB-H receiver base-band PLL application and a high speed (MIPI) transmitter application.

Index terms— Charge-pump, Phase locked loop (PLL), jitter, phase noise.

I. INTRODUCTION

Phase locked loops (PLL) are used to implement different kind of timing related functionality such as frequency synthesis [1], clock and data recovery [2] and clock de-skewing . The output of such a PLL may be used in variety of applications like analog sampling, system clock generation, high speed serial links etc. In this paper the architectural choice and design approach have been presented for PLLs for different applications.

The paper is organized as follows. Section II provides an introduction to the various PLL topologies and the advantages and disadvantages of each topology. Section III presents various figures-of-merit for a PLL. Section IV discusses the various noise sources in a charge-pump PLL. Section V presents the performance requirements for PLLs in a variety of application. Section VI presents the design aspects of charge-pump PLLs and the conclusion is given in Section VII.

II. PLL TOPOLOGIES

The PLL is a negative feedback loop where the feedback clock edge is aligned to the input clock edge [3]-[6],. The most popular architectures of choice are the charge-pump architecture [7] and digital architecture [8].

Fig. 1 shows a typical block diagram of a charge-pump PLL. The Phase Frequency detector (PFD) generates a pulse-width modulated signal, where the pulse width is proportional to the timing difference between the reference clock (F_{ref}) edge and the feedback clock (F_{fbclk}) edge. The Charge Pump (CP) output is a current pulse whose pulse-width is determined by PFD. The loop-filter converts the current pumped by CP into voltage where the voltage is proportional to timing difference between two clocks at PFD input. The voltage controlled oscillator's (VCO) frequency changes proportionally to the applied correction-voltage.

At steady state (for an ideal charge-pump PLL) ,VCO input voltage is static, CP output current is zero, Timing error between clock-edges at PFD input is zero and hence $F_{fbclk}=F_{ref}$. Now $F_{fbclk}=F_{sys}/N$ where N is the division number. This implies $F_{sys}=N*F_{ref}$

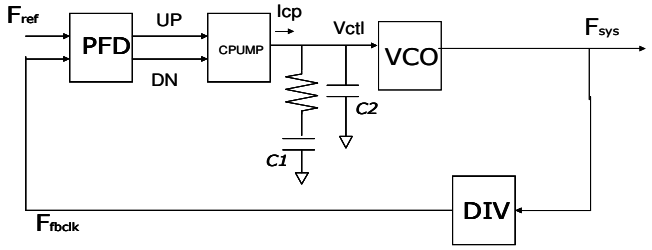


Fig. 1 Block diagram of a charge-pump PLL

Fig.2 shows a typical block diagram of a digital PLL (DPLL).

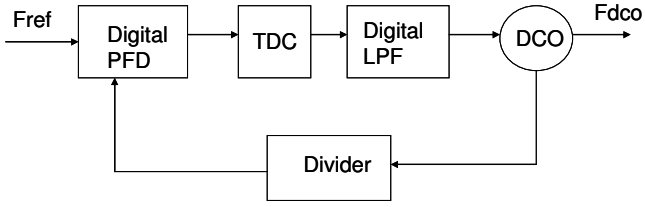


Fig. 2 Block diagram of a DPLL

PFD for DPLL is similar to the PFD for a charge-pump PLL. The time to digital converter (TDC) is equivalent to Time to CP current in a charge-pump PLL. The resolution of

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.97

the TDC limits the accuracy to which phase can be aligned between reference and feedback clocks. The digital LPF is synthesized as Z domain transfer function equivalent of loop filter of a charge-pump PLL. Generally, frequency of operation of loop filter is F_{ref}. The VCO in a charge-pump PLL is replaced by as DCO in the case of digital PLLs. The DCO normally consists of a digital-to-analog converter (DAC) and a current controlled oscillator (ICO). Fig.3 shows a typical DCO configuration.

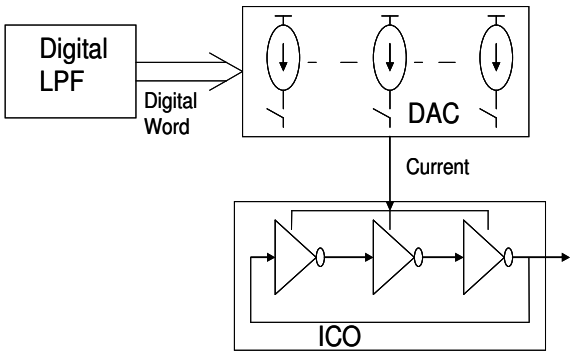


Fig. 3 Block diagram of a DCO

The resolution of the DCO is defined by the gain of the ICO and the resolution of the DAC. This kind of PLL normally gives higher jitter compared to a charge-pump implementation as the LPF has a finite resolution and increasing the number of bits requires exponential increase in the DAC elements. Matching becomes a big problem in a DAC array with large number of elements. A sigma-delta DAC can be implemented to overcome these issues to get a good resolution out of a DCO.

Comparing the two architectures, it can be seen that the digital implementation has a frequency quantization error in addition to all the noise sources of a charge-pump PLL. Due to this a digital PLL exhibits more frequency accuracy error compared to a charge-pump PLL. In terms of phase error, the frequency error accumulates for one cycle of reference clock. Also the TDC resolution adds to the long term phase error as the loop doesn't react when the phase error is within the resolution of TDC.

Charge-pump PLLs usually need an external capacitor in the loop filter for very low bandwidth applications. The advantage of using a digital architecture is that it can get rid of the external component in such applications. Also the PLL loop parameters like the bandwidth can be very well controlled with initial calibration. A digital PLL is less sensitive to dc drifts and reduces design time because of easy portability. These advantages make this architecture more amenable to processor clocking applications where long term phase error is not of importance (described in section V) and low bandwidth applications whereas charge-pump PLLs are preferred where long term phase error specifications is more critical.

As discussed, the basic structure of both these topologies is similar and hence all the analysis in this paper will be restricted to charge-pump PLLs which is more popular. The analysis can be very easily extended to digital PLLs.

III. PLL Performance Metrics

PLL performance specifications can be clubbed into following categories.

A. Lock time

This is defined as the minimum time required after power up for the output to be usable. The time can be specified as frequency locking time which is defined as the time required for the output to reach within a predefined accuracy. The bound depends on the application. Frequency lock time specifications are pertinent to PLLs which supply the clock for digital systems.

The time can also be specified as phase locking time which is defined as the time required for the output phase errors to reach within a permissible limit. Phase lock time specifications are pertinent to PLLs which supply the clock for analog sampling applications.

B. Clock stability metrics

An oscillatory signal can be represented as

$$V(t) = VSin(\omega t + \phi)$$

The phase of the signal can be represented as

$$\Phi(t) = \omega t + \phi(t)$$

If $\phi(t)$ is time variant, the zero crossings of $V(t)$ will have an uncertainty. The measure of this uncertainty is represented as jitter in time domain and phase noise in frequency domain.

The instantaneous frequency is given by

$$\Omega(t) = \omega + \frac{d\phi}{dt}$$

1) Jitter

The uncertainty of the zero crossings of an oscillatory signal is measured as jitter in time domain [9]. There are different ways to specify jitter. The different types of jitter can mainly be classified as period jitter, cycle-to-cycle jitter and time interval error which is also classified as long term jitter. Each of them can be measured in terms of their root-mean-square (rms) value or peak to peak value. If the jitter is caused by random noise, it is advisable to specify it in terms of its rms value. If the jitter is caused by some deterministic pattern, then a peak-to-peak number represents the jitter distribution in a better way. In any PLL system, the jitter will be caused due to combination of both types of noise and hence it is better to split the contribution and specify the rms number caused due to random noise along with the peak-to-peak number caused due to a deterministic noise. The random contribution number can be converted to a peak-to-peak number by multiplying the rms number by a suitable number depending on the

application. In most cases +/-3 time the rms jitter is sufficient whereas in clock data recovery applications where a bit error rate (BER) of $1e^{-14}$ needs to be achieved, +/-7 times the rms jitter is taken as peak–to-peak. The three basic types of jitter are defined below:

1. Period jitter: This is measured as the variation of time period over a sufficiently large number of samples.
2. Cycle-to-cycle jitter: This is measured as the variation of the difference between successive time periods over a sufficiently large number of samples.
3. Long term/accumulated jitter: This is measured as the deviation of the zero crossing with respect to an ideal zero crossing over sufficiently large samples.

2) Phase noise

An oscillatory signal can have a phase modulation in two ways:

1. Noise modulating the frequency of the signal and hence the phase e.g. noise current in a ring oscillator. An oscillatory signal with this kind of noise can be written as

$$V(t) = V_m Sin\left(\int_0^t (\omega + \omega_m Sin\ \omega_n t)dt\right) \quad (1)$$

Using (1)

$$V(t) = V_m Sin\left(\omega t + \frac{\omega_m Cos\ \omega_n t}{\omega_n}\right) \quad (2)$$

Expanding (2) assuming the modulation is very small

$$V(t) = V_m Sin\ \omega t + \frac{V_m f_m}{2 f_n} Cos\ (\omega t + \omega_n t) + \frac{V_m f_m}{2 f_n} Cos\ (\omega t - \omega_n t) \quad (3)$$

2. Noise directly modulates the phase without affecting the frequency e.g. noise current in a clock slicer, buffers or dividers. An oscillatory signal with this kind of noise can be written as

$$V(t) = V_m Sin\ (\omega t + \phi_n Sin\ \omega_n t) \quad (4)$$

$$V(t) = V_m (Sin\ \omega t + Cos\ \omega t . \phi_n Sin\ \omega_n t) \quad (5)$$

$$V(t) = V_m Sin\ \omega t + \frac{V_m \phi_n}{2} Sin\ (\omega t + \omega_n t) + \frac{V_m \phi_n}{2} Sin\ (\omega t - \omega_n t) \quad (6)$$

Equation (3) and (6) can be used to plot the modulated output spectrum for any type of modulating noise. Fig. 4 shows one particular case where the noise consists of flicker and white noise.

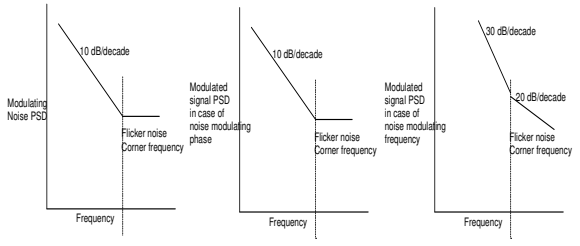


Fig. 4 Modulated output spectrum for phase and frequency modulation

The dominant phase modulation in an oscillatory signal close to the carrier happens due to frequency modulation

For an oscillatory signal $V(t) = V_m Sin\ (\omega t + \phi_n Sin\ \omega_n t)$ (7)

Phase noise at an offset f_n from the carrier is defined as [10]

$$L(f_n) = 10 * \log 10\left[\frac{\text{power in a 1Hz bandwidth at an offset of } f_n}{\text{power at the fundamental}}\right] \quad (8)$$

From (8) $$L(f_n) = \frac{\frac{(V_m . \phi_n)^2}{8}}{\frac{V_m^2}{2}} \quad (9)$$

From (9) $$L(f_n) = \frac{\phi_n^2}{4} \quad (10)$$

a) Relation between jitter and phase noise

Jitter and phase noise can be related by equating the noise energy of phase in frequency and time domain [11]

Noise energy in frequency domain is given by

$$\frac{T_0^2}{4\pi^2} \int_0^\infty S\phi(f)\, df \quad (11)$$

Where $S_\phi(f) =$ power spectral density of the phase error.

