Home

A Configurable Platform for Simulating Multiprocessor Based System-on-Chip

Muhammad Usman Ilyas[§]
Department of Computer Science, Lahore University of Management Sciences, Lahore 54792, Pakistan
Email: muilyas@lums.edu.pk

Syed Ali Khayam[§]
Department of Electrical & Computer Engineering, Michigan State University, East Lansing, MI 48824, USA

Muhammad Omer Suleman[§]
Computing Laboratory, Oxford University, Oxford OX1 3DQ, United Kingdom

Shahid Masud
Department of Computer Science, Lahore University of Management Sciences, Lahore 54792, Pakistan

Abstract – This paper presents a method to create a model for simulating the top-level system design of System-on-Chip using a sequential language. The model constitutes a cycle accurate simulation of the system under test at the required level of abstraction. The need for such a configurable platform arises during the initial system level design phase. The approach described in the paper has been successfully applied in the system level verification of a commercial media processor where the primary application is in VoIP media gateways. The implementation is primarily focused on monitoring and verification of inter-processor communication (IPC) among processing engines in the media processor.

I. Introduction

For years now verification teams of large projects have been struggling to keep up with the increasing gate count of complex ASICs. Project managers need to allocate more and more time to verification and testing phase in the design cycle. The complexity of System-on-Chip (SOC) is now well above ten million gates [1]. The requirement of extensive simulation and testing for this kind of gate complexity has led to a scheme whereby complex hardware modules of the system are replaced with simplified Bus Functional Models (BFM) during verification [2].

The verification and testing of SOC designs takes up a significant fraction of product development cycle. The designs have to be simulated and tested at different levels at different stages of development. First the design is tested functionally during the initial system level design stage and later a more detailed testing is done at register transfer level using the RTL code. Simulations at both levels of detail are traditionally implemented in Hardware Description Languages (HDL) [3].

II. Traditional Simulation Techniques

The verification of complex SOC designs is done using a combination of RTL and high-level behavior description in an HDL such as Verilog or VHDL. Traditionally both the high-level system design as well as the low-level register transfer level testing is done using models implemented in HDL [4]. However, testing using HDL models has following two major disadvantages:
a. An HDL simulation executes at a much slower rate compared to an equivalent piece of code in a sequential language that is functionally identical, and
b. Additional cost of the HDL software tools and the computer platforms required to execute them is significant.

The work presented in this paper addresses both these issue through the replacement of high-level system components by their respective BFM. These functional models not only speed-up the design cycle by orders of magnitude but result in significant reduction in the overall design cost. The rest of the paper is organized as follows: Section III outlines the development of the BFM approach for SOC functional testing. Section IV discusses the application of this approach in a VoIP media multiprocessor SOC development. Section V presents the results followed by conclusions.

III. Outline of the Model

An alternative to using detailed RTL code implemented in HDL is the use of models implemented in sequential languages [5, 6]. This section describes how the BFM model was successfully implemented in a sequential language (Visual C++). Listed below is a list of features of the model.

[§] **These authors were previously at AVAZ Networks, Software Technology Park, Constitution Avenue, Islamabad, Pakistan**

A. Parallelism

Since the model has been implemented in a sequential language, explicit provisions had to be made in the code to ensure the simulation of a parallel execution. The details are given later in the paper.

B. Cycle Accuracy

All events and changes in values of variables in the model are synchronized with the simulator's global clock. The execution of the model is also timed with the global clock.

C. Component Mapping

As explained above, the mapping of components is very straightforward for most designs. Figure 1 depicts a simple mapping scheme. The white boxes represent modules and gray boxes represent the classes to which they are mapped. The details of correspondence between specific HDL-types to classes in C++ are given below:

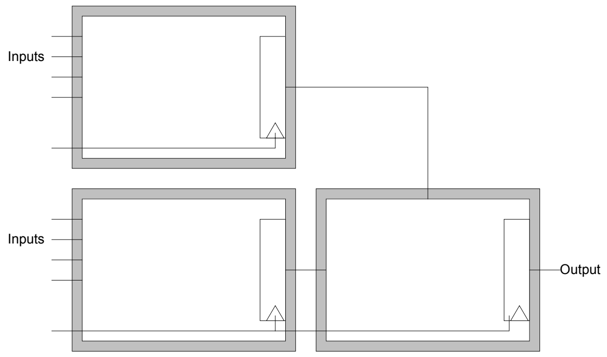


Fig. 1: Simple module to class mapping

i. Module to Class Mapping

Each instance of a module is directly mapped to an object, such as in figure 1. Since the model is applied at the system-level, it can be assumed that the inputs and outputs of the modules are registered. The inputs and outputs of each module instantiated in the model change synchronously with the system clock. In designs where this is not the case; an example is shown in figure 2, the boundaries of the classes are aligned with the boundary of clustered groups of modules all of whose input and output ports are registered. Figure 3 shows the boundaries of the classes in the corrected model. This limitation is due to the reason that data between classes can only be exchanged between clock cycles.

