Multiprocessor on Chip : Beating the Simulation Wall Through Multiobjective Design Space Exploration with Direct Execution

Riad Ben Mouhoub[1] and Omar Hammami[2]

[1]ENSTA
32 Bvd Victor
75739 FRANCE
riad.benmouhoub@ensta.fr

[2]ENSTA
32 Bvd Victor
75739 FRANCE
hammami@ensta.fr

Abstract

Design space exploration of multiprocessors on chip requires both automatic performance analysis techniques and efficient multiprocessors configuration performance evaluation. Prohibitive simulation time of single multiprocessor configuration makes large design space exploration impossible without massive use of computing resources and still implementation issues are not tackled. This paper proposes a new performance evaluation methodology for multiprocessors on chip which conduct a multiobjective design space exploration through emulation. The proposed approach is validated on a 4 way multiprocessor on chip design space exploration where a 6 order of magnitude improvement have been achieved over cycle accurate simulation.

1. Introduction

Systems on chip are increasingly becoming complex to design, test and fabricate. SoC design methodologies make intensive use of intellectual properties (IPs) [16] to reduce the design cycle time and meet stringent time to market constraints. However, associated tools still lag behind when addressing the huge associated design space exposed by the combination of soft IP. In addition, failure to meet an efficient distribution in terms of performance, area and energy consumption makes the whole design inappropriate. Although this problem is already hard to solve in the ASIC domain, it is exacerbated in the system on programmable chip (SoPC) domain. SoPC are large scale devices offering abundant resources but in fixed amount and in fixed location on chip. Implementing embedded multiprocessors on these devices present several advantages the most important being to be able to quickly evaluate various configurations and tune them accordingly. Indeed, embedded multiprocessor design is highly application driven and it is therefore highly advantageous to execute applications on real prototypes. However, due to the fact that specific resources are located at fixed positions on these large chips it is hard not take into account the important impact of place and route results on the critical paths and therefore on the overall performance. In this paper we address this multiobjective optimization problem [6] restricted to performance and area through the combination of an efficient design space exploration (DSE) technique coupled with direct execution on an FPGA board [2]. The direct execution removes the prohibitive simulation time associated with the evaluation of embedded multiprocessor systems. A side effect of this approach is that direct execution requires actual on chip implementation of the various multiprocessor configurations to be explored which provides actual post synthesis and place and route area information. The resulting flow is fully integrated from multiprocessor platform specification to execution. The paper is organized as follows. In Section 2, we review previous work. Section 3 describes an example of soft IP based multiprocessor and the breadth of the problem associated with the design of such multiprocessor on a particular instance embedded memories optimization. Section 4 presents our approach MOCDEX based on Multi-objective Evolutionary Algorithms (MOEA) and direct execution. In Section 5 we describe a case study and validation while Section 6 provides exploration results. Finally, we conclude in section 7 with remarks and directions for future works.

1-4244-0054-6/06/$20.00

2 Previous Work

Our work adresses three different points: (1) how to efficiently and automatically explore the design space of multiprocessors on chip ? (2) how to reduce evaluation time of multiprocessor configurations during design space exploration ? how to integrate implementation issues (i.e. chip area) in this exploration ? Traditional parallel computer architectures performance evaluation techniques do not provide automatic exploration besides exhaustive evaluation on a few parameters [8]. Paul and al [18] propose a technique called MESH a high level modeling and simulation technique of single chip programmable heterogeneous multiprocessors based on a layered approach. This technique does not integrate implementation issues and design space exploration is not fully automatic and restricted. Chidester and al [4] propose the use of parallel simulation for the evaluation of chip multiprocessor architecture through the use of a 9 nodes of dual cluster-CPU workstations. Although the evaluation of a single configuration is improved they neither address the issue of automatic exploration nor they tackle implementation issues. Simulation can be obtained in different modes. The most important ways for simulation are : Event-based simulators, cycle-based simulators, transaction based verification and cosimulation. The event-based simulation provides a functional and timing simulation results. It takes as input an HDL behavioural description, RTL code, gate level or transistor level netlist. Cycle-based simulation is well suited for Systems-On-Chips with an important number of processors and peripherals. The runtime of cycle-based simulation is improved by taking into account only the necessary calculations at each cycle clock edge. Another advantage of the cycle-based simulation is that it uses less memory in the host machine allowing simulation of larger designs (compared to event-based simulation) [20]. Even with the runtime improvements provided by the cycle-based simulation, verification of actual systems is still slow and unfeasible. The Transaction Level Modeling is an alternative approach for simulation where the aims were to provide a real improvement in runtime simulation and cycle accuracy for large scale SoC. This is due to the separation of communication and computation aspects of a design and by making abstraction of the communication by considering transaction requests instead [13]. TLM level shows performance improvements of 353 faster than in RTL simulation and a simulation speed able to reach 456 KCycles/sec. This system level modeling permits a rapid performance evaluation. Although, no real information about physical constraints of implementation are provided and till now there is no tested and approved tool assuming conversion of a TLM specification to a synthesizable RTL description. The idea of using FPGA platform for emulating parallel multiprocessors on chip have been used in the RPM [12] and RPM-2 [15] project. Although the approach is based on FPGA neither an automatic exploration flow nor a multi-objective design space exploration approach is proposed. Finally recently Copolla and al [14] proposed an open platform for developing multiprocessor SoCs based on emulation. However again no automatic and multi-objective approach is proposed.