From (7) $$S\phi(f_n) = \frac{\phi_n^2}{2} \quad (12)$$

If the oscillatory signal is characterized in the time domain by a long term rms jitter of $\sigma(t)$, then the noise energies can be equated using (10) ,(11) and (12) to give

$$\sigma(t)^2 = \frac{T_0^2}{2\pi^2} \int_0^{+\infty} L(f)\, df \quad (13)$$

Equation (13) defines the relation between SSB phase noise and long term rms jitter with respect to an ideal clock.

In case of self referred jitter, the window opening of N cycles modifies the spectrum of $S\phi(f)$

Self referred N-cycle jitter is given by [11]

$$\sigma(NT)^2 = \frac{T_0^2}{\pi^2} \int_0^{+\infty} S\phi(f)\, Sin^2(\pi f N T_o)\, df$$

$$= \frac{T_0^2}{\pi^2} \int_0^{+\infty} 2.L(f)\, Sin^2(\pi f N T_o)\, df \quad (14)$$

Where $T_0 =$ ideal time period

$L(f) =$ SSB phase noise at an offset f from the carrier

Substituting N=1 in (14) the relation between period jitter and phase noise can be obtained as

$$\sigma_{per}(t)^2 = \frac{T_0^2}{\pi^2}\int_0^{+\infty} 2.L(f)\,Sin^2(\pi f T_o)\,df \qquad (15)$$

b) Jitter due to periodic noise

Fig.5 shows a periodic noise modulating frequency of a PLL output clock.

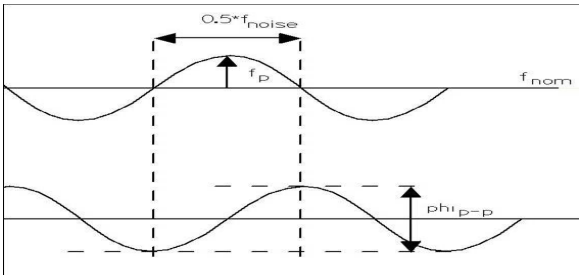


Fig. 5 Frequency modulation with a periodic pattern

The instantaneous frequency is given by

$$f_p Sin(2*\pi*f_{noise}*t) \qquad (16)$$

where f_p = peak frequency variation and f_{noise}= frequency of the periodic noise. Integrating (16) over an interval of $0.5*f_{noise}$. Peak to peak phase variation is given by $2*\frac{f_p}{f_{noise}}$.

Equation can be used to get the rms jitter as

$$\frac{f_p}{sqrt(2)*2*\pi*f_{noise}*f_{nom}} \qquad (17)$$

Where f_{nom}= nominal clock frequency

3) Reference spur

Reference spur is defined as the reference frequency component at the output of a PLL. This is mainly caused due to the static phase offset present in the charge-pump which is caused by the mismatch between the charge and discharge paths.

For a static phase offset of ϕ_e the reference spur is given by [12]

$$20*\log\left(\frac{\sqrt{2}*\frac{I_{cp}R}{2\pi}*\phi_e*K_{vco}}{2*f_{ref}}\right) - 20*\log\left(\frac{f_{ref}}{f_p}\right) \qquad (18)$$

Where I_{cp}= Charge-pump current

K_{vco}= VCO gain

R= loop filter resistor

f_{ref}= Reference clock frequency

f_p= Loop filter pole

IV. PLL Noise Sources

This section gives a qualitative understanding of the effects of various noise sources in a charge-pump PLL [13]. The basic block diagram of the charge pump PLL with the various noise sources is shown in Fig. 6.

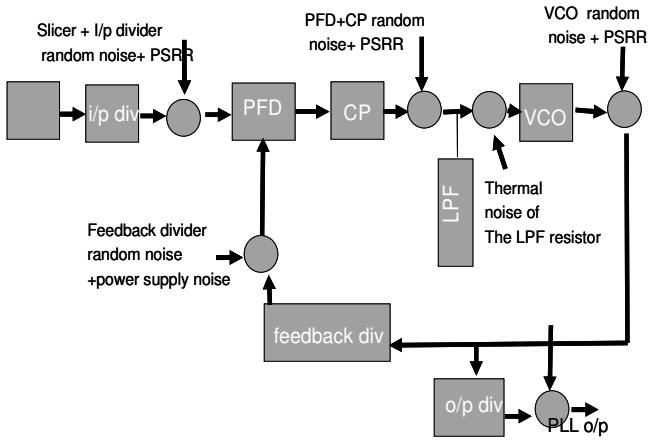


Fig. 6 Charge-pump PLL block diagram with noise sources

Any noise source occurring before the loop filter of the PLL gets low pass filtered (REFCLK noise, PFD + CP noise, Divider noise) and any noise source occurring at the VCO input (thermal noise of loop filter resistor) gets band-pass filtered and any noise source at the VCO output gets high pass filtered.

A. VCO noise

The phase noise of the VCO is typically the most dominant noise contributor in Charge Pump PLLs. This is so because most VCOs are ring oscillator based whose phase noise is inferior to that of LC oscillators. Also, in deep sub-micron processes the flicker noise corner frequency is much higher (hundreds of kHz) causing a high 1/f^3 corner frequency. Hence typically it is the oscillator's flicker noise that contributes maximum phase noise.

Power supply noise can also contribute significantly to jitter at the PLL output. The finite power supply rejection of the VCO changes the frequency which gets corrected by the loop. The noise can accumulate over multiple cycles till the loop correction takes place leading to both accumulated as well as period jitter.

Since the VCO noise gets high pass filtered by the PLL, it is desirable to keep as high a loop bandwidth as possible. A number of design techniques to reduce the phase noise and supply noise for various VCO topologies have been reported

B. *Loop filter resistor thermal noise*

The thermal noise of the resistor appears as a noise on the control voltage of the VCO and hence gets gained up by the VCO gain. The noise contributed gets band-pass filtered by the PLL transfer function. To minimize this noise contribution, the loop filter resistor value and the gain of the VCO needs to be minimized.

C. *PFD + Charge-pump noise*

The transistor noise in the phase frequency detector and charge pump also contribute to overall phase noise and sees a low pass transfer function. Among these the charge pump current noise can contribute significant noise especially in fractional PLLs where the charge pump currents are switched on for longer periods of time. In integer PLLs this noise source is typically much smaller since the time for which the current sources are switched on (determined by static phase offset) is as as low as 100 ps.

To minimize this noise contribution, the static phase offset is minimized by ensuring good matching of the current sources in the charge pump and ensuring negligibly small leakage in the loop filter capacitor. Also, the input current to the charge pump is heavily filtered.

D. *Divider noise*

The random noise contribution of the dividers is typically much smaller when compared to the deterministic noise due to supply noise and crosstalk. Multiple output dividers running at different frequencies share the same supply causes not of asynchronous switching noise on the supply voltage. Crosstalk due to other clock lines routed nearby also introduces periodic jitter.

To minimize the deterministic noise contribution from the dividers, the clock path lengths should be minimized with synchronizers using a low jitter clock (normally the VCO output). Care should be taken in layout to ensure adequate separation (or shielding) between clock lines of different frequencies.

The different noise transfer functions can be summarized as listed in Table I. Here for simplicity, the gains have been grouped into two sections. The forwards path gain comprises the PFD gain, charge-pump gain, loop filter impedance and VCO gain and has been clubbed as G1. The feedback divider has been specified as gain G2.

TABLE I. PLL NOISE TRANSFER FUNCTIONS

Block	NTF	Characteristics
Reference	G1/(1+G1*G2)	Low Pass
Slicer/PFD/CP	1/Kp*G1/(1+G1*G2)	Low Pass
VCO	1/(1+G1*G2)	High Pass
N-Divider	G1/(1+G1*G2)	Low Pass
Loop Filter	Not straight forward	~Band Pass

V. JITTER/PHASE NOISE REQUIREMENTS

PLLs find usage in different kinds of applications. These applications can be broadly grouped into following categories:

1. Sampling clock for data converters (ADC and DAC).
2. Frequency synthesis application for generating system clock from crystal oscillator.
3. Frequency modulation and demodulation needs in wireless communication systems.
4. Extracting clock from random bit stream in asynchronous serial link transmission systems.

This section describes the jitter and phase noise requirements for these four categories.

A. *Analog sampling applications*

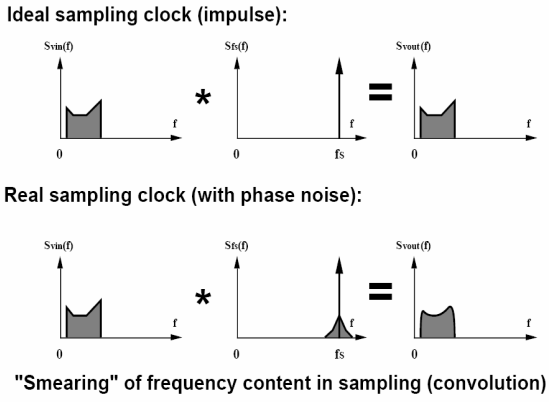


Fig. 7 Effect of phase noise in sampling

Fig. 7 shows the effect of phase noise while sampling a data. Fig. 8 shows the same thing in time domain.

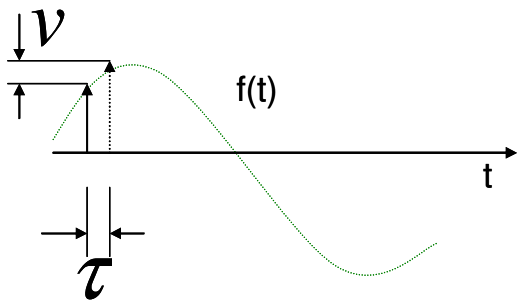


Fig. 8 Effect of jitter in sampling

In Fig.8 the signal is represented as $f(t) = A\sin(\omega 0 * t)$ (19)

From Fig. 8 $v = (df/dt) * \tau$ (20)

From (19) and (20) $v(t) = A * \omega 0 * \tau * \cos(\omega t)$ (21)

Equation (21) can be represented in frequency domain as

$v(f) = A * \omega 0 * \tau(f) * \delta(f - fo)$ (22)

From (22)

$$< v >= A * \omega 0 * < \tau > / sqrt(2) \qquad (23)$$

Equation (23) gives the noise power.

From (23) the SNR at the output of the sampling circuitry when the sampling clock has rms long term jitter of τ is given by

$$SNR = 20 * \log 10(\omega 0 * < \tau >) \qquad (24)$$

From (24) depending on the SNR requirement at the maximum signal frequency, the sampling clock jitter requirement can be found out. The sampling instant decides the sampled voltage and any deviation of the sampling instant with respect to an ideal sampling instant determines the error in the sampled value. The jitter of interest due to this reason is the deviations of the zero crossings with respect to an ideal zero crossing and hence the long term jitter decides the performance.

Table II gives the performance requirement for some typical applications:

TABLE II. JITTER REQUIREMENTS

System	ADC Spec	Maximum Signal Freq	Jitter spec
VDSL	14b 70MSPS	12MHz	2pS (SNR=76dB)
WLAN	10b 80MSPS	10MHz	15pS (SNR=60.5dB)
Base Station	12b 200MSPS	100MHz	350fS (SNR=73dB)

These applications require a maximum long term rms jitter specification to be met on the sampling clock. The input clock in most cases is provided from a clean crystal oscillator and VCO phase noise and power supply rejection becomes the dominant contributor. This mandates a PLL to be used with maximum bandwidth permissible by the input frequency and the VCO needs to have a low phase noise at its output. Also in most of the cases reference spur needs to be lower than a specified value. This mandates a charge-pump with low static phase offset.

