ESE Back End 2.0

D. Gajski, S. Abdi
(with contributions from H. Cho, D. Shin, A. Gerstlauer)
Center for Embedded Computer Systems
University of California, Irvine

http://www.cecs.uci.edu
Technology advantages

- **No basic change in design methodology required**
  - ES methodology follows present manual design process
- **Productivity gain of more than 1000X demonstrated**
  - Designers do not write models
- **Simple change management: 1-day change**
  - No rework for new design decisions
- **High error-reduction: Automation + verification**
  - Error-prone tasks are automated
- **Simplified globally-distributed design**
  - Fast exchange of design decisions and easy impact estimates
- **Benefit through derivatives designs**
  - No need for complete redesign
- **Better market penetration through customization**
- **Shorter Time-to-Market through automation**
ES Environment

ESE Front – End
System Capture + Platform Development

Timed TLM

ESE Back – End
SW Development + HW Development

C + RTL

Application Tools: Compilers/Debuggers
Commercial Tools: FPGA, ASIC

Decision User Interface (DUI)
Create
Select
Partition
Map
Compile
Replace

Validation User Interface (VUI)
Compiler
Debugger
Stimulate
Verify
TIMED

CYCLE ACCURATE
Compile
Check
Simulate
Verify
Input: Transaction Level Model (TLM)

TLM is generated automatically from ESE Front-End for given Spec/Platform
TLM Features

• **Universal Bus Channel (UBC)**
  • Bus is modeled as universal channel with send/recv, read/write functions
  • Well defined functions for routing, synchronization, arbitration and transfer

• **SW modeling**
  • Application SW is modeled as processes in C
  • A RTOS model or real RTOS is used for dynamic scheduling of processes
  • Communication with peripherals, memory or other IP is done using UBC

• **HW modeling**
  • Application HW is modeled as processes written in C
  • Communication with processor, memory or other IP is done using UBC

• **Memory modeling**
  • Memory is modeled as array in C
  • Controller is modeled by function in UBC
Cycle-Accurate Software Synthesis

CPU

Program

EXE

RTOS

HAL

CPU Bus

Compile

RTOS Synthesis

Bridge

IP Bus

HAL

B1

B2

B5

OS

CPU

HW

IP
SW Synthesis Issues

• **Compiler selection**
  • The designer specifies which compiler is used for the SW

• **Library selection**
  • Libraries are selected for SW support such as file systems, string manipulation etc.
  • Prototype debugging requires selection of additional libraries

• **RTOS selection and targeting**
  • Designer selects an RTOS for the processor
  • RTOS model is replaced by real RTOS and SW is re-targeted

• **Program and data memory**
  • Address range for SW program memory is assigned
  • Address range for data memory used by program is assigned
  • For large programs or data, off-chip memory may be allocated
Cycle-Accurate Hardware Synthesis

CPU Bus

Mem

Bridge

IP Bus

CPU

HW (RTL)

Behavior in C

IP (RTL)

Cycle-accurate Synthesis

B6  B3
HW Synthesis Issues

- **IP insertion**
  - C model of HW is replaced with pre-designed RTL IP, if available

- **RTL synthesis tool selection**
  - RTL synthesis tool must be selected for custom HW design

- **SystemC code generation**
  - C/SystemC code for input to RTL synthesis tool is generated

- **Synthesis directives**
  - RTL architecture and clock cycle time is selected
  - UBC calls are treated as single cycle operations, to be later expanded during interface synthesis

- **HDL generation**
  - RTL synthesis result in cycle accurate synthesizable Verilog code
Cycle-Accurate Interface Synthesis

CPU

IC

Mem

Bridge

Custom HW

Arbiter

Interface Synthesis

IP
Interface Synthesis Issues

- **Synchronization**
  - UBC has unique flag for each pair of communicating processes
  - Flag access is implemented as polling, CPU interrupt or interrupt controller

- **Arbitration**
  - Selected from library or synthesized to RTL based on policy

- **Bridge**
  - Selected from library or synthesized using universal bridge generator

