

Hardware/Software System Co-Verification of an Active Reconfigurable Board with SystemC-VHDL

Yan Basile-Bellavance¹, Étienne Lepercq¹,
Yves Blaquière² and Yvon Savaria¹

(1) GR2M, École Polytechnique de Montréal,
(2) Département d'Informatique, Université du Québec à Montréal

Abstract— This paper reports on the challenges encountered, and the solutions adopted in the hardware and software co-verification of a wafer-scale integrated (WSI) system. The WaferBoardTM is designed for rapid prototyping of complex digital systems. This advanced technology embeds a smart active substrate that can be configured through a set of JTAG ports to interconnect digital integrated chips placed on its surface. The software that generates JTAG bit streams and the related hardware infrastructure embedded in the WaferBoardTM must be validated early and efficiently in the design process to improve productivity. The system co-verification methodology presented in this paper relies on a unified SystemC-VHDL environment making the design and verification tasks more effective. Results demonstrate that the simulation time complexity grows quadratically with the circuit size in number of gates if one JTAG port is used.

Index Terms— Co-verification, SystemC, reconfigurable circuit board, Wafer Scale Integration, JTAG, IEEE 1149.1.

I. INTRODUCTION

To meet the requirements of the PCB/multichip module industry, designers must constantly innovate to offer increasing interconnection density and faster interconnect data rate. Moreover, MCM/PCB design time (time-to-market) must constantly decrease in order to stay competitive. This creates a requirement for innovative technologies that increase productivity and performance of products. The need to improve design productivity justifies exploring new ideas and concepts such as the reconfigurable circuit board technology proposed in [1,2], depicted in figure 1. This technology is based on a wafer scale circuit, the WaferICTM.

The WaferICTM sits on a WaferBoardTM. It is used as an active substrate that can transmit digital information between any pins of a set of conventional chips placed by the user (uICs) anywhere on its surface. Moreover, this circuit embeds an array of tiny programmable pads (NanoPads) that can make electrical contact with uIC balls through which required signal connections can be established and power can be supplied to the uICs. The uICs can be CPUs, FPGAs, memories or any other available integrated circuit whose pinout is compatible with the NanoPad array, which is true of most existing ICs. Each Unit Cell includes a programmable crossbar and 4×4 programmable NanoPads (Fig. 1(c)). The interconnection of cell's crossbars creates a global configurable interconnection network.

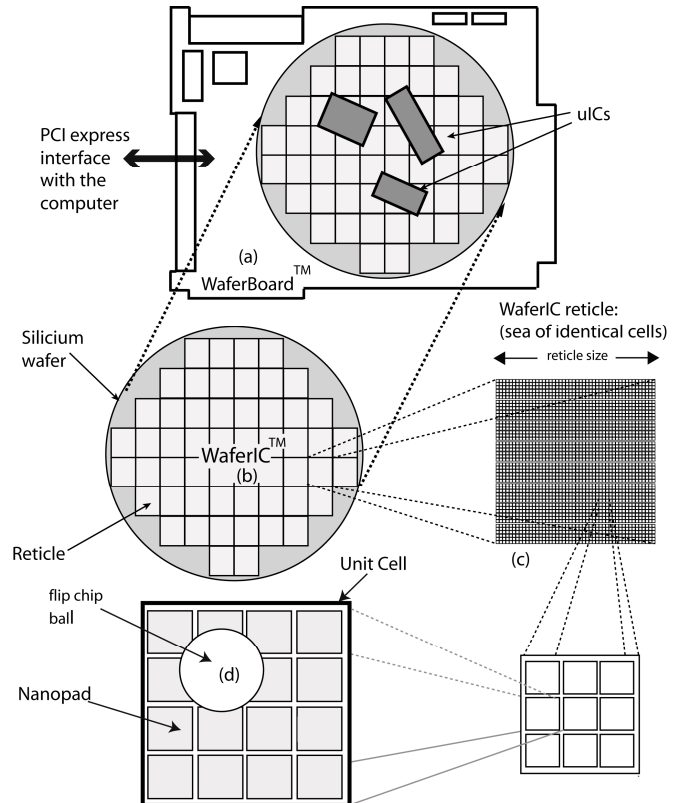


Figure 1: Illustration of the hierarchical structure of WaferBoardTM. (a) WaferBoardTM is connected to a computer via a PCI-express connection. (b) WaferICTM, is a sea of identical cells(c) residing on an external pc-board. The uIC/WaferICTM contact is represented by a ball (d).