B. System clock generation

In this kind of application the PLL provides clock for the digital core. In most cases data is latched at the rising edge and then other operations are performed and the data has to be stable before the next rising edge. The constraints that drive the clock purity in this kind of application is the frequency deviation or time period variation as the time window of interest is the difference between two consecutive edges of the clock. In this kind of application period jitter specification becomes the important parameter. Fig. 9 demonstrates the period jitter.

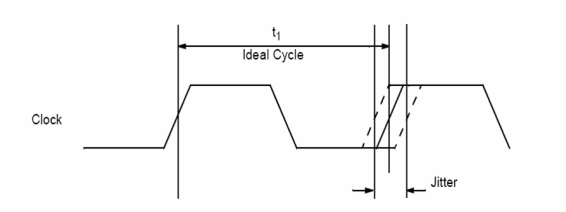


Fig. 9 Illustration of period jitter

Fig.10 shows the effect of period jitter on the data set-up time.

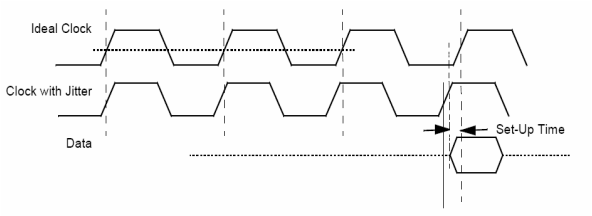


Fig. 10 Effect of period jitter

These applications require a maximum peak-to-peak period jitter specification to be met on the sampling clock. Input clock jitter, VCO phase noise, power supply rejection of the VCO and the jitter introduced by the delay variation due to power supply noise in any buffer path after the VCO becomes the dominant contributor. The PLLs for these applications are designed with a very low bandwidth to filter out any input jitter and also to prevent large corrections at the VCO control voltage. This doesn't introduce any stringent design requirement for the VCO as from (15), it can be easily seen that only the high frequency portion of the VCO phase noise contributes to the period jitter. An open loop oscillator's output phase noise can be represented as [14]

$$L(f) = \frac{a}{f^3} + \frac{b}{f^2} \qquad (25)$$

Using (14) and (25)

$$\sigma(NT)^2 = \frac{T_0^2}{\pi^2}\left[\int_0^{+\infty} 2.\frac{a.(\pi.N.T_0)^2}{f} Sinc^2(\pi fNT_0)df + \int_0^{+\infty} 2.b.(\pi.N.T_0)^2 Sinc^2(\pi fNT_0)df\right] \qquad (26)$$

The Sinc function has nulls at

$$f = \frac{m}{NT_0}; m = 1,2.... \qquad (27)$$

Equation (26) and (27) can be graphically represented as in Fig. 11

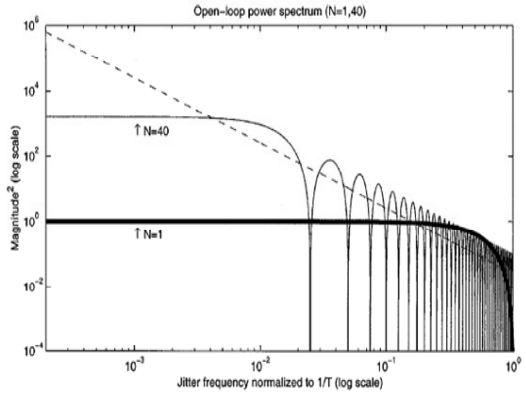


Fig. 11 Integrand plot for N=1 and N=40 [14]

Fig.11 clearly explains that only the high frequency portion of the phase noise contributes to period jitter.

Also care needs to be taken for reducing any buffer path delay to reduce the jitter introduced by them. In case long buffer delays are unavoidable, care has to be taken to make the buffer path immune to power supply noise.

C. RF front end applications

Fig. 12 shows the effect of sampling clock phase noise on receiver.

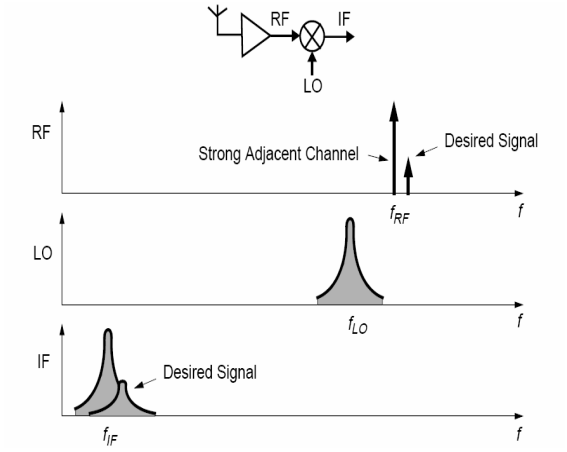


Fig. 12 Effect of phase noise in a RF receiver

Rx SNR:

Mixing of PLL phase noise and signal brings noise components from complete band due to convolution in frequency domain. This governs integrated phase noise of PLL.

Rx Interferers level with respect to desired signal:

Out of band interferers convolve with PLL phase noise and get converted to in-band noise during down conversion. Out of band phase noise is governed by this.

Tx Mask specification

Out of band PLL phase noise convolves with transmitted base-band signal and produces noise. Out of band phase noise is governed by this.

D. Clock data recovery applications

In clock data recovery applications in high speed asynchronous serial links, the PLL specifications are determined by the jitter generation mask, jitter transfer mask, jitter tolerance mask and the receiver BER.

The PLL needs to be designed to meet a maximum closed loop peaking number for meeting the jitter transfer characteristics. The jitter tolerance mask decides the PLL bandwidth. The jitter generation mask along with the BER requirement decides the VCO phase noise requirements.

VI. DESIGN ASPECTS OF CHARGE-PUMP PLLS

The first step in designing a PLL is to fix the loop parameters e.g. bandwidth, closed loop peaking etc. A phase domain model can be used to fix these parameters. The PFD gain is taken to be 1. All the dividers are set by the user depending on the input and output requirements. This leaves one with the task of choosing the VCO gain, charge-pump current and loop filter order and loop filter pole zero locations. The loop filter order is most of the times a second order filter with a zero unless the application demands more rejection of input noise (which can come either from the reference clock itself or from a sigma delta modulator used in fractional-N PLLs). Even in the case of higher order loop filters , till 3 times bandwidth the characteristic is governed by a 2nd order kind of filter. So for all practical purposes the loop parameters can be chosen assuming a 2nd order loop filter with a zero.

The PLL bandwidth is chosen to be less than 0.1 times the reference clock frequency. The bandwidth is chosen depending on the output jitter and phase noise requirements and the noise contributors. As seen in section IV , any noise at the input of the PFD sees a low pass transfer function whereas noise injected anywhere after the VCO to the output point sees a high pass transfer function. Also from section III , it can be seen that period jitter at the output is governed by the high frequency components of the phase noise whereas the long term jitter is governed by the low frequency components of the phase noise. The bandwidth is decided based on what kind of jitter is important for the system and also whether the input clock comes from a clean reference like a crystal or from some noisy source like another PLL or from a RC oscillator. E.g. if the performance of interest is long term jitter and the PLL gets a clean reference clock from a crystal oscillator, the PLL bandwidth should be made as high as 0.1 times the reference clock. But if the reference clock is noisy, the best bandwidth has to be found. Similarly if period jitter is of interest, a very low bandwidth should be chosen as that reduces the amount of correction on the control voltage.

From the phase domain model, it can be seen that the PLL open loop transfer function has two poles at origin, one coming from the loop filter and the other one coming from the frequency to phase conversion in VCO. So the loop filter zero has to come before the open loop bandwidth of the loop. The relative location of the zero with respect to the open loop bandwidth can be fixed depending on the closed loop peaking requirement of the PLL. The 2nd pole of the loop filter is placed more than 3 times the open loop bandwidth of the system.

The VCO gain is fixed from the VCO range that needs to be supported and the control voltage range that can be used without degrading the power supply rejection and the phase noise performance of the VCO. The VCO gain has two contradictory requirements. A small VCO gain ensures that loop filter components can be at its minimum value which ensures a small area. Also the noise of the loop filter has lesser gain when it comes at the output. But this has the disadvantage of supporting a smaller VCO frequency range. Normally the VCO minimum operating frequency to maximum operating frequency ratio is kept as 1:2. The range problem can be taken care of by adding a separate control loop to lock the VCO very close to its operating frequency before the main loop takes over. Then the actual loop has to take care of smaller range mainly due to temperature variation of the oscillator frequency.

The charge-pump current can be put as minimum as possible keeping the linearity good. Normally the charge-pump structures suffer from problems like static phase offset and gain nonlinearity. Gain nonlinearity originates from clock feed-through and charge-injection errors in the charge-pump switches and modifies the bandwidth depending on the phase error and can cause more jitter at the output. Careful charge-pump architecture selection, careful switch sizing, clock feed-through cancellation and loop filter capacitor selection reduces this effect and the gain can be well controlled to within +/- 5%. Static phase offset is caused due to current mismatch in the charging and discharging paths in a charge-pump. This causes a reference spur at the output as the control voltage is modulated every reference cycle.

PFD architecture should be chosen such that there is no dead zone in the PFD. Presence of dead zone makes the PLL system open loop once the phase error enters this range. This also increases the jitter at the output of the PLL.

Once the loop parameters are fixed, the next step is to derive the noise specifications for each block. The phase domain model can be used to find the phase noise requirement to meet the performance. The reference spur specification decides the static phase offset that can be tolerated. The power supply rejection ratio of each block can be found depending on the noise amplitude that needs to be tolerated.

Three examples are discussed in the following subsections using the design methodology discussed above.

A. *WLAN baseband Analog frond end applications*

Table III shows a typical specification for a WLAN base-band analog front end PLL application. The PLL supplies the sampling clock for the receiver ADC, the data latching clock for the transmitter DAC and other clocks for running the digital logic of the SOC.

The input reference clock to the PLL is typically from a crystal oscillator hence the PLL has to support the typical crystal frequencies as shown in the table. The VCO frequency is chosen to provide flexibility.

TABLE III. WLAN BASEBAND PLL REQUIREMENTS

Parameter	Value	Units
Reference frequency	13,19.2,20,26,39.2,40	MHz
ADC sampling clock	80	MHz
Long term jitter (rms)	10	pS
Lock time	40	uS

The accumulated jitter requirement is to ensure that the degradation in RXADC (or TXDAC) SNR due to sampling clock jitter is significantly lower than the SNR specification which is typically 52 dB for a maximum input bandwidth of 10 MHz. Applying (24), for sampling clock jitter of 12 ps rms it can be seen that SNR is limited only to 62 dB due to clock jitter. The period jitter requirement on SOC clocks is to ensure sufficient timing margins for the digital blocks.

Since the input clock comes from the relatively jitter free crystal oscillator, it is the VCO noise (both due to phase noise as well as due to power supply) that dominates. So it is desired to keep the PLL bandwidth as high as possible but since the minimum input clock frequency is 13 MHz, the PLL's bandwidth has to be kept below 1 MHz to ensure that the continuous time approximation is still valid. The bandwidth of the PLL is typically kept the same for all input clock frequencies by proportionally changing the charge pump current.

