TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 15, No. 1, July 2015, pp. 78 ~ 86
DOI: 10.11591/telkomnika.v15i1.8085

Multilevel MPSoC Performance Evaluation, New ISSPT Model

A. Alali*[1], I. Assayad[2], M. Sadik[3]
Departement of Electrical Engineering, RTSE Team, HASSAN II University,
Ecole Nationale Supérieure d"Electricité et de Mécanique, Route D'El Jadida, Casablanca, Morocco
*Corresponding author, e-mail: hakim.alali@gmail.com[1], i.assayad@ensem.ac.ma[2],
m.sadik@ensem.ac.ma[3]

Abstract

To deploy the enormous hardware resources available in Multi-Processor Systems-on-Chip (MPSoC) efficiently, rapidly and accurately, Design Space Exploration (DSE) methods are needed to assess the different design alternatives. In this paper, we present a platform that makes fast simulation and performance evaluation of MPSoC possible early in the design flow, thus reducing the time-to-market. In this framework and within the Transaction Level Modeling (TLM) approach, we present a new definition of Instruction Set simulation (ISS) level by introducing two complementary modeling sublevels ISST and ISSPT. This later, that we illustrate an arbiter modeling approach that allows a high performance MPSoC communication. A round-robin method is chosen because it is simple, minimizes the communication latency and has an accepted speed-up. Two applications are tested and used to validate our platform: Game of life and JPEG Encoder. The performance of the proposed approach has been analyzed in our platform MPSoC based on multi-MicroBlaze. Simulation results show with ISSPT sublevels gives a high simulation speedup factor of up to 32 with a negligible performance estimation error margin.

Keyword: *multiprocessor systems, estimation of performance, MPSoC, TLM, SystemC, ISS, CABA, priority management*

1. Introduction

The literature shows that much of the design time is spent in the performance evaluation. In addition, the iterations in the design flow become prohibitive for complex systems. Therefore, achievement of high performance MPSoCs is a challenge. The solution is strongly linked to the availability of fast and accurate methods for the design and performance evaluation [1]. A modeling approach to reduce the time of design and validation time for MPSoCs is to use the Transaction Level Modeling models (TLM) [2]. So with TLM, we can validate the behavior for both the hardware and the software components of MPSoC platform as well as the interaction between them.

Besides, TLM cosimulation also allows the performance evaluation of the whole system at the earlier stages of the design flow before making a prototype, which is faster than HDL register-transfer level (RTL) simulation [3, 4].

For this work, an open source ISS is used and components modeled with SystemC language Version 2.2.0 [5] and TLM methodology, derived from SocLib [6].

We adopt a strategy for estimating the performance at two levels: Cycle Accurate Bit Accurate (CABA) and Instruction Set Simulator with priority management and timing ISSPT.

Our objectives in this publication are:

a) to develop a rapid exploration of performance of design MPSoC tool;
b) to show that the ISSPT model offers a better alternative than (a: fast simulation and imprecise) and (b: simulation with an increased accuracy but at the cost of longer simulation), but at the cost of an additional modeling effort. This latest effort is nevertheless quite acceptable in contrast to the loss of accuracy in (a) or loss of simulation speed in (b). In spite of these losses, (a) and (b) are now widely used in system evaluation, this is only because it lacked a better alternative.

The rest of this paper is organized as follows: an overview of related work on ex-isting platform of simulation with TLM for MPSoC is provided in section 2. Section 3 describes the

Received November 2, 2014; Revised January 26 2015; Accepted February 15, 2015

architecture of the multi-MicroBlaze system. Section 4 presents the simulation platform and modeling of ISSPT with round-robin approach. Section 5 describes the performance estimation in ISSPT. Section 6 presents the exemples of software. Finally in section 7 describes the results of the applications running on the platform.

2. Related Work

A lot of works on design exploration and performance evaluation for embedded systems MPSoC have been conducted. As a result of these researches, many of exploration environments are proposed, such as MILAN [7], STARSoC [8] and SimSoC [9]. The work presented in this paper can be seen as complementary to these environments.