This paper presents the hardware and software co-verification strategy used in the design of the WaferICTM. HW/SW co-verification is defined as a methodology to co-simulate hardware and software for functional verification purpose, with two goals: [3]: (1) Debug system software in conjunction with the supporting hardware infrastructure before chips and boards are available. (2) Ensure that the hardware can run the system software. Such co-simulation methodology increases productivity of hardware and software developers and avoids costly hardware design errors. The originality of the work presented in this paper is to adapt and apply a co-verification methodology in a unified SystemC-VHDL verification environment for WaferICTM design.

Four components are integrated in the proposed SystemC-VHDL environment: (1) the interface software generating JTAG bit streams [4] for configuration, test and diagnosis; (2) the mechanisms to validate the algorithms implemented for testing and diagnosing; (3) algorithms for on-the-fly WaferIC™ reconfiguration; (4) an environment to accomplish thorough verification of the RTL features.

Previous works have already demonstrated the feasibility and the efficiency of co-verification methods for various classes of VLSI systems. For example, co-verification of complex SoC circuits that run software [5], or telecom ASIC chips [6] have already been reported, but no prior works focus on mechanisms to validate the JTAG interface for the test, diagnosis and on-the-fly reconfiguration of a wafer-scale reconfigurable board. Those specific features bring a set of particular constraints for which practical solutions are reported in this paper.

The next section summarizes previously reported results on Waferboard™ and WaferIC™. Section III explains the requirements and the internal architecture that a co-verification environment for the WaferIC™ must meet. Section IV details the internal architecture of the SystemC-VHDL environment and section V shows some results from the SystemC-VHDL simulations. The main contributions of this paper are summarized in Section VI.

II. RELATED WORK ON THE ACTIVE RECONFIGURABLE BOARD

WaferIC™ is to PCBs what FPGAs are to logic circuits. Software rapidly configures the WaferIC™ to interconnect several uICs according to user's instructions. In order to support any type of interconnection between uICs, it is necessary to have a reconfigurable network able to support a high density of interconnections between every configurable NanoPads. An example of reconfigurable routing between 2 distinct uICs is shown in figure 2. This is done using a network called WaferNet™ described in reference [2]. A study of its interconnection network presented in [7] focuses on low level characteristics like delays and silicon area. The short-circuit detection or capacitive detection [8] techniques can be used to detect uIC ball contact on NanoPads.

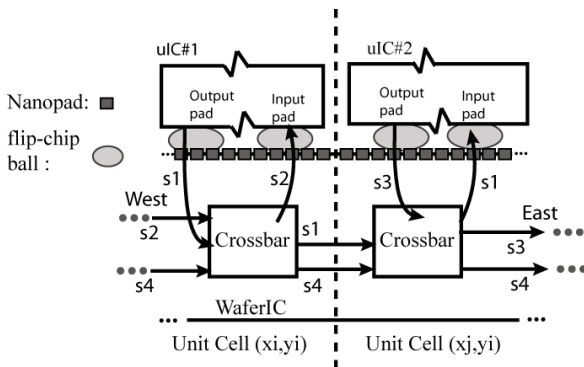


Figure 2: Routing example of a signal s1 from a NanoPad in Unit Cell(xi,yi) redirected to another NanoPad of Unit Cell(xj,yi) towards the east. Signal s4 passes through Unit Cells without interaction with the local uIC.