To the best of our knowledge our work is the first to fully integrate and therefore close the gap between design automation tools and architecture design space exploration technique in a multi-objective constraints paradigm with actual execution for all multiprocessor on chip configurations explored during the design space exploration process.

3 Soft IP Based Embedded Multiprocessor System

Embedded systems based on soft IPs are SoC including; soft IP processors, interconnect infrastructure and memories. An example of such a system is described below. It is principally based on Xilinx Embedded Development Kit (EDK) IPs [21].

3.1 The Microblaze soft IP processor

The Microblaze soft [21] is a 32-bit 3-stages single issue pipelined Harvard style embedded processor architecture provided by Xilinx as part of their embedded design tool kit.

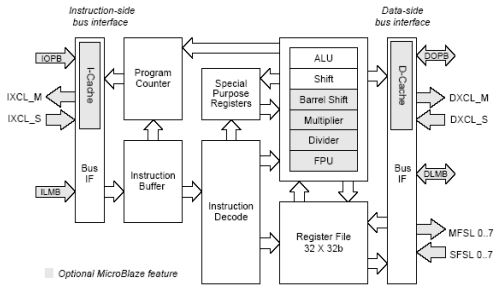

Figure 1. Microblaze soft IP processor

Both caches are direct-mapped, with 4 word cache line allowing configurable cache and tag size and user selectable cacheable memory area. Data cache uses a write-through policy. The Microblaze core configurability extends to functional unit through user selectable

barrel shifter (BS), hardware multiplier (HWM), divider (HWD) and floating point unit (FPU). The Microblaze has neither static nor dynamic branch prediction unit and supports branches with delay slots. For its communication purposes, the Microblaze uses either a bus or a direct link. The On-Chip-peripheral Bus (OPB) is part of IBM CoreConnect [21] bus architecture and allows the design of complete single processor systems with peripherals and user designed hardware accelerators (e.g. [3]). However, even for a simple embedded processor such as the Microblaze, the OPB bus is not suitable for multiprocessors designs because of its lack of scalability. Another approach is provided by "Fast Simplex Link" [21] which allows direct connection between embedded processors through FIFO channels.

Caches	Values
Instruction	1K, 2K, 4K, 8K, 16K, 32K, 64K
data	2K,4K,8K,16K,32K,64K

Table 1. Cache memories configurable values

3.2 Microblaze Fast Simplex Link

The Fast Simplex Link (FSL) [21] is an IP developed by Xilinx to achieve a fast unidirectional point-to-point communication between any two components. The FSL link is implemented as a 32-bit wide FIFO with configurable depth and width option. The FSL can be either a master or a slave interface depending upon its use.

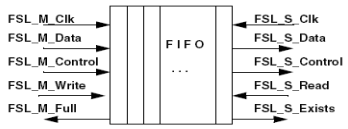

Figure 2. Fast Simplex Link

The Microblaze soft embedded processor allows up to 8 masters and slaves FSL interfaces. Basic software drivers are provided to simplify the use of FSL connection. They consist of read/write routines and control functions. The read/write routines can be executed in two different ways: blocking and non blocking mechanism.