Since the first appearance of TLM in 2000 [10], an increasing number of research projects have examined the problem of its definition, which led to several frameworks [11-13] and a multitude versions latest is TLM 2.0.1. All these studies have two factors in common:

1) TLM is featured on several levels;

2) The aspects of communication and computing platforms are separated.

Viaud and al. [14] were the first who proposed have an efficient TLM with timing modeling and simulation environment based on parallel discrete event principles. They obtained a long runtimes simulation factor but they did not measure this runtimes on real applications. Their model is also different from ours. Firstly, with our approach we can be applied for hierarchical or distributed MPSoC design, and secondly, it is open-source.

Kim [15] and Boukhechem [8] propose a new technique for HW/SW co-simulation for heterogeneous MPSoC platforms in timing model PVT, we have all advantages of PVT that we refined in order to add it as a priority management. Also we integrated computation and communication simulation.

3. Architecture

The basic architecture of the platform implemented in VHDL and generated from Xilinx Platform Studio, consists of: 1, 2 or 3 MicroBlazes each one connected with a private memory 64 KB BRAM via the LMB bus processors. Processors are also connected to the OPB bus [17], SRAM memory of 32MB [16], an interrupt handler, VGA controller, timer and GPIO. A high-level view of the architecture of multi-core MicroBlaze is illustrated in Figure 1.

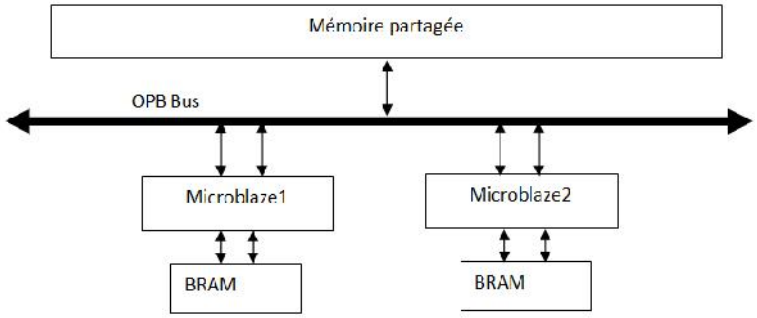


Figure 1. Multi-MicroBlaze basic architecture

The same concept that is implemented in VHDL, it is implemented in SystemC environment. Figure 2 summarizes our framework proposal.

In our case study, in ISS (Instruction Set Simulator) simulation and by using inter-process communication, we connect two ISSs (two processors) with SystemC communication models. Therefore, it is easy to add or to remove a processor from the MPSoC design. The interconnection is based on OPB bus [17] described in TLM SystemC. Communication model uses communication mechanism for the shared memory, the bus arbitration mechanism is managed by bus arbiter which implemented by the round-robin arbitration policy (described in Network interconnection model section).

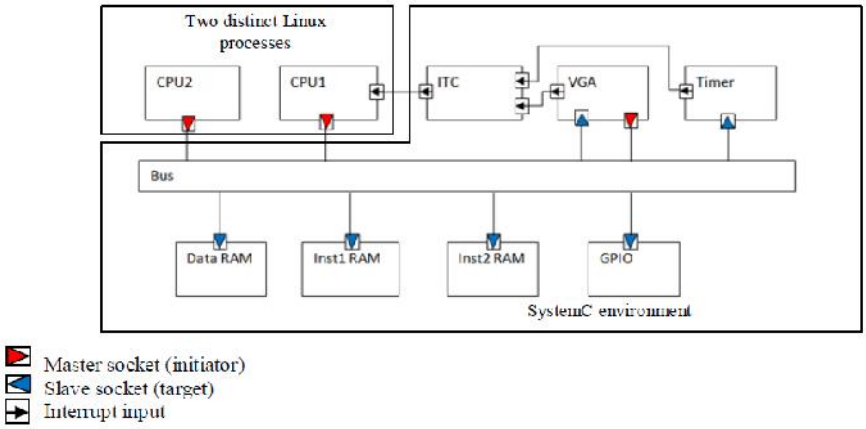


Figure 2. Platform architecture with 2 processors

4. SystemC Simulation Platform

Provide a statement that what is expected, as stated in the "Introduction" chapter can ultimately result in "Results and Discussion" chapter, so there is compatibility. Moreover, it can also be added the prospect of the development of research results and application prospects of further studies into the next (based on result and discussion).