- **Addressing**
  - All communicating processes are assigned unique bus addresses

- **SW communication synthesis**
  - UBC functions are replaced by RTOS functions and assembly instructions

- **HW communication synthesis**
  - DMA controller in RTL is created for each custom HW component
  - Send/Recv operations are replaced by DMA transfer states
Pin/Cycle-Accurate Model

CPU

Program

EXE
RTOS
HAL

IC

Mem

Bridge

Arbiter

HW

IP

PCAM is downloaded automatically for fast prototyping with FPGAs
MP3 Player Prototyping with ESE Back-end

• **TLM Input**
  - TLM is generated by ESE front-end for MP3 application and platform

• **Interface synthesis**
  - Polling or interrupt mechanism is selected for synchronization
  - Arbiter is selected for buses with multiple masters
  - Bridge between CPU bus and peripheral bus is created by Bridge Generator

• **SW synthesis**
  - Compiler/RTOS for SW is selected and addresses are generated for memories

• **HW synthesis**
  - RTL is generated for custom HW cores by NISC compiler or C→RTL tools

• **Export to FPGA design tools**
  - Files are generated for creating complete project for FPGA tools

• **FPGA download and test**
  - FPGA design tools create bit-stream for programming the board
  - MP3 player prototype runs directly on FPGA board
MP3 Decoder Application

- **Functional block diagram (major blocks only)**

  ![Diagram](image)

- **Application features**
  - 12K lines of C code
  - IMDCT and DCT are compute intensive
    - Candidates for HW implementation
  - Left channel and right channel are data independent
    - Concurrent execution possible
MP3 Player TLM (SW+4HW)

- MP3 encoder mapped to SW (MicroBlaze), DCT and IMDCT to HW
- Mem1 (on OPB bus) for program, Mem2 (on LMB bus) for data
- Custom HWs on DoubleHdshk (DH) bus, with bridge to OPB
- Interrupt signals and connections are selected
- Arbiter is selected and request / grant pins are connected
- RTL code for Bridge is generated using BridgeGen
SW synthesis

- Compiler, RTOS and libraries are selected for SW
- Default addresses for all addressable memory/bus is generated by ESE
HW synthesis

- RTL code for HW components is generated using C → RTL tools
Export to FPGA Design Tools

- Platform and SW specification files are created for FPGA design tools
- C code for Microblaze and Verilog for HWs and Bridge is exported
Benefit: FPGA Prototype in 1 Week

- Bit stream from FPGA design environment is downloaded to board
- Implemented prototype is tested with MP3 music files
Design Quality: Manual

- **Area**
  - % of FPGA slices and BRAMS

- **Performance**
  - Time to decode 1 frame of MP3 data
- **Area**
  - ESE designs use fewer FPGA slices and more BRAMs than manual HW

- **Performance**
  - ESE designs execute at same speed as manual designs
Development Time: ESE vs. Manual

- **ESE drastically cuts RTL and Board development time**
  - Manual development includes months of RTL coding
  - Models can be developed at Spec level with ESE
  - TLM, RTL and Board models are generated automatically by ESE
Validation Time: ESE vs. Traditional

- **ESE cuts validation time from hours to seconds**
  - No need to verify RTL models for every design change
  - Designers can perform high speed validation with TLM and board
ESE Back-end Advantages

- HW synthesis in ESE removes the need to code and debug large RTL HDL models
- Transducer and interface synthesis allows flexibility to include heterogeneous IP in the design
- SW driver synthesis removes the need for SW developers to understand HW details
- SW and HW application can be easily upgraded at TL and validated on board
- C and graphical input of TL model allows even non-experts to develop and test HW/SW systems with ESE
Acknowledgments

- We would like to acknowledge the previous R&D teams who contributed many concepts and methods used in ESE 2.0
  - SpecCharts/SpecSyn (’92): F. Vahid, S. Narayan, J. Gong, S. Bakshi
- We also want to thank P. Chandraiah for MP3 reference code