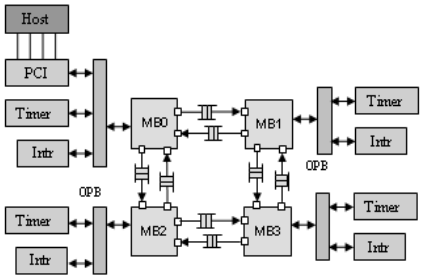

Figure 3. Mesh Platform 2x2

3.3 IBM Interconnect

The IBM Coreconnect [21] represents a set of IPs used to develop SOC devices. It includes the PLB and OPB bus, a PLB-OPB bridge and various peripherals.

3.4 MPSoC platform description

Our FPGA multiprocessor platform consists of four Microblaze processors with instruction and data cache units. These processors are connected with each other through FSL channels. Each Microblaze is connected, as shown in Figure 3 to an OPB bus, to use a timer and an interrupt controller for threads and OS execution. Microblaze MB0 is connected to the OPB bus which is connected to a PCI of the host bus (Work station). This allows the designer to send and receive data from the host to the multiprocessor system. We implemented a soft layer of communication in each Microblaze which performs send and receive functions of packets. The packets consist of headers representing the destination and source addresses and the number of flits in the payload. A wormhole routing algorithm was used since it uses less memory, making it suitable for network on chip communication. As it can be seen, a 4 way multiprocessor have been built based on the previously described soft IPs. The implementation of such a soft IP multiprocessor on FPGA platform requires a variable amount of resources as each soft IP composing the multiprocessor requires a variable amount of resources depending on the configuration options [21].

Such a soft IP multiprocessor can be easily adapted to the need of a particular application. However, these systems for best efficiency and low memory latency require the use of embedded on chip memories. Unfortunately, embedded memories are scarce resources for which processors instruction and data cache memories as well as bus and network on chip FIFO based interfaces will compete. This competition is dominated by the absolute requirement of efficiency in performance,

area and energy consumption [1]. If we focus on cache and FSL configurability we have for each cache memory 4 possible configurations and for each FSL 11 possible configurations. The design space associated with those parameters is $4^8 \times 11^8$ thus 14,048,223,625,216 different configurations requiring 26,727,974 years of simulation for 1 min simulation per configuration.

4 MOCDEX Multi-Objective Design Space Exploration

4.1 Problem Formulation

The design challenge represented by soft IP based multiprocessor design is a multiobjective optimization problem [6]. The multi-objective optimization problem is the problem of simultaneously minimizing the n components (e.g. Chips area, number of execution cycles, energy consumption), f_k , k = 1,..., n of a possibly non linear function f of a general decision variable x in a universe U where :

$$f(x) = (f_1(x), f_2(x), ..., f_n(x)) \quad (1)$$

The problem has usually non unique optimal solution but a set of non dominated alternative solutions known as the Pareto-optimal set. The dominance is defined as follows:
Definition 1: (Pareto dominance); A given vector $u = (u_1, u_2, ..., u_n)$ is said to dominate $v = (v_1, v_2, ..., v_n)$ iff u is partially less than $v (u_p < v)$, i.e. $\forall i \in \{1, ..., n\}\, u_i \leq v_i$ and $\exists i \in \{1, ..., n\}\, u_i < v_i$
Definition 2: (Pareto optimality): A solution $x_u \in U$ is said to be Pareto-optimal iff there is no $x_v \in U$ for which $v = f(x_v) = (v_1, v_2, ..., v_n)$ dominates $u = f(x_u) = (u_1, u_2, ..., u_n)$.
Pareto-optimal solutions are also called efficient, non-dominated and non inferior solutions. The corresponding objective vectors are simply called non-dominated. The set of all non-dominated vectors is known as the non-dominated set or the tradeoff surface of the problem.

4.2 Multi-objective Evolutionary algorithms

Multi-objective optimization has not been addressed properly by traditional optimization techniques (gradient based, simulated annealing, linear programming) since most of these techniques are mono-objective. Extending these techniques through approaches using aggregation functions does not represent true multi-objective optimization and does not produce multiple solutions. Multi-objective Evolutionary algorithms (MOEA) are more appropriate to solve optimization problems with concurrent conflicting objectives and are particularly suited for producing Pareto-optimal solutions. Several Pareto-based evolutionary algorithms have been proposed during the last decade SPEA-2,PESA and NSGA-II [6, 5] to solve multi-criteria optimization problems. The NSGA-II [7] is an MOEA considered to outperform other MOEA [19] and is briefly presented below.