4.1. Processor Model and Simulation

With the TLM approach, the behavior of a processor has two major descriptions ISS and CABA (Cycle Accurate/Bit Accurate). In the ISS, the processor description is modeled with a specific instruction level simulator. Instructions are executed sequen-tially without referencing to the micro-architecture of the component. CABA models the behavior of the system at each cycle similar to the RTL level, Indeed, the CABA level modeling is based on the theory of "finite state machine (FSM) interconnected synchronous" (Synchronous Communicating Finite State Machines) [19-21].

The estimated performance in new ISSPT (Instruction Set Simulation with timing and priority management) level returns to evaluate performance of two parts calcula-tion and communication time.

For calculation time, to assess the time of each task we used the simulator Micro-Blaze processor ISS level but adding time. For this we mainly identified the number and type of instructions executed as relevant activities in the processor component.

Timing execution instructions of MicroBlaze processor is estimated from the tech-nical documentation provided by Reference Guide of MicroBlaze [16].

Below is an example of our thread implementation to implement the functionality of the calculation part (processor) described in level ISSPT. For communication time is detailed in Bus and Network interconnection model section.

```
void MicroBlazeIss::step(void) {
/* decode of instruction outstanding */
IDecode(m_ir, &ins_opcode, &ins_rd, &ins_ra, &ins_rb, &ins_imm);
switch (ins_opcode) {
//execution of instruction
case OP_ADD:
next_pc = r_npc + 4;
Wait(ADD_delay,sc_core::SC_NS)
break;
……
//load of data
case OP_LW:
…..
```

```
LOAD(READ_WORD, addr, time);
next_pc = r_npc + 4;
Wait(Transaction_delay,sc_core::SC_NS)
break;
……
}
}
```

4.2. Memory Model

The memory module that we designed is a passive component "slave" type is cur-rency in two parts, one for instructions and one for data and transaction includes two methods: read and write. This structure allows us to accelerate the simulation. These two methods are called and executed directly in the thread initiator connected to the memory component.

In our environment, the target port is connected directly to the bus. Data part memory is shared between the processors. Access time and cycle time parameters are added to the component description to estimate performance.

4.3. Bus and Network Interconnection Model

Our architecture platform is designed around the OPB bus(On-Chip Peripheral Bus) that its architecture was developed by IBM [17]. The bus supports various features depending on the desired bus operations: single cycle read/write, multiple masters, block transfer. In our work, we use OPB Bus connected Xilinx MicroBlaze processor.

In this paper, we have limited develop an integrated Bus crossbar, which is based on two main features routing and arbitration see Figure 3 and 4. The router is a generic component that directs a request from an initiator to the target in question, using a routing table specified. When a new transaction get from initiator, the router reads the corresponding addresses and selects an output port, this process is illustrated in Figure 3.

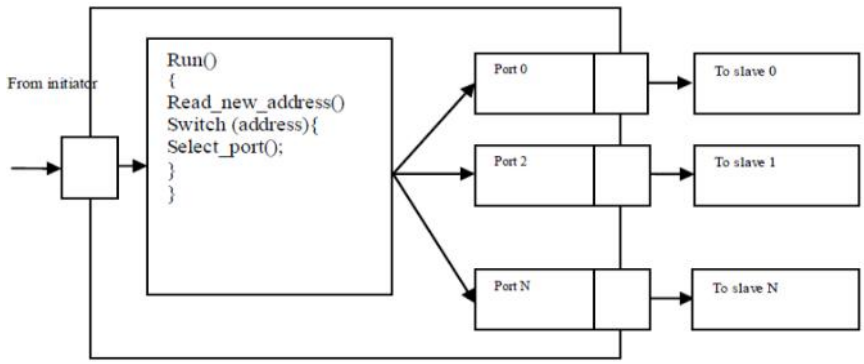


Figure 3. Router component

To manage conflicts between multiple simultaneous requests to a target, we developed an active component called "arbiter" to schedule the access to shared resources. When initiator needs access to a shared target, via the interconnection network, it sends a request using the corresponding communication channel and waits for the response. At the arbiter, one thread reads queries present in the FIFO of each communication channel and selects the priority request based on the arbitration round-robin strategy. After processing by the target, the arbiter transmits the response in the corresponding FIFO of the communication channel.