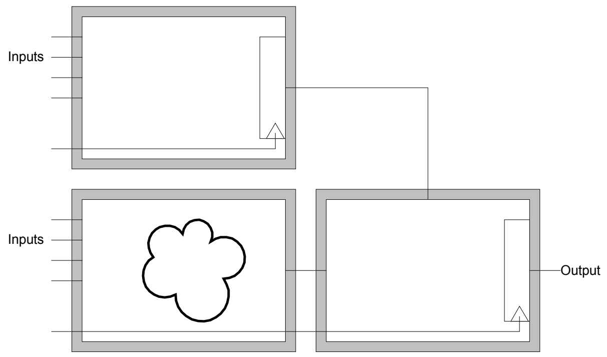


Fig. 2: Incorrect module to class mapping

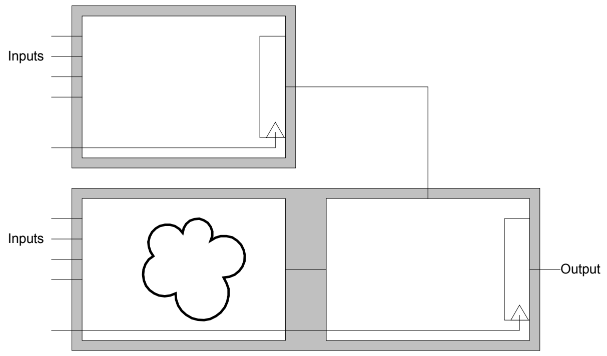


Fig. 3: Corrected module to class mapping

ii. Registers and Memories

Registers are simply modeled as pairs of variables. One variable contains the actual value of the register that is visible to all modules for reading. The other variable is a shadow variable and represents the value at the register's input port at the clock edge. In case of a pipeline register, the register variable is replaced by its corresponding shadow variable every clock cycle. In case of a register with a write-enable signal, there is an additional condition that the write-enable signal be asserted.

Similarly, memories are modeled as pairs of memory arrays and corresponding shadow arrays. The update mechanism of the memory array is similar to that of the register variable.

iii. Combinational Logic

The modeling of modules containing combinational logic is similar to that of registers with one difference. The input to the register variable is preceded by a logical expression the result of which is then fed to the input of the register. The code sequence in figure 4 depicts a module consisting of a combinational logic circuit with a registered output.

Top

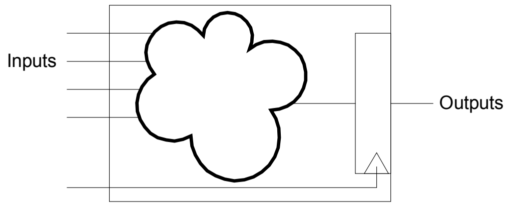


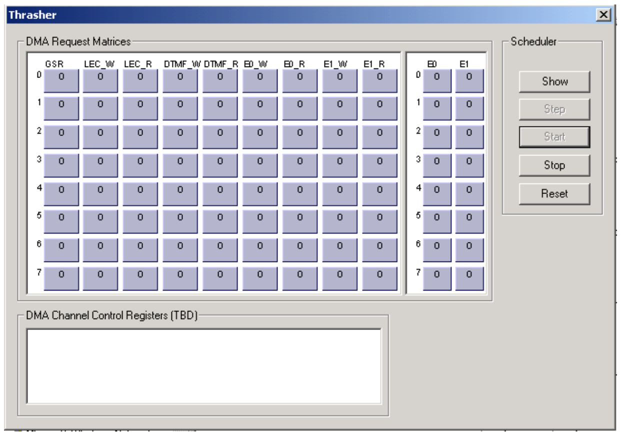
Fig. 4: Schematic of a mapable combinational logic module.

iv. Finite State Machines

Finite State Machines (FSM) have been modeled by a switch statement in C++ and a register variable in which to store the current state of the FSM. Depending on the type of FSM, the model may contain a second register variable to store the “next state”.

IV. Application in the Development of VoIP Media Processor VZM-1000

The modeling approach described in this paper was successfully applied to the verification and testing of the AVAZ Networks VZM-1000 Media Gateway processor. This SOC consists of a number of RISC Processing Elements (PE). Each element is capable of receiving messages from other PEs and executing tasks accordingly. The communication and synchronization of tasks on PEs is performed by a complex IPC network. The IPC network is dedicated to the dispatching, decoding and routing of messages between the processing elements. The purpose of the simulator was to verify the proprietary network protocols used in it. The model was implemented using C++. Figure 5 depicts a screenshot of the Graphical User Interface (GUI) showing the state of each PE in the system.