The NSGA-II algorithm runs in time $O(GNlog^{M-1}N)$ where G is the number of generations, M is the number of objectives and N is the population size [19]. In addition, our previous experience on multi-objective optimization of soft IP embedded processor emphasizes this choice [11].

4.3 MOCDEX

It is clear that MOEAs such as NSGA-II requires the evaluation of individuals (MPSOC configurations) with regard to the 3 objectives considered, BRAM, slices and number of cycles. Although, BRAM and slices could be estimated, we advocate the full use of design automation tools including place and route to access this information. Indeed, for complex systems on large platform FPGA, place and route impact can not be overlooked and can hardly be estimated with sufficient accuracy to be used in an automatic multi-objective design space exploration tool. The execution time of multiprocessor on chip can be obtained through simulation either at RTL level which would be prohibitive for large design space exploration without massive use of computing resources (compute farms) or at TLM level (SystemC) as often advocated [9, 10]. However although, SystemC level simulation has been regularly proved to outperform RTL VHDL level simulation, it does not outperform actual execution on FPGA. We argue that for large scale MPSOC, platform FPGA represents an opportunity to both : (1) reduce simulation time through actual execution and (2) increase the design space exploration through this reduction of the evaluation of each MPSoC configuration. Our proposal follows:

MOCDEX

1. Generate random population of MPSOC configurations

2. For all configurations

    a. Generate hardware/software platform specification files

b. Generate through system EDA and IPs Hw/SW model of the MPSOC
c. Synthesize/place and route MPSOC configuration
d. Record place and route reports
e. Download configuration file on FPGA platform
f. Execute MPSOC configuration and record execution clock cycles
g. Rank the solution

3. Generate new population using MOEA algorithm
4. If the Pareto front is not satisfactory or the number of generations is not reached goto 3
5. Final Pareto front MPSOC configurations available for selection.

As shown in figure 4, both the DSE and physical design are executed on a host PC while the execution is achieved on PCI based FPGA platform which communicates execution results to the host.

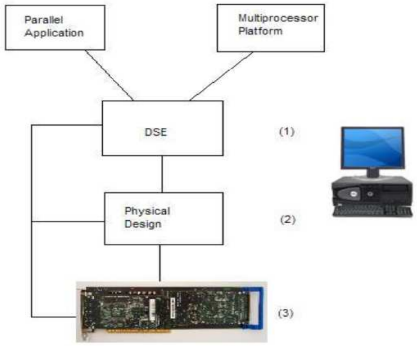

Figure 4. MPSoC exploration flow

5 Case study and validation

The previously described design flow have been applied in the framework of Xilinx FPGA platforms.

5.1 Filtering application

A design of four Xilinx Microblaze processors, communicating with eight FSL channels in a mesh topology and executing image filtering algorithms was implemented at 100 MHz in Xilinx Virtex-II 8000 device [22]. This application was chosen because it requires extensive data processing and data communication among the filters for a good and fast testing of our exploration framework.

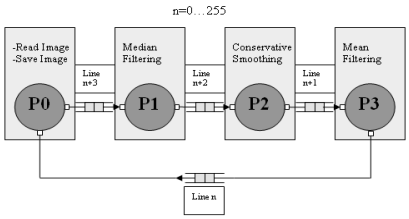

Figure 5. Filtering platform

The Figure 5 shows our filtering methodology. As we can see, the execution is achieved in a pipelined way where image lines are sent from a processor to another as soon as the previous processor has finished its work on it. Obviously, this type of execution makes us save a significant amount of time and memory which are often the major constraints for embedded systems in general and for our platform in particular. Indeed, performing this task in a pipelined way allows us to have a maximum of three image lines stored in the associated processor's memory rather than the whole image. The rest of the image lines will enter the FIFOs (FSLs) of their respective processors one by one. The processor P0 in Figure 5, receives image data from the host computer through the PCI bus. Once it receives the data it instantly send it to the next processor which is P1. P1 performs a median filtering which results in noise reduction from the image. It is performed on a 3 by 3 pixel window where the center pixel value is replaced by the median of the neighboring pixel values. This value is obtained by sorting the pixels based on their numerical values and then replacing the pixel to be processed by the middle value. The processor P2 fetches the line coming from P1 and performs a conservative smoothing on it which is an operation that preserves the high spatial frequency details. Finally, the third processor P3 performs a mean filtering which consists of very simple method used for noise reduction where the pixel to be processed is replaced by the average value of its neighbors. We can visibly mention that the three operations have clearly different behavior and use various arithmetic operators. Thus the execution time for each algorithm differs and hence involves an unequal FIFOs occupancy. We remind that the purpose of our study is to have an optimal distribution of memory utilization. Therefore, the application used has to be naturally unbalanced to thoroughly analyze the problem. At this time, we suppose that the filter algorithms are optimally implemented and the only improvement can be achieved from the effective sizing of

the FIFO interfaces and the different cache memories.