The initiator retrieves the response and completes the transaction. This communication management blocks a router during the processing of the request by the target, which can be a drawback. However, it has the advantage, simplification of the protocol and reduced the number of ports. These two factors can accelerate the simulation. Our arbiter plays a second role very important, is used for the estimation of delays in the interconnection network.

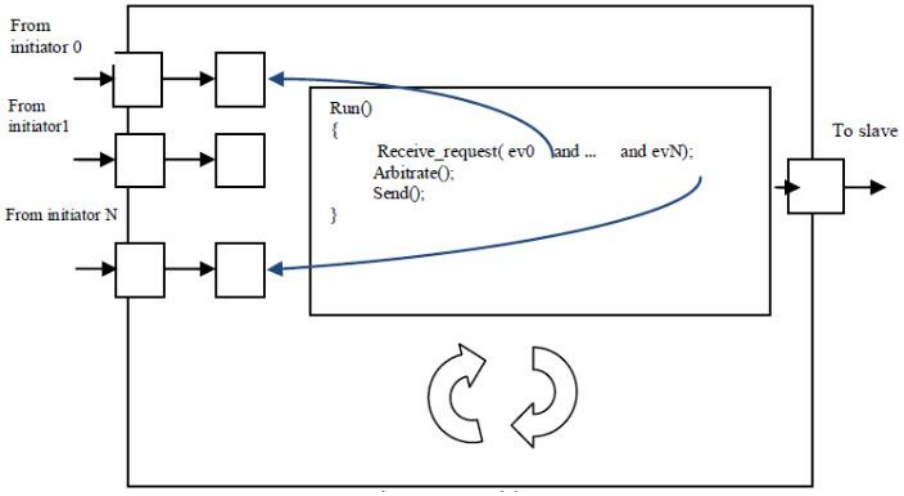


Figure 4. Arbiter component

Figure 5 shows the implementation of a crossbar from the two modules "router" and "arbiter". This architecture is relatively simple, but sufficient to achieve our objective to observe the restraints and retrieve information about latencies. Several interconnection topologies can be designed as multi-stage network.