III. REQUIREMENTS OF THE HARDWARE/SOFTWARE CO-VERIFICATION ENVIRONMENT

The objective of our hardware/software co-verification environment is to make tractable the complex task of verifying the WaferBoard™ system. The main functionalities of the WaferIC™ can be classified in one of the three following successive steps:

A. Step 1: Test and diagnosis

The internal test and the diagnosis of WaferIC™ have three goals. First, it is important to make the circuit testable to detect manufacturing defects. Second, diagnostic mechanisms are essential to locate the faults in WaferIC™, and third, to improve robustness and manufacturability, it is essential to configure the system around those faults. Those essential features for wafer scale system must be efficiently verified.

B. Step 2. Detection procedure for ball/WaferIC™ contacts

The WaferIC™ can interact with uICs deposited on its surface through the hundreds of thousands NanoPads. Those NanoPads form a regular array spaced with a sufficient resolution to exchange I/O data with the smallest uIC packages. To implement a user specified netlist interconnecting uICs, it is mandatory to know exactly which NanoPads are in contact with a uIC ball and the ones that are floating. Therefore, a specific procedure must be developed and validated to detect electrical contacts between NanoPads and uIC balls.

C. Step 3. Netlist upload in the WaferIC™

This step configures the crossbars and NanoPads to create the "netlist" according to user's specifications by sending bitstreams through JTAG interfaces, similar to that available for FPGAs.

All system level functionalities listed above come from the coordinated functionalities contributed by a sea of small identical cells (Fig. 1). These identical cells are logically differentiated after the configuration procedure. To verify and validate the WaferIC™ features, a top level entity that makes connections between Unit Cells is needed. That entity is composed of an array of reticles, and each reticle contains a regular array of cells. It is only through the reticle level or the WaferIC™ level of abstraction that systematic simulation of this regular array of linked cells is possible to verify and validate the complete system that they form. The reticle top level entity can be parametrized with respect to "n", the size in term of Unit Cells in the reticle, and "m" the size of WaferIC™ in term of reticles.

IV. THE SYSTEM CO-VERIFICATION ENVIRONMENT

SystemC is a powerful modelling language often used to control and configure top-level VHDL blocks[9,10]. Such SystemC models can then be seen as testbenches that control a "device under test". This approach is somewhat restrictive, because SystemC can be a much more powerful tool to create a unified co-verification environment.

A. The Software emulation part of the co-verification environment

Our co-verification strategy is applied by suitably controlling the VHDL model of the WaferIC' I/O with

software routines that emulate the behaviour of a “virtual tester”. This “virtual tester” is defined as any software based test and diagnostic procedure applied to the final fabricated WaferIC. Figure 3 illustrate how the software routines are interacting with the RTL model. There are 4 types of I/O in the VHDL model of WaferIC™.

1) *Input JTAG port*: even if an input JTAG port is 4 bits wide (tdi, tms, trst*, tck), it is not trivial to master the integrity of the serial bit stream. A single fault in a very long bit stream jeopardizes the entire command flow. A high level library written in C++ has been developed to generate the JTAG bitstreams.

2) *Output JTAG port (tdo)*: the serial data must be adequately interpreted by C++ routines that extract the data included in the output “tdo” bit stream. Then, they must be interpreted to check the validity of the JTAG emulator and the hardware structure of the WaferIC™.

3) *Data coming from the internal registers*: these data allow verifying the validity of commands sent to the WaferIC™ in the form of JTAG bitstreams. It is possible to precisely define the result expected in the internal registers.

4) *Data stimulation and recording coming from the NanoPads*: the final objective of WaferIC™ is to interconnect the uICs, according to a “netlist” downloaded by a user. The interconnections between the NanoPads can be validated by stimulating and observing the NanoPads behavior.