The PLL locking behavior can be split into two portions; frequency locking and phase locking. During frequency locking, the charge pump current continuously charges the loop filter capacitor (slewing) and brings the VCO frequency very close to N*Input clock frequency, where N is the feedback division number. During phase locking, the settling time is decided by the PLL loop bandwidth. So to ensure lock time specification is met apart from having maximum possible bandwidth we also need to have sufficiently high charge pump current to ensure fast frequency settling.

This PLL was fabricated in 65 nm process node. Fig. 13 shows the measurement results which show very close match with the simulation results. The measurements were done with PLL input of 20 MHz, VCO running at 1920 MHz and output division of 196 to get a 20 MHz output. The measured long term jitter is 8.79 ps (rms) and the period jitter rms at 20MHz output is 5.89 ps.

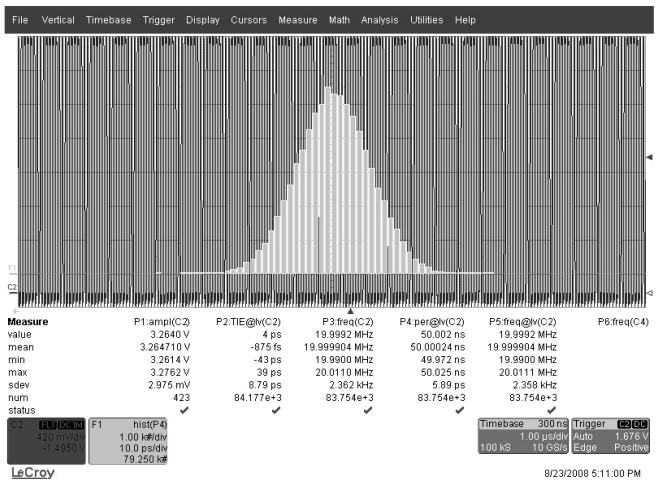


Fig. 13 Measured jitter histogram

B. Receiver baseband clock for DVB-H application with frequency tracking

Table IV shows a typical specification for a DVB-H base-band analog front end PLL application. The PLL supplies the sampling clock for the receiver ADC and the digital core.

TABLE IV. DVB-H BASEBAND PLL REQUIREMENT

Parameter	Condition	Value	Units
Reference frequency		10-40	MHz
Output clock		135-225	MHz
Frequency step at output	For full input range	0.5	Hz
Long term jitter (rms)	Integrated from 1 KHz – 5 MHz	30	pS
Phase noise at Output clock/6	1.45 MHz offset	-130	dBc/ Hz
	10 MHz offset	-145	

The frequency step of 0.5 Hz coupled with the low long term jitter and out of band phase noise requirements mandates the use of sigma delta fractional-N architecture. The requirement of 0.5 Hz frequency step at the output is derived from the requirement that the receiver PLL should be able to track the frequency drift of the transmitter clock with respect to the receiver clock. Also the steps have to be linear throughout the full fractional range of -1 to +1 for the frequency correction loop to have a controlled gain.

The long term jitter specification is derived from the ADC SNR requirements. Using (24), for a 4 MHz maximum signal frequency, the SNR due to clock jitter is going to be 62 dB which is 10 dB below the typical ADC requirement of 52 dB. The phase noise specification comes from out of band attenuation requirement of adjacent channels. In a DVB-H application, the N+1 adjacent carrier is at 1.45 MHz. The maximum power difference between the interferer and the wanted signal is 38 dB. The required SNR needs to be 25 dB and the noise bandwidth is 8 MHz. The single side band (SSB) phase noise required at 1.45 MHz offset can be calculated as [15]

$$\text{Phase noise} = -38 - 25 - 10\log_{10}(BW) \quad (28)$$

Using (28) SSB phase noise required at 1.45 MHz offset is -132 dBC/Hz. The N+2 adjacent carrier is at 9.45 MHz. The maximum power difference between the interferer and the wanted signal is 45 dB. The required SNR needs to be 25 dB. The required SSB phase noise can be similarly calculated to be -142 dBC/Hz at 9.45 MHz offset.[*]

The PLL parameter selection in this case is governed by both the jitter and phase noise requirement. A very high bandwidth would reduce the accumulated jitter and phase noise contribution of the VCO but would increase the phase noise contribution due to sigma-delta (SD) noise at higher frequency and it would be difficult to meet the phase noise mask beyond 1 MHz. A low bandwidth would attenuate the SD noise but would increase the VCO contribution and would require lot more power to be burnt in the VCO to meet the specifications. An optimization needs to be done to choose the SD noise transfer function, bandwidth and VCO phase noise to get the best power performance for meeting the specifications.

This PLL was fabricated in 90 nm process node. A 3rd order SD modulator with a 3rd order loop filter was used. The bandwidth is kept at the maximum possible value allowed by the sigma delta noise transfer function. Fig.14 shows the fractional linearity data. Fig. 15 shows the measured jitter for a 13 MHz input across the whole output range.

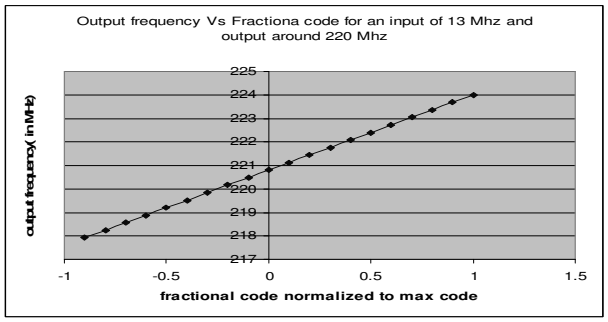


Fig. 14 Output frequency linearity with fractional code

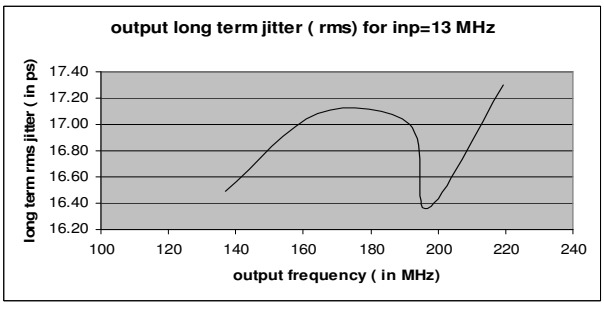


Fig. 15 Measured jitter vs output frequency

C. *High frequency clock for synchronous serial link transmiters e.g. MIPI/HDMI*

Table V shows a typical specification for a PLL for MIPI transmitter application in an image sensor chip. The PLL supplies the high frequency clock for the MIPI transmitter.

TABLE V. MIPI PLL REQUIREMENT

Parameter	Condition	Value	Units
Reference frequency		6 - 64	MHz
Output clock		200 - 800	MHz
Peak-to-peak period jitter		0.1	UI
	Output clock=800 MHz	125	pS
Tolerable supply noise(p-p)	3.3V supply	50	mV
	1.5V supply	30	mV

The input clock in this case comes from a noisy source. The PLL is required to filter the input jitter other than meeting the specifications mentioned in the table. This dictates a bandwidth low enough to filter the input clock. A prior knowledge of input jitter spectrum helps in choosing the bandwidth. Also among the specifications, achieving a low period jitter of 0.1UI at 800 MHz in the presence of the supply noise becomes a complex design issue. As discussed earlier, a lower bandwidth is normally preferred in case period jitter spec is the main specification as that reduces any large excursion of control voltage to the VCO.

In this kind of PLL the 3.3V supply will be shared with I/O buffers and 1.5V supply will be shared with digital core. This makes the supply to the PLL very noisy and the PLL performance needs to be guaranteed in the presence of this noise. Different blocks needs to have a power supply rejection (PSRR) specification depending on the supply noise magnitude. In the present case, the VCO has been budgeted to contribute 10 ps p-p period jitter at 800 MHz output for a supply tone of 50 mV at 3.3 V supply. The VCO consists of a Voltage-to-current converter (V2I) and a current controlled oscillator (ICO). The ICO gain is 1MHz/uA. The ICO is pretty linear and it needs 800uA for generating an 800 MHz output. For a VCO, the noise at the supply would modulate the current into the ICO and that will modulate the output frequency. For a peak frequency modulation of $f_{n\,at}$ a frequency f_{nom} the peak-to-peak period jitter is given by $2 * \frac{f_n}{f_{nom}^2}$ (29)

Using (29) and using the ICO gain of 1 MHz/uA, for a peak-to-peak period jitter of 10 ps , the peak-to-peak current variation into ICO needs to be less than 6.4 uA. The V2I has to be designed such that for the worst case noise tone (V2I rejection would have a zero and the PSRR profile would peak at some frequency) of 50 mV p-p the current change into the ICO has to be less than 6.4 uA p-p.

Most dominant contributor to period jitter is the noise on the 1.5V supply. This noise changes the digital path delay and contributes most of the period jitter. Normally the process would have a worst case number for percentage delay variation per 100 mV of noise. That would dictate the maximum noise that can be tolerated depending on the path length. This noise can also get coupled to the oscillator and create lot more jitter. Care has been taken so that this noise couples least to the oscillator and the digital path delay has been made minimum to reduce the effect of this noise.

VII. CONCLUSION

Various performance parameters for a PLL have been presented. PLL specifications have been derived for analog sampling applications and have been qualitatively discussed for other applications. Different noise sources and their effect at the output have been presented for charge-pump type PLLs. Finally, three design examples have been discussed.

ACKNOWLEDGMENT

The authors like to thank Prasunkali Bhattacharyya and Sriram G for their valuable contribution towards the concepts and results presented in this paper. Authors would also like to thank the Cosmic Circuits testing team for providing with the measurement data reported in this paper.

REFERENCES

[1] J.Alvarez, H. Sanchez, G.Gerosa and R. Countryman, A Wide Bandwidth Low-Voltage PLL for PowerPC™ Microprocessors," IEEE J. Solid-State Circuits,vol. 30, no. 4, pp. 383-391, April 1995.

[2] L.Devito, J.Newton, R.Croughwell, J. Bulzacchelli and F.Benkley, "A 52 MHz and 155MHz Clock-Recovery PLL," ISSCC Dig. Tech. Papers, pp 142-143, Feb. 1991.

[3] F.M.Gardner, Phaselock Techniques, 2nd edition, John Wiley & Sons, New York, 1979.

[4] R. E. Best, Phase-Locked Loops: Theory,Design and Applications, 2nd edition, McGraw-Hill Inc, 1993

[5] J. A. Crawford, Frequency Synthesizer Design Handbook , Artech House, 1994.

[6] W. F. Egan, Frequency Synthesis by Phase Lock, John Wiley & Sons, 1981.

[7] F. M. Gardner, "Charge-Pump Phase Lock Loops," IEEE Trans. Commun., vol COM-28, no. 11, pp. 1849-1858, Nov. 1980

[8] R. Tonietto, E. Zuffetti, R.Castello, I. Bietti, " A 3 MHz Bandwidth Low Noise RF All Digital PLL with 12 ps Resolution Time to Digital Converter," ESSCIRC 2006, pp. 150-153.

[9] F. herzel, B. Razavi, "A Study Of Oscillator Jitter Due To Supply And Substrate Noise," IEEE CASII:Analog aand Digial Signal Processing, vol.. 46, no. 1, pp. 56 -62, Jan. 1999.