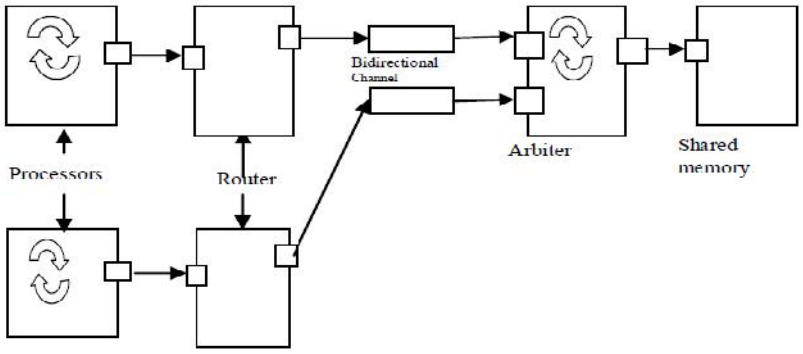


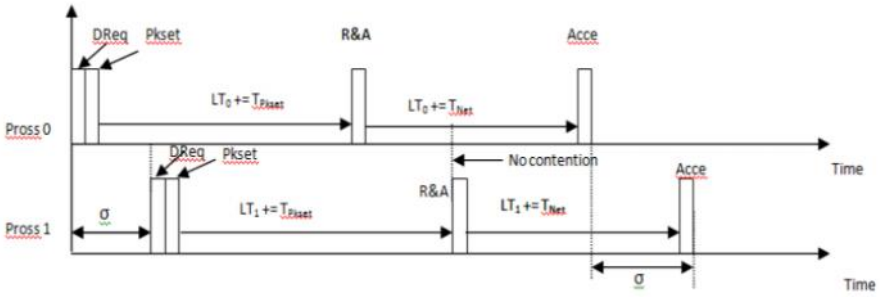
Figure 5. Crossbar implementation

5. Performance Estimation in ISSPT

The moment when a processor performs its corresponding memory access can affect the access time of the other processor in a collision. Figure 6 shows an example of contention detection error in the interconnection network due to non -compliance with the deadlines events (Packet setup, Routing and Arbitration). When the transmitted packet from processor 1 arrives at the router (R & A in Figure 6), there is no possibility of detecting the occupation of the router by the processing of the packet coming from processor 0. In effect, events in the sub-level ISST are instantly executed (zero delay). This abstraction changes the behavior of the party in the communication system, which reduces the precision of performance estimation. To solve this problem, we have improved the sublevel ISS by introducing synchronization instructions. They take into account the time of component activities, delays in forwarding packets and finally the communication protocol. These are the characteristics of the ISSPT level.

To compare the estimation error between ISSPT level and CABA level in our platform, we had to implement the specifications of the OPB protocol. To emulate the same behavior of the OPB protocol in ISSPT sublevel and observe the behavior of components, the wait () statements have been added in the description of the components and before the transmission

of orders and answer queries. The wait () statements added require arguments expressed in units of time such as nano seconds (ns) or number of cycles. In our experiments, these arguments are measured from CABA platform. Table 1 shows time made in the ISSPT level.



LT0: Local Timer 0; **LT1**: Local Timer 1; **DReq**: Data Request
Pkset: Packet Setup; **R&A**: Routing and Arbitration; **Acce**: Memory Access
T_{Pkset}: Packet Setup time; T_{Net}: Network time

Figure 6. Timing estimation in ISSPT sublevel

Table 1. Time activities used in the experiments

Activities	Time (cycles)
Preparing an OPB command request	4
Preparing a response OPB request	5
Execution of an instruction	1
read memory access	2
write memory access	2
VGA	360000

6. Software Integration

The application layer has two software was tested in the platform:

a) The game of life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, alive or dead. Every cell interacts with its eight neighbors, which are the cells that are horizontally, vertically, or diagonally adjacent [18],

b) JPEG Encoder is a minimalistic JPEG encoder written in C. It is both “portable” (tested on x86 and MicroBlaze) and “lightweight” (around 600 LOC). Application allows us to write JPEG compressed images from input image data on memory. It works in “grayscale only” (monochrome JPEG file): there is no support for color so far.

1) It produces baseline, DCT-based (SOF0), JFIF 1.01 (APP0) JPEG-s,
2) It supports “8x8 blocks only”,
3) It includes default quantization and Huffman tables that are not customizable at runtime.

Generally, for each application, it executed by 1, 2 or 3 processors, it is stored in their local memory and they executed in parallel and synchronized by the same clock system.

7. Results and Discussion

We present results that we carried out to validate our platform and evaluate its performances. We also compare performances among the different abstraction levels.

The same environment was employed for simulation on different abstraction levels. The results of simulation were gotten by running the platform on a core 2 duo with a RAM memory size of 1GB, based on Linux Fedora 8 Core 3.1. ISS was built by the GNU cross-compiler (GCC version 3.4.6).

7.1. Simulation Results in CABA, ISST and ISSPT

Figure 7 and 8 shows Speedup and precision simulation results for CABA, ISST and ISSPT with Game of Life as a software, and the same simulation shown in Figure 10 and 11 with JPEG Encoder. Speedup corresponds to the simulation or execution time of software at the different simulation abstraction levels.

t2 = „end time", t1 = „start time".
Speed-up formula:

$$\frac{(t2-t1)x}{(t2-t1)bit}$$

Precision formula: Δx -Δbit
With „x" = „CABA"," ISST" or „ISSPT".