	GSR	LEC_W	LEC_R	DTMF_W	DTMF_R	E0_W	E0_R	E1_W	E1_R
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0

	E0	E1
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0

Fig. 5: This figure shows the GUI of the system-level simulator for the Media Processor

Figure 6 shows a screenshot of another window of the verification application. This window shows the schedule of the tasks dispatched over the IPC network.

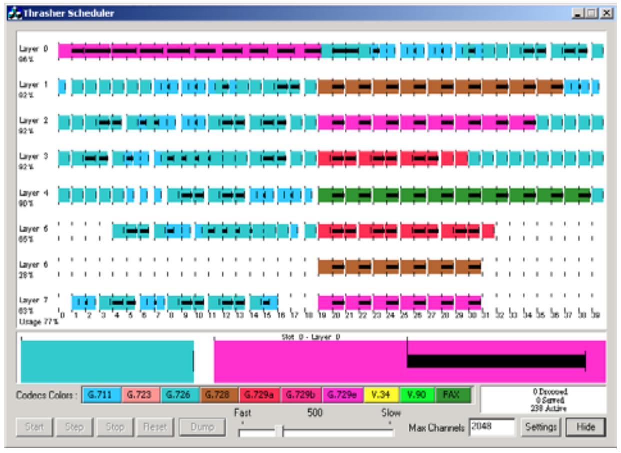


Fig. 6: This figure shows the system state of the simulator during execution

V. Results

For the purpose of comparison, the IPC network was also implemented in Verilog HDL. The VZM-1000 has now been implemented as an ASIC in 0.13µ technology. The IPC network on the processor is performing precisely as simulated. Some observations from the performance of BFM approach are presented below:

A. Code Complexity

The level of complexity of the source code of RTL (in HDL) as well as BFM models (in C++) was approximately the same. This was achieved by implementing functionality at the same levels of abstraction.

B. Lower Execution Time

The C++ model achieves a considerable improvement in speed and flexibility over the Verilog HDL model. The C++ model of the system executes approximately two orders of magnitude faster than the Verilog HDL model. Both models were described in the system at the same level of abstraction and were tested using an identical set of input vectors [7, 8].

C. Shorter Time-to-Market

The reduction in time taken for the initial design iterations resulted in improving the time-to-market of the design from 18 months to about 10 months.

Top

VI. Conclusions

The implementation results show that sequential languages such as C and C++ readily lend themselves to replace HDLs as the dominant design verification tool during the initial system-level design phase. The detailed register-transfer level verification should be performed on the frozen RTL that is used for synthesis.

However, using the sequential language model approach at different stages of the design phase has its drawbacks. The foremost among them is the fact that RTL description of the design is the input to the synthesis tool. Therefore the low-level HDL model of the design has to be implemented in a complete design in all cases. The sequential language coding may result in extra effort on the part of designers.

Acknowledgements

The work presented in this paper was done with the help of Dr. Shoab Ahmed Khan of AVAZ Networks. We are also grateful to our colleagues at AVAZ Networks for their cooperation.

References

[1] Keith Westgate and Don McInnis, “Cycle-based simulation”, http://www.quickturn.com/tech/cbs.htm on December 19th, 2002.

[2] Patrick A. McCabe, “VHDL-based system simulation and performance measurement”, VHDL International Users’ Forum Meeting, May, 1994, Oakland, CA USA.

[3] Namseung Kim, Hoon Choi, Seungjong Lee, Seungwang Lee, In-Cheol Park and Chong-Min Kyung , “Virtual chip: Making functional models work on real target systems”, 35th ACM DAC98 , June, 1998, San Francisco, CA USA.

[4] Luc Semeria, Andrew Seawright, Renu Mehra, Daniel Ng, Arjuna Ekanayake and Barry Pangrle, “RTL C-based methodology for designing and verifying a multi-threaded processor”, DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

[5] Moon Gyung Kim, Byung In Moon, Sang Jun An, Dong Ryul Ryu, and Yong Surk Lee, “Implementation of a cycle-based simulator for the design of a processor core”, IEEE Asia-Pacific ASIC conference (AP-ASIC), 1999.

[6] Namseung Kim, Hoon Choi, Seungjong Lee, Seungwang Lee, In-Cheol Park and Chong-Min Kyung, “Virtual Chip: Making Functional Models Work On Real Target Systems”, 35th ACM DAC98, June, 1998, San Francisco, CA USA.

[7] H. Al-Asaad, D. Van Campenhout, J. P. Hayes, T. Mudge and R. B. Brown, “High-level design verification of microprocessors via error modeling”, Proceecdings IEEE International Workshop on High Level Design Validation and Test, Nov. 1997, pp. 194-201.

[8] Yufeng Luo, Tjahjadi Wongsonegoro and Adnan Aziz, “Hybrid Techniques for Fast Functional Simulation”, 35th ACM DAC98, June, 1998, San Francisco, CA USA.

Top