[10] J. Craninckx, M. Steyaert, Wireless CMOS Frequency Synthesizer Design, Kluwer Academic Publishers, 2001

[11] B.Drakhlis, "Calculate oscillator jitter by using phase-noise analysis," Microwaves and RF, Jan/Feb. 2001.

[12] W. Rhee, "Design Of High-Performance CMOS Charge Pumps In Phase-Locked Loops," in Proc. Of ISCAS '99, vol. 2, pp. 545-548.

[13] A. Hajimiri, "Noise in phase-locked loops," in Southwest symposium on Mixed-Signal Design, 2001, pp. 1-6.

[14] Un-Ku Moon, K. Mayaram and J.T.Stonick, "Spectral analysis of time domain phase jitter measurements," IEEE Trans. Circ. Sysyt-II,Vol. 49, No. 5, pp. 321-327, May. 2002.

[15] P. Antoine et.al., " A Direct-Conversion Receiver for DVB-H," IEEE J. Solid-State Circuits, vol. 40, no. 12, pp. 2536 – 2546, Dec.2005

Invited Paper

Design for Variations

2009 22nd International Conference on VLSI Design

Coping With Variations Through System-level Design

Nilanjan Banerjee[†], Saumya Chandra[‡], Swaroop Ghosh[†], Sujit Dey[‡], Anand Raghunathan[†] and Kaushik Roy[†]
[†]School of Electrical and Computer Engineering, Purdue University
[‡]Department of Electrical and Computer Engineering, University of California at San Diego

Abstract— *Manufacturing and operation-induced variations have emerged as a critical challenge in designing integrated circuits (ICs) under the nanometer technology regime. Most work on addressing variations has focused on device, circuit, and logic-level solutions. As the magnitude of parameter variations increases with technology scaling, these techniques are not sufficient to address the negative impact that variations have on IC performance, power, yield, and design time. Therefore, in recent years, the research community has shown great interest in techniques to address variations starting from the other end of the design process, i.e., at the system level. In this paper, we provide an overview of various techniques that we have developed for coping with variations through system-level design. The presented techniques include a paradigm for designing variation-tolerant systems through critical path isolation for timing adaptiveness, application-specific techniques to achieve variation-tolerance by trading off quality of the result, variation-aware system-level power analysis, and system-level power management under variations. These techniques demonstrate that addressing variations during system-level design can greatly mitigate the effects of variations, enabling the design of integrated circuits in scaled technologies.*

I. INTRODUCTION

Variations in the characteristics of integrated circuits have always been an inevitable result of the fabrication process used to manufacture them, and the side-effects of their operation. In technologies with larger feature sizes (> 90nm), variations were sufficiently small that their effect could be incorporated using simple approaches such as the use of design margins or guard banding. With the scaling of device feature sizes, however, the incessant increase in the magnitude of variations implies that these approaches lead to excessively conservative designs [1][2][3]. Consequently, designers are faced with a choice between significant overheads due to over-design, increased design turn around times, an inability to meet performance or power targets, or parametric yield loss.

The sub-90nm era has witnessed a surge in research efforts directed at modeling and addressing the effects of variations. These techniques are largely at the mask, circuit, and logic levels. At the mask level, techniques such as optical proximity correction (OPC) can compensate for variations that arise due to lithographic effects. At the circuit level, techniques such as adaptive body biasing and voltage scaling [4], clock tuning [5], dynamic error correction [6], and variation-tolerant logic families [7] have been developed. At the logic level, statistical timing and power analysis [8][9][10] has been widely investigated and several commercial EDA tools now support this capability. Statistical timing analysis has been used to drive logic synthesis optimizations such as gate sizing [11][12].

From the viewpoint of simplicity in design methodologies, it would be ideal if techniques at the circuit and logic levels could solve the problems introduced by variations. However, there is significant concern in the semiconductor industry that these techniques are not sufficient to contain the increasing impact of variations as circuits scale into the deep nanometer regime [1][2][3]. Consequently, there has been a lot of interest in recent years to address variations earlier in the design cycle, namely at the architecture and system levels, where it is possible to effectively trade-off variation-tolerance, power, performance, and other metrics such as "quality of results". System-level design for variations can take advantage of information that is not easily available at the lower levels of abstraction, provide designers feedback about the impact of variations early on in the design cycle, and facilitate better design decisions at the system-level.

In this paper, we provide an overview of our recent work in the area of system-level design for variations. In Section II, we provide a brief background on the sources and impact of variations. Section III describes a paradigm for the design of variation-tolerant digital systems that is based on isolation of paths that may become critical under variations, and adaptively stretching them over multiple clock cycles. Section IV discusses how an understanding of the application can lead to variation-tolerant designs where performance (or power) is maintained by trading off the quality of the result. Section V presents a methodology for system-level power analysis considering the impact of variations. Sections VI shows how widely used power management techniques such as shutdown and dynamic voltage scaling can be adapted in the presence of variations. Collectively, the techniques described clearly establish the value of coping with variations through system-level solutions.

While this paper presents an overview of the authors' work on system-level design for variations, it bears mentioning that several other researchers have significantly contributed to this area [13]-[18]. We encourage readers interested in a broader overview of the field to read these and other related publications.

II SOURCES AND IMPACT OF VARIATIONS

Variations can be classified based on whether they are caused by the manufacturing process (manufacturing-induced variations) or due to the circuit's operation (operation-induced variations). Variations can also be classified as spatial or temporal. Spatial variations in the circuit characteristics occur between different parts of a die, across dies, or across wafers, as they manifest at "t=0", *i.e.*, at the beginning of the circuit's operational lifetime. Temporal variations, on the other hand, refer to variations in circuit characteristics over time.

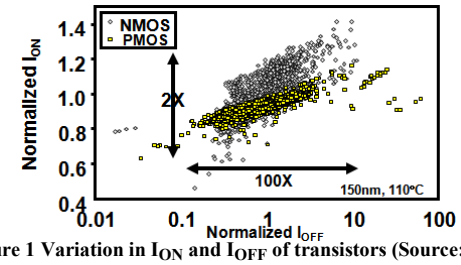


Figure 1 Variation in I_{ON} and I_{OFF} of transistors (Source: Intel)

Manufacturing-induced variations are artifacts of the lithography-based IC manufacturing process that has been in use for decades. These effects are exacerbated by the scaling of device feature sizes to a point where they are smaller than the wavelength of light used in the lithography process. Imperfect manufacturing processes result in fluctuations in transistor length (L), width (W), and oxide thickness (T_{OX}), as well as dopant fluctuations and line edge irregularities. For example, let us consider random dopant fluctuations. The number of dopant atoms per transistor has been steadily decreasing with feature size, from thousands to hundreds, and will reach tens in sub-45nm technologies. The manufacturing process cannot precisely control the number of dopant atoms in the channel of each transistor. The resulting statistical fluctuation becomes more significant as the nominal number of dopant atoms per transistor decreases, translating into increased variation in the threshold voltages of transistors. Another source of variations is line

1063-9667/09 $25.00 © 2009 IEEE
DOI 10.1109/VLSI.Design.2009.96

edge roughness (LER), which results in irregular channel edges, changing transistor strengths and causing mismatches between identical transistors within a chip.

Figure 1 illustrates the impact of manufacturing-induced variations on the off current (leakage) and on current (drive strength) of transistors. The data demonstrates that the spread in leakage is two orders of magnitude, while the spread in on current is a factor of two. The large impact of variations on leakage current is compounded by the increasing contribution of leakage to total power dissipation [19].

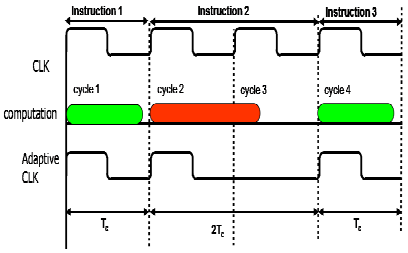


Figure 2(a) Timing diagram illustrating CRISTA

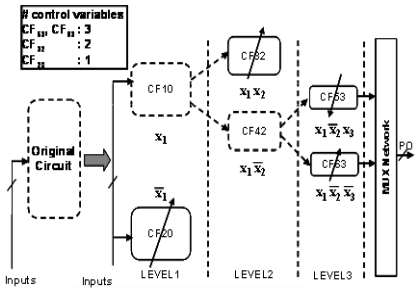


Figure 2(b) Shannon expansion based CRISTA design for random logic

The operation of ICs puts stress on the devices that constitute them, leading to operation-induced variations. For example, operation of an IC leads to a variation in the temperatures at different parts of the IC due to fluctuations in the workload or applications executed, impacting the devices' operating characteristics. Similarly, time-dependent dielectric breakdown (TDDB) can lead to variations in leakage current or threshold voltage due to degradation and formation of conductive paths in the dielectric material. Negative Bias Temperature Instability (NBTI) is a phenomenon that impacts PMOS transistors, increasing their threshold voltage, reducing channel mobility or inducing parasitic capacitances, thereby degrading performance. Electromigration is the transport of material caused by the gradual movement of the ions in a conductor due to the momentum transfer between conducting electrons and diffusing metal atoms. The resulting thinning of the metal lines over time increases the resistance of the wires and ultimately leads to failures.

From the perspective of a system designer, variations manifest as deviations from the expected behavior of components within an IC, or across different instances of the IC. In the following sections, we describe techniques to address variations during system level design.

III CRITICAL PATH ISOLATION FOR TIMING ADAPTIVENESS

CRISTA [20] is a new paradigm for low-power, variation-tolerant digital system design, which allows for aggressive voltage over-scaling. The CRISTA design principle (a) isolates and predicts the paths that may become critical under process variations, (b) ensures that they are activated rarely, and (c) avoids possible delay failures in the critical paths by adaptively stretching the clock period to two cycles. This allows the circuit to operate at reduced supply voltage while achieving the required yield with small throughput penalty (due to rare two-cycle operations). The concept of CRISTA is illustrated in Figure 2(a) through an example of three pipelined instructions, where *Instruction2* activates the critical path. The possible delay failure during *Instruction2* is avoided by extending the clock period to two cycles. This is achieved by gating the third clock pulse during the execution of *Instruction2*. The regular clock and gated CRISTA clock waveforms are shown in Figure 2(a). The critical paths are isolated in the logic through hierarchical Shannon expansion and gate sizing (Figure 2(b)). Silicon measurement of a two-stage pipelined ALU designed using CRISTA shows ~40% power savings with only 9% area overhead and very small throughput penalty [21]. Note that CRISTA can be applied at the circuit level as well as the architecture level. In [22], the authors describe a variant of CRISTA called Trifecta, which is an architectural technique that completes common-case sub-critical path operations in a single cycle but uses two cycles when the critical path is exercised. Trifecta improves the parametric yield while preserving performance and power. This technique was applied to the critical pipeline stages of a superscalar out-of-order and a single issue in-order processor, namely instruction issue and execute, respectively. Experiments show that the rare 2-cycle operations result in a small decrease in Instructions per Cycle (5% for integer and 2% for floating point benchmarks of SPEC2000). However, the increased delay slack causes an improvement in yield-adjusted throughput by 20% (12.7%) for an in-order (out-of-order) processor configuration.