5.2 Exploration Results

For this work we initially executed two explorations where the first consisted of a population size of 22 individuals and 10 generations (Figure 6). Figure 7 represents Pareto solutions for the second exploration where we attempted to increase the population size to 30 individuals and the number of generations to 14 in order to observe the behavior of the evolutionary algorithm for bigger explorations. From the results of second exploration it is obvious that the algorithm is converging to optimal solutions showing that larger population size and generation number increase convergence. From these two exploration flow executions, we observed that we have not a very significant variation in the number of occupied slices, however the variations are much more significant concerning both the number of occupied BRAMs and the execution time. So we decided to continue the execution of the proposed exploration flow in order to see its evolution.

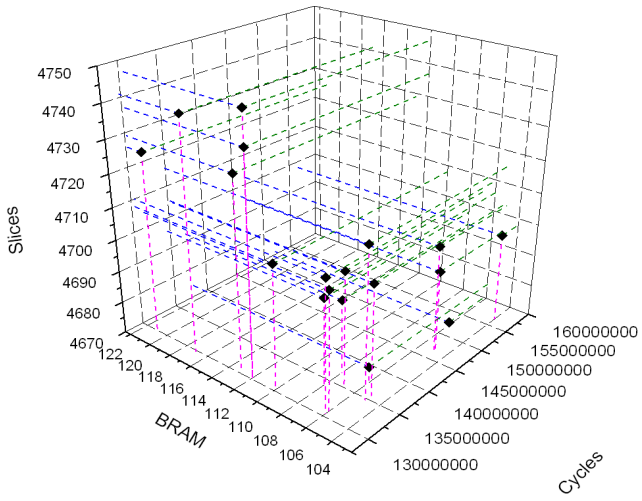

Figure 6. For 10 generations - popsize = 22

For this second part of the exploration, we fixed arbitrarily the population size to 30 individuals and changed the number of generation to 30 and finally 60 generations (Figure 9). From these different figures we can clearly observe that the NSGA-II evolutionary algorithm tends to converge to the optimal Pareto solutions front which proves the correct implementation of the algorithm.

The figures show different execution times for the same BRAM occupation meaning that using more BRAM will not systematically result in performance improvements. However, to achieve better results BRAM needs to be well distributed among the IPs where it would be used for getting optimal resource utilization.

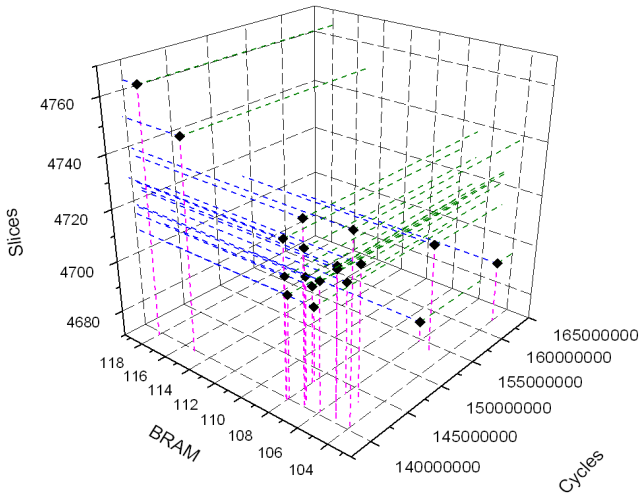

Figure 7. For 14 generations - popsize = 30

Users willing to invest more on design space exploration by increasing the number of generations and the population size, might be done in the same way that parallel computer architecture performance evaluation studies.

The run time of a single embedded multiprocessor platform for our chosen application takes 15 minutes on a 4GB P4 desktop WS and thus 7.5 hours for one population size of 30 individuals. 99.86% of the time is spent in the synthesis and Place/Route step. For resolving this execution time problem, we currently parallelize the NSGA-II evolutionary algorithm by executing concurrently the SPR step of a whole population on a workstations network, thus reducing the SPR time of a generation to the SPR time of one individual.
As an example, the tables 2 and 3 give two different individuals with there respective FSL and both instruction and data cache memory size configuration.