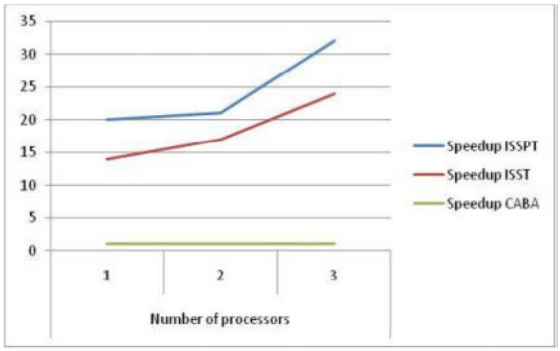


Figure 7. Speedup simulation results for CABA, ISST and ISSPT, software used: Game of life

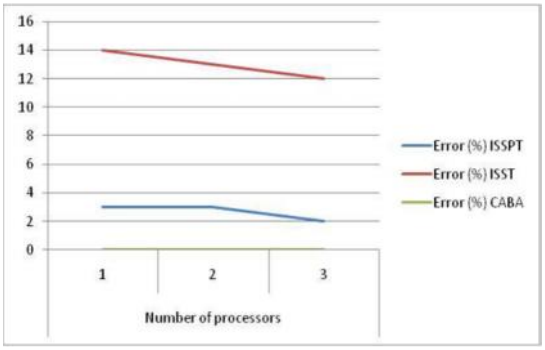


Figure 8. Precision simulation results for CABA, ISST and ISSPT, software used: Game of life

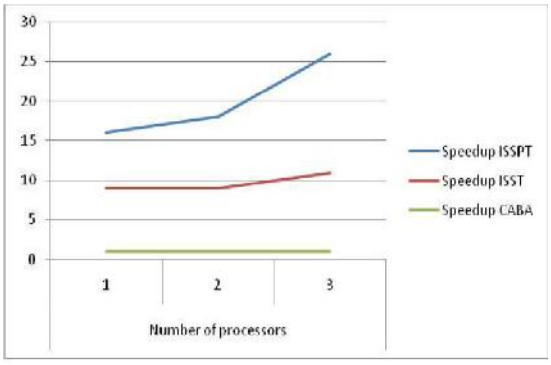


Figure 9. Speedup simulation results for CABA, ISST and ISSPT, software used: JPEG Encoder

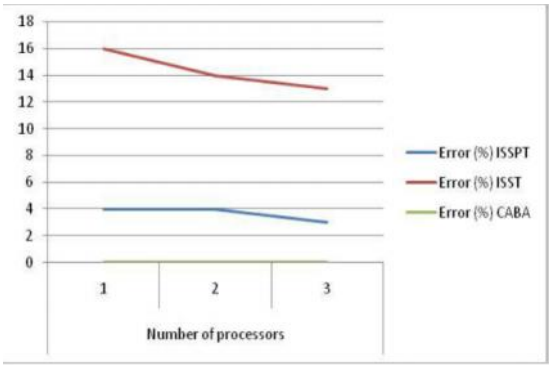


Figure 10. Precision simulation results for CABA, ISST and ISSPT, software used: JPEG Encoder

Models were simulated: ISSPT, ISST and CABA, we tested CABA model by its comparing a synthesizable RTL model (using VHDL generated from Xilinx Platform Studio and simulated with ModelSim) [19]. The first two models are used to validate the software and the system architecture (they include the ISSs, bus model at transaction level, and models of other components, all blocks use SystemC models).

We validated the simulation by the same Testbenches, after running the environment we obtained these experimental results:

a) The CABA has an important precision [19].
b) ISSPT model is about 20 times faster than the CABA model.

c) ISSPT model is about 2 times faster than the ISST model.
d) The addition of new processors in the system increases the acceleration factor, which is explained by the amplification of the communication between the processors and shared memory modules.
e) The nature of the software running on the platform impacts performance in differences levels.

A precise analysis of the trace produced by the SystemC simulator shows that 80% of the simulation time is made for the execution of the function of the bus while the simulation time of the calculation part is low which reflects our choice to treat the case of ISSPT.

7.2. Modeling Effort

So far we have shown the usefulness of our approach in terms of acceleration of the simulation and in terms of performance estimation. However, this approach has proven effective also in terms of modeling effort. It allows designers the development and validation of MPSoC systems in less time. Table 2 presents the modeling effort expressed in terms of lines of code (LOC) needed to design an MPSoC system in the CABA and ISSPT levels. According to the results, the modeling effort with ISSPT is reduced of a factor of 59%. The use of a multi-level simulation strategy (with objectives) quickly allows focusing on a subset of MPSoC systems without having to increase the modeling efforts for each level of abstraction.