IV. VARIATION-TOLERANCE BY TRADING OFF QUALITY-OF-RESULTS

Application Specific Integrated Circuits that implement DSP and media processing algorithms provide an extra knob to tolerate process variations namely, quality of the result. For example, variation induced timing errors in an image processing system can be tolerated by exploiting the fact that not all computations are equally important to obtain "good" image quality. It is possible to identify computational paths that are vital in maintaining high quality and design the architecture such that more important computations (in terms of quality) have shorter paths than less important ones. Such an architecture can enable us to tolerate variation-induced errors with minimal Peak Signal to Noise Ratio (PSNR) degradation of the image. Next, we consider various DSP applications (namely, DCT, color interpolation, and FIR filters) and demonstrate designs that enable trade-offs between power, quality, and process tolerance.

1. DCT: It has been noted [23] that all coefficients of the 2-D

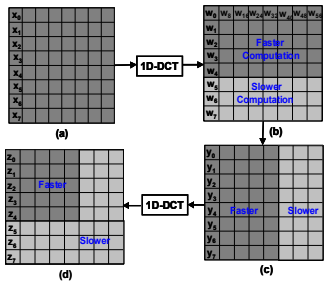


Figure 3(a) Voltage scalable, variation tolerant DCT architecture

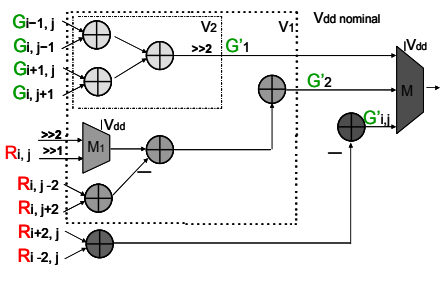


Figure 3(b)Voltage scalable, variation tolerant color interpolation architecture

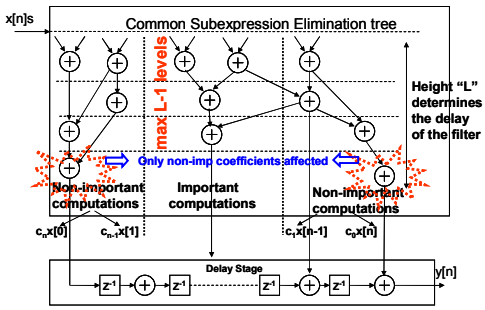


Figure 3(c) Voltage scalable, variation tolerant FIR filter architecture

DCT matrix do not affect the image quality in a similar manner. Analysis of various images shows that most of the input image energy (85% or more) is contained in the first 20 entries of the DCT matrix after the 2D-DCT operation. With this in mind, [23] proposes an architecture that computes the high-energy components of the final DCT matrix faster than the low-energy components (Figure 3(a)). Designing the architecture in such a manner a) isolates the computational paths based on high energy and low energy contributing components, and b) allows voltage overscaling to trade-off power dissipation and image quality under variations. Results show that under process variations and supply voltage scaling (1V down to 0.8V), there is an acceptable degradation in image quality (33dB down to 23dB) with considerable power savings (41% to 62%) compared to traditional implementations which fail under such conditions.

2. Color Interpolation: In order to develop a variation-tolerant color interpolation architecture, the computations required for interpolation are divided into two parts based on their impact on the output image quality [24]. The first part consists of the "bilinear" component (absolutely necessary for interpolation), and the second part consists of a "correction" term (used to improve PSNR). Interpolation of a missing color value at a pixel is achieved through linear combination of the bilinear and correction terms. The correction term is defined as a 2-D vector gradient of an image intensity function, with the components given by the derivatives in the horizontal and vertical direction at each image pixel. The basic idea behind the variation-tolerant architecture is to retain the bilinear component in all scenarios, and vary the size and complexity of the gradient component to achieve low power and variation-tolerance (Figure 3(b)). To maintain reasonable image quality under variations, [24] reduces computation complexity by varying the number and values of coefficients used in evaluation of the gradient component. Simulation results show that even at a scaled voltage (53% of nominal) and under severe process variations, the design provides reasonable image PSNR and 72% power savings [24].

3. FIR filter: The concept of segregating the more and less important computations for variation tolerance has been applied to FIR filters by developing a Level Constrained Common Subexpression Elimination (LCCSE) algorithm [25]. LCCSE can constrain the number of adder levels required to compute each of the coefficient outputs. By specifying a tighter constraint (in terms of adder levels) on the important coefficients, LCCSE ensures that the later computational steps process only the less important coefficients. In case of errors due to voltage scaling or process variations, only the less important outputs are affected, resulting in graceful degradation of filter quality and 25-30% power savings.

V. VARIATION-AWARE SYSTEM-LEVEL POWER ANALYSIS

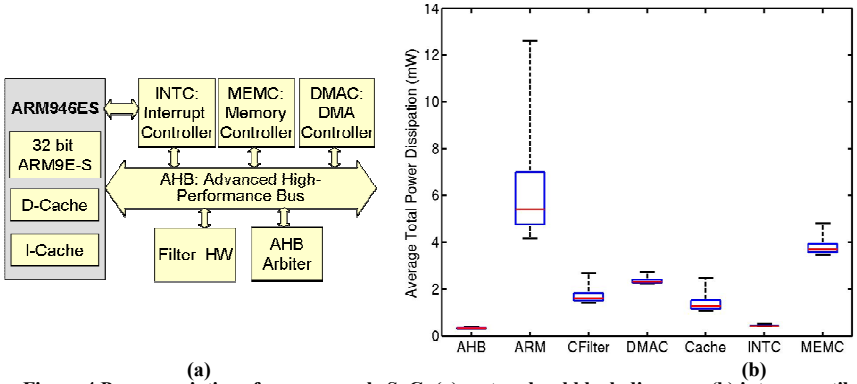
In this section, we show how to incorporate manufacturing and operation-induced variations into system-level power analysis. We consider both inter-die and intra-die variations in parameters such as effective channel length (L_{eff}) and threshold voltage (V_{th}). The power analysis methodology [35] effectively combines fast trace analysis, power-state based leakage modeling, efficient thermal analysis, and Monte Carlo sampling to generate System-on-Chip (SoC) power distributions and power variability over time.

V.1. Impact of Variations on SoC Power

We illustrate the issues involved in variation-aware system-level power analysis by analyzing the impact of variations on power for an example SoC shown in Figure 4(a). The SoC contains an ARM946 processor [26] that executes an image processing application and dedicated hardware (*Filter_HW*) that accelerates image filtering operations. Other SoC components are an on-chip bus (*AHB*) [27], a memory controller, a DMA controller and an interrupt controller. The SoC is implemented using a commercial, 90 nm standard cell library [28] and is operated at a frequency of 206 MHz and a voltage of 1 V.

Example 1: In this example, we compute the power consumption of the SoC for a given workload, using a simulation-based system-level power analysis tool [29]. We assume inter-die variations in L_{eff} with $3\sigma = 0.3\mu$ and use conventional Monte-Carlo analysis techniques to compute the power distribution. A box-and-whisker plot of the power distribution for each component is shown in Figure 4(b). The lower and upper extremities of each box represent the 25th and 75th percentiles of the power distribution. We notice that the inter-quartile range (the box height) for the ARM processor core is 25% of its average power, while it is only 8% for the AHB, suggesting that variations impact different components differently. This can be explained as follows. While leakage power is highly sensitive to channel length variations, dynamic power is relatively immune. Hence, the extent to which variations impact power consumption of a component depends upon the contribution of the leakage power to total power. The contribution of leakage in turn depends on how much time the component spends in different power-states (active, sleep, deep-sleep, *etc*.) while processing the given workload. In this study, the ARM core spends a lot of time in the sleep power-state, in which its power consumption is almost entirely due to leakage. On the other hand, the AHB is mostly active, serving requests from either the ARM core or other masters. This example shows that *the impact of variations on power consumption in SoC components depends on their workload characteristics*. ∎

Example 2: In this example, we illustrate the impact of taking thermal variations and spatially correlated intra-die variations into account. We integrated our power analysis tool with floor-planning [30] and thermal modeling [31] tools to accurately capture the inter-dependences between die temperature and total power consumption. We obtain the average power distributions of the SoC under three scenarios: (a) considering inter-die variations only (D0), (b) considering inter-die variations and thermal variations (D1), and (c) considering inter-die and spatially correlated intra-die variations and thermal variations (D2). We assume that the effective standard deviation ($\sqrt{(\sigma^2_{inter} + \sigma^2_{intra})}$) in L_{eff} in case (c) is same as the standard deviation σ_{inter} used in cases (a) and (b). We compare the power distributions for all three scenarios in Figure 5. Note that distribution D1 has a high variance due to a long tail, which has been truncated in Figure 5 for clear illustration. First, we note that *including thermal effects increases the mean and standard deviation of the power distribution significantly*. Secondly, we note that *ignoring intra-die spatial variations leads to a highly pessimistic estimate of the spread*



(a) (b)

Figure 4 Power variations for an example SoC: (a) system-level block diagram; (b) inter-quartile range of power variations for various system components

of the power distribution. Including thermal effects and intra-die spatial correlations changes the mean and standard deviation by 32.5% and 55%, respectively (D0 *vs.* D2 in Figure 5). ∎

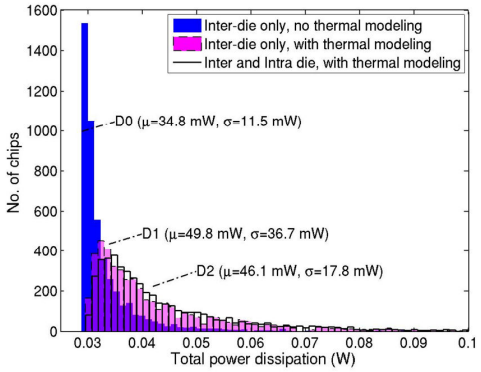


Figure 5 Impact of temperature variations and spatially correlated intra-die variations on the average power distribution

The above examples demonstrate the importance of properly incorporating the inter-dependencies between manufacturing-induced variations, component workload characteristics, leakage power, and temperature variations, during system-level power analysis. Simple spreadsheet analysis based on the distribution of process parameters and typical operating temperatures fails to accurately capture these inter-dependencies and can lead to highly inaccurate power distributions [35]. On the other hand, Monte Carlo techniques that involve iterative system simulation with integrated thermal analysis are too slow for use in architecture exploration [35].

V.2. Variation-aware Power Analysis Methodology

We have developed a methodology to accurately and efficiently estimate the total power distribution under variations [35]. The key attribute of our methodology is that speedup is achieved by performing both system simulation based power analysis and thermal analysis *outside the Monte Carlo loop without compromising significantly on accuracy*. To achieve this, we use a three-phase approach as shown in Figure 6.

- In *Phase 1*, system simulation is performed just once to capture necessary information in the form of component-level dynamic power and power-state traces.
- In *Phase 2*, a small number ($\mathbf{N_c}$) of chip instances are generated and a corresponding database of temperature traces is obtained as follows. For each instance, the power-state traces are used to compute component-level leakage traces using leakage models described in [35]. These leakage traces are combined with dynamic power traces to perform thermal analysis. Leakage and thermal analysis are performed iteratively until convergence. This compute-intensive iterative analysis is limited to a small number ($\mathbf{N_c}$) of chip instances.
- In *Phase 3*, Monte-Carlo analysis is performed wherein a large number of random chip samples are generated. For each sample, we find the chip instance analyzed in Phase 2 that is "closest" to it, and its leakage power is obtained using the corresponding pre-computed temperature trace in the database. The rationale behind this approximation is that temperature does not vary significantly for small variations in power and hence, this "quantization" does not impact accuracy of the overall power distribution significantly.