The configurations chosen represents respectively 69,64 %, 61,90 % and 64,88 % of all BRAM resources. 11,11 % BRAM reduction is obtained in the 2nd configuration for a 0.004 % increase in execution time while a 6,8 % BRAM reduction is obtained in the 3rd configuration for a 0.009 % increase in the execution time.

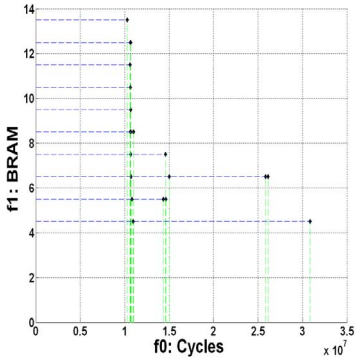

Figure 8. For 30 generations - popsize = 30

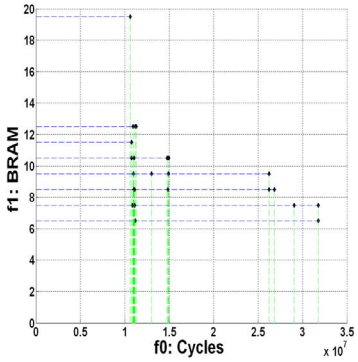

Figure 9. For 60 generations - popsize = 30

6 Discussion

The results achieved in the previous section required the performance evaluation of 3120 different multiprocessor on chip configurations. These evaluations have been cycle-accurate after actual implementation on single chip large scale FPGA devices. Contrary to traditional board based multiprocessor, multiprocessor on chip are implemented on single chip and due to the complexity of these architectures and the scale of the target devices, it is not possible to overlook the impact of place and route on the number of cycles required for various operations and on the cycle time. It results from this fact that comparing different multiprocessors on chip configuration on the number of execution cycles is meaningless if one does not take into account the impact of place and route on each distinct configuration resulting from actual implementation. From this point mainly two alternatives exist:(1) post place and route simulation which will accurately represents the multiprocessor on chip behavior (2) emulation through direct execution. We conducted cycle accurate simulations using a powerful multilangage (SystemC, VHDL, Verilog HDL) simulator Modelsim 6.0 [17].Indeed, ModelSIM6.0a SE can handle large and complex designs and allow their simulation in a behavioral, post-synthesis and post-Place&Route modes.Table 4 illustrates the very important time saving while using direct execution instead of simulation.

Procs	FSL1Out	FSL2Out	D-Cache	I-Cache
MB0	2048	2048	1024	4096
MB1	512	512	1024	1024
MB2	2048	512	2048	2048
MB3	1024	1024	4096	4096

Table 2. Pareto cycles: 138,844,064 BRAM:117

Procs	FSL1Out	FSL2Out	D-Cache	I-Cache
MB0	2048	128	2048	2048
MB1	256	32	2048	512
MB2	512	16	4069	512
MB3	1024	32	4096	2048

Table 3. Pareto cycles: 138,974,816 BRAM:109

Timings		
E.A (ms)	Indi. Gene.	190
	Obj Functions Eval.	293
	Selection	0.116
	Crossover	0.033
	Mutation	1.118
Synthesis (sec)	Synth.	523.503
	P and R	655.174
	P/R & Bitgen	797.856
Execution	Simulation 64x64	30 Hours
	Direct Exe. 256x256	2796.136 ms
Total	Explor. 60x30	
	Sim. Lena64x64	2250 Days
	Exec. Lena256x256	1.39 Hour

Table 4. Cycle Accurate Simulation vs Emulation

In order to reach the same speed simulation at this level of accuracy would require a compute farm (grip computing) of well over 25,000 workstations.

7 Conclusion

The design space exploration of complex multiprocessor on chips can hardly be conducted solely based on simulation due to prohibitive simulation time.Although various techniques have been developed to reduce simulation time they increasingly face a simulation wall resulting from the exponential increase of Moore's law. Reducing the design space by reducing the number of parameters to be tuned or by reducing the potential values of these parameters in order to alleviate this problem obviously prevent numerous potential performance improvements resulting in inefficient designs. In this paper we describe a fully automatic design space exploration flow for the design of multiprocessor on chips which is not based on simulation and take advantage of the reconfigurable nature of FPGA devices to evaluate multiprocessor on chips configurations through direct execution on single chip large scale FPGA. This automatic performance analysis is conducted in conjunction with chip area evaluation for each multiprocessor on chip through actual chip implementation for each multiprocessor configuration. The performance modeling and evaluation of such parallel systems is therefore achieved through a multiobjective design space exploration taking into account both performance and chip area with a 6 order of magnitude improvement over a cycle accurate simulation.