Table 2. Comparing the modeling effort

Abstract level	CABA		ISSPT
Modeling effort(LOC)	Processor	1578	1259
Bus	399		170
Memory	312		133
VGA	650		167
Timer	340		231
Total	3279		1960
Reduction (%)		59%	

8, Conclusion

In this paper, we have presented and validated our methodology for MPSoC cosimulation at a high level of abstraction (SystemC-TLM) within a single simulation environment based on SystemC language. Our environment is based on the use of open source ISSs models of MicroBlaze wrapped under SystemC by using UNIX inter-process communication. Comparing three different abstraction levels, namely, ISSPT which implement ISS (Instruction Set Simulator) with priority and timing management, ISST level which implement ISS with timing and finally, CABA Cycle Accurate Bit Accurate.

The experimental results show that the use the ISSPT approach with SystemC-TLM reduces the design validation time and permit developing models rapidly with an acceptable precision.

This motivates our choice for SystemC and TLM as a system design methodology, dedicated to architecture exploration in our project which is the main contribution of this work.
As perspective, we think to develop models for estimating the energy consumption at different levels.

References

[1] L Benini, et al. MPARM: Exploring the Multi-Processor SoC Design Space with SystemC. *Springer J. of VLSI Signal Processing*. 2005.
[2] FRANK GHENASSIA. TLM with SystemC Concepts and Applications for Embedded Systems. 2005.
[3] AA Jerraya, A Bouchhima, F P´etrot. *Programming models and HW-SW interfaces abstraction for multiprocessor SoC*. In Proceedings of the 43rd Annual Conference on Design Automation (DAC ‟06). San Francisco, Calif, USA. 2006: 280-285.
[4] K Hines, G Borriello. *Dynamic communication models in embedded system co-simulation*. In Proceedings of the 34th Design Automation Conference (DAC ‟97). Anaheim, Calif, USA. 1997: 395-400.
[5] Systemc homepage, http://www.systemc.org/.

[6] http://www.soclib.fr/trac/dev/wiki/Component.
[7] S Mohanty, VK Prasanna, S Neema, J Davis. *Rapid design space exploration of heterogeneous embedded systems using symbolic search and multi-granular simulation*. In Conference on Languages, compilers and tools for embedded systems. Berlin, Germany. 2002.
[8] S Boukhechem, EB Bourennane. *TLM platform based on systemC for STARSoC design space exploration*. In AHS '08, NASA/ESA Conference. Noordwijk. 2008.
[9] C Helmstetter, V Joloboff. SimSoC: A SystemC TLM integrated ISS for full system simulation. In APCCAS. Macao, China. 2008.
[10] D Gajski, et al. SpecC: Specification Language and Methodology. Kluwer. 2000.
[11] A Donlin. *Transaction level: flows and use models*. In CODES+ISSS "04. Stockholm, Sweden. 2004.
[12] L Cai al. *Transaction level modeling: an overview*. In CODES+ISSS "03. New York, USA. 2003.
[13] L Benini, *et al*. SystemC cosimulation and emulation of multiprocessor SoC designs. *IEEE Computer.* 2003; 36(4).
[14] E Viaud, F Pecheux, A Greiner. *An efficient TLM/T modeling and simulation environment based on parallel discrete event principles*. In DATE"06. Munich, Germany. 2006.
[15] D Kim, Y Yi, S Ha. *Trace-driven HW/SW cosimulation using virtual synchronization technique*. In Design Automation Conference" 05. Anaheim, California. 2005.
[16] MicroBlaze Processor v5.4. Reference Guide, UG081 (v5.4). 2006.
[17] On-Chip Peripheral Bus Architecture Specifications V2.1.
[18] http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life.
[19] A Alali, I Assayad, M Sadik Modeling and simulation of multiprocessor systems MPSoC by SystemC/TLM2. *International Journal of Computer Science Issues (IJCSI).* 2014; 11(3).
[20] M Sgroi, L Lavagno, A Sangiovanni-Vincentelli. Formal Models for Embedded System Design. *IEEE Design and Test of Computers.* 2000; 17(2):14-27.
[21] L Lavagno, A Sangiovanni-Vincentelli, E Sentovich. System-level synthesis, chapter Models of computation for embedded system design. Kluwer Academic Publishers. 1999: 45-102.