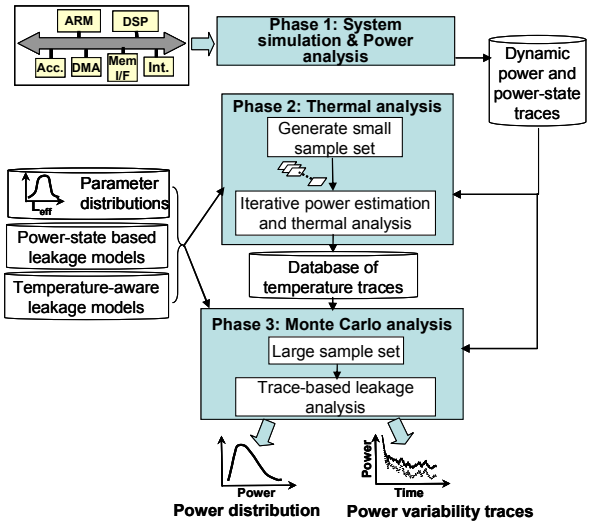


Figure 6. Variation-aware system-level power analysis

We estimated the power distribution for the image processing SoC shown in Figure 4 for a workload trace of *500ms* and $\mathbf{N_c}$=10. The power distribution obtained using the above methodology (μ =52.3 mW, σ = 35.3 mW) closely matches the one obtained using direct Monte Carlo simulations (μ = 52.7 mW, σ = 36.4 mW). Furthermore, mean error is less than 0.5% for each sample point. The error can be further reduced by increasing $\mathbf{N_c}$. In addition, the proposed methodology achieves speedups of 3-4 orders of magnitude over an optimized implementation of direct Monte Carlo simulation [35].

VI. VARIATION-AWARE POWER MANAGEMENT

Power management is one of the most widely used power reduction techniques at the system level. In [36], we addressed the problem of designing effective power management schemes in the presence of manufacturing-induced variations. We consider both shutdown-based and slowdown-based power management techniques. We demonstrate that conventional power management schemes, designed using nominal power characteristics, can result in substantial power wastage, and propose techniques and that can lead to significant improvements in the energy distribution.

VI.1. Overview

We propose *design-specific* and *chip-specific* approaches to designing variation-aware power management schemes. In the design-specific approach, a set of values for parameters that control the power management scheme is obtained at design time, and fixed across all fabricated instances of the chip. The difference from conventional power management schemes is that these parameter values are computed with an objective to optimize the resulting power or energy distribution across all chip instances. In the chip-specific approach, parameters values are determined for each chip instance based on its individual characteristics. To realize this, each chip needs to sense its power characteristics after fabrication and configure the power management parameters accordingly.

Metrics of optimization: Under process variations, the effectiveness of a power management scheme must be evaluated in the context of an energy distribution rather than a deterministic value. The distribution mean may not always provide a meaningful measure. For example, in Figure 7(a), the distribution corresponding to the solid line has a lower mean compared to the one corresponding to the dashed line. However, the dashed distribution might be more desirable since it results in a larger number of chips meeting a certain energy constraint (*energy yield*). To better capture these effects, metrics that consider higher moments of the energy distribution, such as $\mu+\sigma$, may be used. Another example is the $\mathbf{N^{th}}$ percentile of the energy distribution. In Figure 7(b), we see that the dashed cumulative distribution (CDF) is more desirable since it has a lower value of the given percentile, and hence, corresponds to a larger number of chips with better energy characteristics.

VI.2. Shutdown-based Power Management

In shutdown-based dynamic power management (DPM), a power manageable component (PMC) is put into low power states during

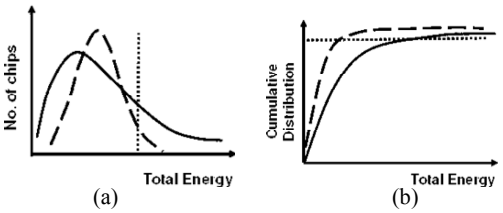


Figure 7. Metrics for power management policy evaluation

periods of inactivity. These transitions are governed by power management policies, such as timeout-based, history-based and stochastic policies [32]. We have developed variation-aware policies in the context of two power management policy frameworks, namely an ideal Oracle-based framework and a timeout-based framework.

A Power Manageable Component (PMC) can be in the *Active* state or in one of *n* low-power states *(S_1, S_2 ... S_n)*. The power dissipation in state S_j is denoted by P_{Sj}. It consists of the dynamic power dissipation $P_{D,Sj}$ and the leakage power dissipation $P_{L,Sj}$. Transition between *Active* and S_j is associated with a power overhead denoted by $P_{tr,Sj}$ and a delay overhead denoted by $T_{tr,Sj}$.

A shutdown-based policy can be specified using parameters that decide when to transition to a low-power state. In the Oracle-based framework, the length of the idle period, T_{idle}, is known a priori. Hence, the optimal low-power state can be determined as soon as an idle period is encountered. The policy parameter is the minimum idle period length, or threshold ($T_{th,Sj}$), above which it is beneficial to transition to S_j. The threshold time for a low-power state S_{j+1} can be computed using simple breakeven analysis as follows:

$$T_{th,S_{j+1}} = \frac{(P_{tr,S_{j=1}} - P_{S_{j+1}})T_{tr,S_{j+1}} - (P_{tr,S_j} - P_{s_j})T_{tr,S_j}}{P_{S_j} - P_{S_{j+1}}} \quad (1)$$

For a timeout-based framework, the system waits in each state for a pre-specified timeout ($T_{to,Sj}$) to expire before it transitions to the next low power state S_j. In this case, the parameters are the timeout values.

Impact of variations: Let us consider the ARM946 processor core presented in the previous section. When an idle period is encountered, it can either transition to a *Sleep* state or a *Deep-sleep* state. The core is assumed to be clock-gated in the *Sleep* state, and hence its power consumption is dominated by leakage power ($P_{L,S}$). In the *Deep-sleep* state it is voltage gated, and hence has negligible power consumption. Let us denote the threshold values computed without considering variations (using mean characteristics) as $T_{th,S}$ and $T_{th,DS}$. Now for a chip instance '*i*' with leakage power greater than the mean leakage power ($P^i_L > \mu[P_L]$), the optimal threshold time to *Deep-sleep* $T^i_{th,DS}$, computed using Equation 1 and its own power characteristics is smaller than $T_{th,DS}$. Intuitively, higher leakage makes it beneficial to pay the transition penalty and exploit the *Deep-sleep* state even for certain idle periods that are smaller than the conventionally chosen threshold $T_{th,DS}$. Therefore, for instance '*i*', if the threshold is set to $T_{th,DS}$, it will be forced to transition to *Sleep* for all idle periods $T_{idle} \in [T^i_{th,DS}, T_{th,DS})$, even though it is more beneficial to transition to *Deep-sleep*. Similarly, for an instance with leakage characteristics such that $P^i_L < \mu[P_L]$, the optimal threshold value is higher than that computed using the mean. Hence, conventional power management policies can perform quite sub-optimally for some chip instances. The extent of sub-optimality depends upon the extent to which the power characteristics are impacted by variations, and the idle period distribution in the workload. Next, we present brief descriptions of the variation-aware design-specific and chip-specific power management approaches.

VI.2.1 Design-specific approach

We wish to calculate the policy parameters ($T_{th,Sj}$ or $T_{to,Sj}$) such that a metric of the total energy distribution $Met[E_{TOT}]$ is optimized. For the Oracle-based framework, we have shown that if *Met* is linear in the energy dissipation in different power states, then the optimum parameter set can be obtained using a modified break-even analysis as follows:

$$T_{th,S_{j+1}} = \frac{(Met[P_{tr,S_{j=1}}] - Met[P_{S_{j+1}}])T_{tr,S_{j+1}} - (Met[P_{tr,S_j}] - Met[P_{s_j}])T_{tr,S_j}}{Met[P_{S_j}] - Met[P_{S_{j+1}}]} \quad (2)$$

Furthermore, it can be shown that the metrics (μ+σ) and the $\mathbf{N}^{th}$ percentile of the total energy distribution satisfy the linearity property [36]. For the timeout-based framework, closed-form analytical computation of the optimal T_{to} parameter is not possible. Hence, we have developed various search based methods [36] to compute the optimal timeout parameter values.

VI.2.2 Chip-specific approach

In the chip-specific approach, the power management policy parameters of each fabricated instance of the SoC are configured according to its specific leakage characteristics. Each chip needs to be calibrated in order to determine its leakage characteristics. Leakage calibration requires the use of on-chip measurement circuitry [33]. In [36], we have proposed three calibration methods, namely full calibration, approximate calibration and random calibration. These methods tradeoff calibration effort for the optimality of the power management policy parameters derived.

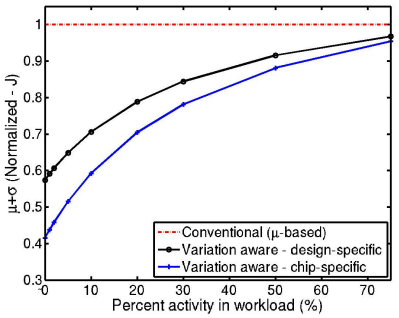


Figure 8. Benefits of variation-aware power management

We evaluated the benefits of variation-aware power management using a cycle-level power model of an ARM946 processor core, over a wide range of workloads. The proposed approaches result in significant improvements in metrics such as μ+σ and $\mathbf{N}^{th}$ percentile compared to conventional variation-agnostic policies. For example, improvements in μ+σ of up to 59% for the Oracle-based framework and up to 43% for the timeout-based framework are obtained.

VI.3. Slowdown-based Power Management

In slowdown-based power management or dynamic voltage scaling (DVS), slack in the workload is exploited to allow execution at the lowest possible frequency level such that the performance targets are met. This allows the supply voltage to be scaled down in order to reduce energy consumption. In conventional DVS schemes, a small number of discrete voltage-frequency pairs **(FL-VL)** are determined at design-time based on the IC's frequency-voltage (F-V) characteristics at nominal or worst-case process corners. We denote them as **FL** = $\{FL_1...FL_K\}$ and **VL** = $\{VL_1...VL_K\}$. Furthermore, the pairing of voltage and frequency levels is also fixed at design time, *i.e.*, operating frequency F_k is associated with voltage level V_k for all chip instances. We refer to this as a *fixed discrete DVS* scheme.

Impact of variations: In order to demonstrate the sub-optimality of conventional *fixed discrete DVS* under variations, we consider an ARM946 processor core implemented in 90nm technology [28] at an operating frequency of F_{req} =133.3 Mhz. Using nominal F-V characteristics, we computed that a minimum voltage level of 0.82V is required to meet this frequency target. However, in the presence of variations, each chip instance has different F-V characteristics (see Figure 9) and hence the required voltage levels are different. For example, *Instance1* requires supply voltage of 0.96V to operate at 133MHz, whereas *Instance2* can operate correctly at even 0.72V.

This indicates that setting the voltage to 0.82V for all chips is highly sub-optimal. For the workload under consideration, it can cause *Instance1* to exhibit timing errors (since the chip operates slower than F_{req}) and *Instance2* to dissipate more energy than required.