To the best of our knowledge our work is the first to address the automatic performance analysis of parallel systems (i.e. multiprocessors on chip) through multiobjective design space exploration with direct execution on reconfigurable large scale single chip device.

References

[1] A.A.Jerraya and W.Wolf. *Multiprocessor Systems-on-Chips*. Morgan Kaufmann, June 2004.

[2] Alpha-Data. Adm-xrc-ii pci mezzanine card. Available on: http://www.alpha-data.com/adm-xrc-ii.htm.

[3] R. Benmouhoub, I. Aouadi, and O. Hammami. System on programmable chip plateform based design of jpeg-2000 entropy coder. In *Workshop on synthesis and system integration of mixed information technologies (SASIMI'04)*, pages 103–106, Oct 2004.

[4] C.Matthew and A.George. Parallel simulation of chip-multiprocessor architectures. In *ACM Trasaction on Modeling and Computer Simulation*, volume 12, pages 176–200, July 2002.

[5] C. A. C. Coello. An updated survey of ga-based multiobjective optimization techniques. In *ACM Computing Surveys*, pages 109–143, June 2000.

[6] C. A. C. Coello, D. V. Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*, volume 5. Genetic Alrorithms and Evolutiunary Computation Kluwer Academic Publishers, May 2002.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm:nsga-ii. In *IEEE Transaction on Evolutionary Computation*, volume 6, pages 182–197, Apr 2002.

[8] D.E.Culler, A.Gupta, and J.P.Singh. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, 1997.

[9] F.Fummi, S.Martini, G.Perbellini, and M.Poncino. Iss-systemc integration for the co-simulation of multi-processor soc. In *DATE*, 2004.

[10] F.Ghenassia. *Transaction-Level Modeling with SystemC TLM Concepts and Applications for Embedded Systems*. Springer, 2005.

[11] K.Ghali and O.Hammami. Embedded processor characteristics specification through multiobjective evolutionary algorithms. In *IEEE International Symposium on Industrial Electronics*, volume 2, pages 907–912, June 2003.

[12] L.A.Barroso, S.Iman, M.Dubois, and K.Ramamurthy. Rpm: a rapid prototyping engine for multiprocessor systems. In *Computer*, volume 28, pages 26–34, Feb 1995.

[13] L.Cai and D.Gajski. Transaction level modeling: an overview. In *Hardware/Software Codesign and System Synthesis*, pages 19–24, Oct 2003.

[14] M.D.Nava, P.Blouet, P.Teninge, M.Coppola, T.Ben-Ismail, S.Picchiottino, and R.Wilson. An open platform for developing multiprocessor socs. In *IEEE Computer*, volume 38, pages 60–67, July 2005.

[15] M.Dubois, J. Jeong, Y. H. Song, and A.Moga. Rapid hardware prototyping on rpm-2. In *Design Test of Computers, IEEE*, volume 15, pages 112–118, july-Sept 1998.

[16] M.Keating and P.Bricaud. *Reuse Methodology Manual for System-On-A-Chip Designs*. Springer, 2002.

[17] ModelSim. Modelsim 6.0a se. Available on: http://www.model.com/.

[18] J. M.Paul, D. E.Thomas, and A. S.CASSIDY. High-level modeling and simulation of single-chip programmable heterogeneous multiprocessors. In *ACM Transactions on Design Automation of Electronic Systems*, volume 10, pages 431–461, July 2005.

[19] M.T.Jensen. Reducing the run-time complexity of multiobjective eas: The nsga-ii and other algorithms. In 5, editor, *IEEE Transactions on Evolutionary Computation*, volume 7, pages 305–515, October 2003.

[20] P. Rashinkar, P. Paterson, and L. Singh. *System-On-a-Chip Verification, Methodology and techniques*. Kluwer Academic Publishers, 2001.

[21] Xilinx. Embedded system tools guide. Available on: http://www.xilinx.com/ise/embedded/edk_docs.htm.

[22] Xilinx. Virtex-ii data sheet and user guide. Available on: http://www.xilinx.com/.