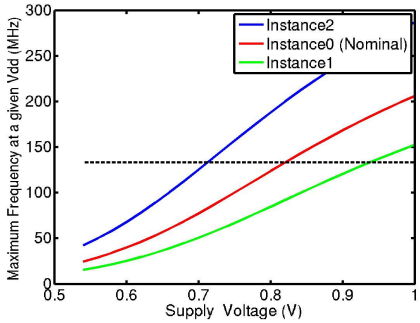


Figure 9 Impact of variations on slowdown based power management techniques

VI.3.1 Variation-aware DVS

We propose a variation-aware DVS scheme called *variable discrete DVS*, which is based on two key ideas:

- At design time, voltage levels are determined while taking variations into account. For a given frequency F_k, the required voltage V_{req} is a random variable whose distribution depends on the impact of variations. Therefore, determining the set of voltage levels based on this distribution, instead of nominal or worst case characteristics, will improve the overall energy distribution.
- At runtime, the voltage level for a given frequency level is selected based on the F-V characteristics of each chip instance. For a given F_k, each chip instance '*i*' has a different optimum voltage of operation, V^i_{req}, based on its F-V characteristics. However, since only a limited set of voltage levels are available, in the proposed scheme, the operating voltage for instance '*i*' is selected as $V^i_k = \min(\mathbf{VL} > V^i_{req})$. This involves decoupling the fixed pairing between frequency and voltage levels.

We applied the *variable discrete DVS* technique to an ARM946 processor core with 5 frequency levels. Our results indicate that (i) matching the voltage levels with frequency levels uniquely for each chip results in significant improvements over *fixed discrete DVS* (up to 118% in energy yield), and (ii) variation-aware selection of voltage levels yields an additional 30% improvement.

In summary, we have demonstrated that variation-aware power management can significantly improve the energy distribution of an IC in the presence of variations.

VII. CONCLUSIONS

Process variations have emerged as a critical challenge in the design of ICs in nanoscale technologies, and in the extreme case may prevent us from realizing the benefits of scaling. Effectively addressing this challenge requires that variations should be considered from the early stages of the design cycle. In this paper, we presented various approaches for coping with variations through system-level design, which complement lower-level techniques and show great promise in enabling cost-effective design of variation-tolerant ICs.

Acknowledgment: This work was supported in part by the Focus Center Research Program (GSRC) and the Semiconductor Research Corporation.

References

[1] International Technology Roadmap for Semiconductors, http://public.itrs.net/reports.html.
[2] S. Borkar *et al.*, "Parameter variations and impact on circuits and micro-architecture," *Design Automation Conf.,* pp. 338-342, June 2003.
[3] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10-16, Nov. 2005.
[4] J. Tschanz, S. Narendra, R. Nair, and V. De, "Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors," *IEEE J. Solid-State Circuits*, vol. 38, no. 5, pp. 826-829, May 2003.
[5] P. Mahoney, E. Fetzer, B. Doyle, and S. Naffziger, "Clock distribution on a multi-core multithreaded Itanium-family processor," *IEEE Int. Solid-State Circuits Conf.*, pp. 292-293, Feb. 2005.
[6] D. Ernst *et al*.,, "Razor: Circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, no. 6, pp. 10-20, March 2005.
[7] C.H. Kim, K. Roy, S. Hsu, R. Krishnamurthy, and S. Borkar, "A process variation compensating technique with an on-die leakage current sensor for nanometer scale dynamic circuits," *IEEE Trans. VLSI Systems*, vol. 14, no. 6, pp. 646-649, June 2006.
[8] C. Visweswariah *et al*., "First-order incremental block-based statistical timing analysis," *Design Automation Conf.*, pp. 331-336, June 2004.
[9] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single PERT-like traversal," *Proc. Int. Conf. Computer-Aided Design*, pp. 621-625, Nov. 2003.
[10] A. Agarwal, V. Zolotov, and D.T. Blaauw, "Statistical timing analysis using bounds and selective enumeration," *IEEE Trans. Computer-Aided Design*, vol. 22, no. 9, pp. 1243-1260, Sept. 2003.
[11] E. T. A. F. Jacobs and M.R.C.M. Berkelaar, "Gate sizing using a statistical delay model," *Design, Automation, and Test Europe*, pp. 283-291, March 2000.
[12] A. Agarwal, K. Chopra, D. Blaauw, and V. Zolotov, "Circuit optimization using statistical static timing analysis," *Design Automation Conf.,* pp. 321-324, June 2005.
[13] D. Marculescu and E. Talpes, "Energy awareness and uncertainty in microarchitecture-level design," *IEEE Micro*, vol. 25, no. 5, pp. 64–76, Sep. 2005.
[14] D. Marculescu and S. Garg, "System-level process-driven variability analysis for single and multiple voltage-frequency island systems," *Int. Conf. Computer-Aided Design*, pp. 541–546, Nov. 2006.
[15] N. Azizi, M. M. Khellah, V. De, and F. N. Najm, "Variations-aware low-power design with voltage scaling," *Design Automation Conf.*, pp. 529–534, June 2005.
[16] J. Donald and M. Martonosi, "Power efficiency for variation-tolerant multicore processors," *Int. Symp. Low-Power Electronics and Design*, pp. 304–309, Oct. 2006.
[17] D. Sylvester, D. Blaauw, E. Karl, "ElastIC: An adaptive self-healing architecture for unpredictable silicon," *IEEE Design & Test*, vol. 23, no.6, pp.484-490, Nov. 2006.
[18] N. R. Shanbhag, "Reliable and energy-efficient digital signal processing," *Design Automation Conf.*, pp. 830-835, June 2002.
[19] K. Roy, S. Mukhopdhayay, and H. Mahmoodi, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proc. IEEE*, vol. 91, no. 2, pp. 305-327, Feb. 2003.
[20] S. Ghosh *et al*., "CRISTA: A new paradigm for low-power and robust circuit synthesis under parameter variations using critical path isolation," *IEEE Trans. Computer-Aided Design*, vol. 26, no. 11, pp. 1947-1956, Nov. 2007.
[21] S. Ghosh. K. Kim, P. Batra, and K. Roy, "Process-tolerant *low-power* adaptive pipeline under scaled-Vdd," *Custom Integrated Circuits Conf.*, pp. 733-736, Sep. 2007.
[22] P. Ndai *et al*., "Trifecta: A non-speculative scheme to exploit common, data-dependent subcritical paths," *IEEE Trans. VLSI Systems* (to appear).
[23] N.Banerjee *et al*., "Process-variation tolerant low power DCT architecture," *Design, Automation, and Test Europe,* pp. 630-635, Apr. 2007.
[24] G. Karakonstantis, N. Banerjee, and K. Roy, "Design methodology to trade off power, output quality and error resiliency: Application to color interpolation filtering," *Int. Conf. Computer-Aided Design,* pp. 199-204, Nov. 2007.
[25] N. Banerjee, J. Choi, and K. Roy, "A process variation aware low power synthesis methodology for fixed-point FIR filters," *Int. Symp. Low Power Electronics and Design*, pp. 147-152, Aug. 2007.
[26] ARM946ES core, http://www.arm.com/products/CPUs/ARM946ES.html.
[27] "AMBA 2.0 Specification" http://www.arm.com/armtech/AMBA.
[28] "Cell Based IC CB-90 L/M/H Type Features/Basic Specification," NEC Electronics Corp. http://www.necel.com/cbic/en/cb90/cb90.html.
[29] N. Bansal, K. Lahiri, A. Raghunathan, and S. T. Chakradhar, "Power monitors: A framework for system-level power estimation using heterogeneous power models," *Int. Conf. VLSI Design*, pp.579-585, Jan. 2005.
[30] "UC Santa Cruz Floor-planning Tool." http://www.cse.ucsc.edu/research/surf/GSRC/progress.html.
[31] "HotSpot 3.0 Temperature Modeling Tool." http://lava.cs.virginia.edu/HotSpot.
[32] L. Benini, A. Bogliolo, and G. D. Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Trans. VLSI Systems*, vol. 8, no. 3, pp. 299–316, June 2000.
[33] C. Kim, K. Roy, S. Shu, K. R. Krishnamurthy, and S. Borkar, "On-die CMOS leakage current sensor for measuring process variation in sub-90nm generations," *VLSI Circuits Symp.*, pp. 250–251, June 2004.
[34] A. Gersho and R. M. Gray, "Vector Quantization and Signal Compression", Kluwer Academic Publishers, 1992.
[35] S. Chandra, K. Lahiri, A. Raghunathan, and S. Dey, "Considering process variations during system-level power analysis," *Int. Symp. Low Power Electronics and Design*, pp. 342-345, Oct. 2006.
[36] S. Chandra, K. Lahiri, A. Raghunathan, S. Dey, "Variation-tolerant dynamic power management at the system-level," *IEEE Trans. VLSI Systems* (to appear).

Author Index

Author Index

Author Index

Author Index

Author Index

IEEE Computer Society Conference Publications Operations Committee

CPOC Chair

Chita R. Das
Professor, Penn State University

Board Members

Mike Hinchey, *Director, Software Engineering Lab, NASA Goddard*
Paolo Montuschi, *Professor, Politecnico di Torino*
Jeffrey Voas, *Director, Systems Assurance Technologies, SAIC*
Suzanne A. Wagner, *Manager, Conference Business Operations*
Wenping Wang, *Associate Professor, University of Hong Kong*

IEEE Computer Society Executive Staff

Angela Burgess, *Executive Director*
Alicia Stickley, *Senior Manager, Publishing Services*
Thomas Baldwin, *Senior Manager, Meetings & Conferences*

IEEE Computer Society Publications

The world-renowned IEEE Computer Society publishes, promotes, and distributes a wide variety of authoritative computer science and engineering texts. These books are available from most retail outlets. Visit the CS Store at *http://www.computer.org/portal/site/store/index.jsp* for a list of products.

IEEE Computer Society *Conference Publishing Services* (CPS)

The IEEE Computer Society produces conference publications for more than 250 acclaimed international conferences each year in a variety of formats, including books, CD-ROMs, USB Drives, and on-line publications. For information about the IEEE Computer Society's *Conference Publishing Services* (CPS), please e-mail: cps@computer.org or telephone +1-714-821-8380. Fax +1-714-761-1784. Additional information about *Conference Publishing Services* (CPS) can be accessed from our web site at: *http://www.computer.org/cps*

IEEE Computer Society / Wiley Partnership

The IEEE Computer Society and Wiley partnership allows the CS Press *Authored Book* program to produce a number of exciting new titles in areas of computer science and engineering with a special focus on software engineering. IEEE Computer Society members continue to receive a 15% discount on these titles when purchased through Wiley or at: *http://wiley.com/ieeecs*. To submit questions about the program or send proposals, please e-mail jwilson@computer.org or telephone +1-714-816-2112. Additional information regarding the Computer Society's authored book program can also be accessed from our web site at:
http://www.computer.org/portal/pages/ieeecs/publications/books/about.html

Revised: 21 January 2008

CPS Online is our innovative online collaborative conference publishing system designed to speed the delivery of price quotations and provide conferences with real-time access to all of a project's publication materials during production, including the final papers. The ***CPS Online*** workspace gives a conference the opportunity to upload files through any Web browser, check status and scheduling on their project, make changes to the Table of Contents and Front Matter, approve editorial changes and proofs, and communicate with their CPS editor through discussion forums, chat tools, commenting tools and e-mail.

The following is the URL link to the ***CPS Online*** Publishing Inquiry Form:
http://www.ieeeconfpublishing.org/cpir/inquiry/cps_inquiry.html