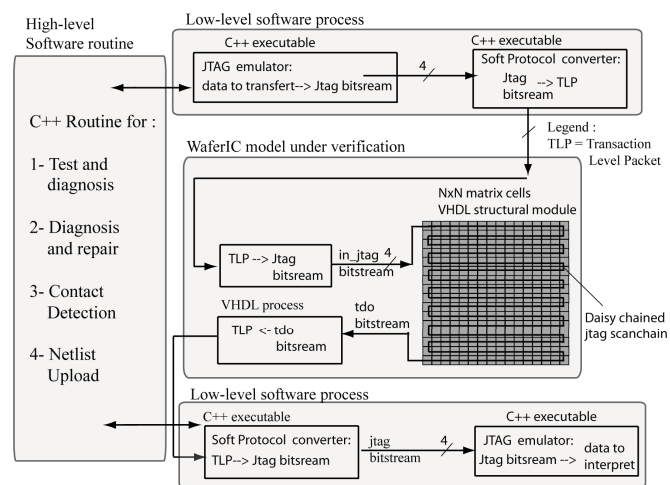


Figure 3: Relationships between the hardware and the software in the WaferIC™ co-verification environment.

Therefore, with the architecture shown in figure 3, developing a testcase for a systematic verification plan is equivalent to directly developing a reusable routine for the WaferIC™ control software. For example, the diagnosis of short-circuit between adjacent NanoPads needs special test algorithms to be validated, which are based on well known short circuit diagnosis algorithms [11]. To validate this kind of feature, one needs to validate the software control and the specific hardware infrastructure. The C++ routines of the control software generate the required bit streams. In addition, the bit streams are applied to the JTAG port through the SystemC-VHDL interface. The auto-verification is completed through software interpretation of the tdo bit

streams. The former software process is re-used in the final control software.

B. Internal Architecture Model in SystemC-VHDL

Figure 4 shows how the various modules of the system co-verification environment interact. A multi-threaded graph (MTG) has been used to describe the environment. MTG is a well known representation for SystemC internal logic dependencies, structures and time dependent interactions [12]. Figure 4 shows the relationships between SystemC processes that emulate the behavior of a “virtual tester” interacting with the WaferIC™.

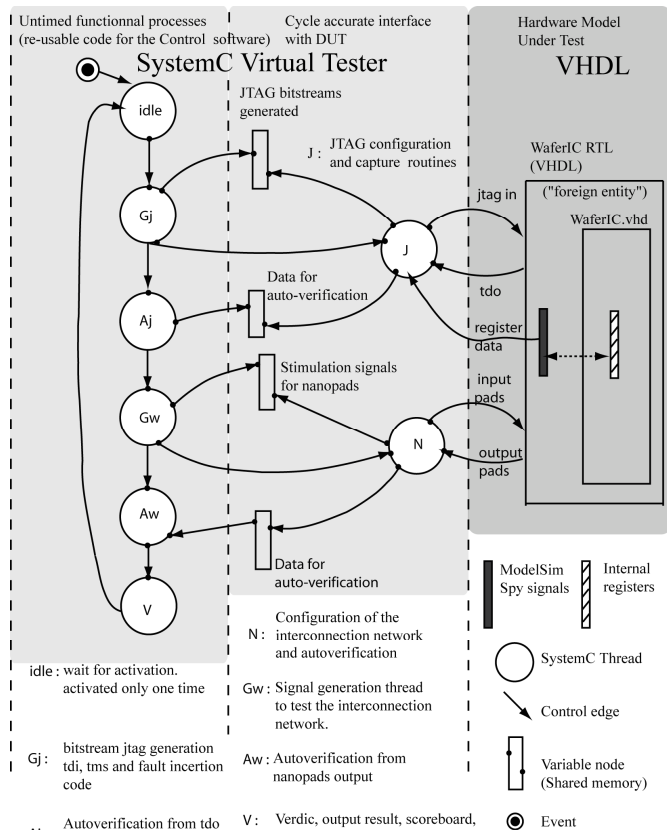


Figure 4: Multi-threaded graph flow for the SystemC-VHDL environment interacting as a virtual tester for the device under test.

The SystemC-VHDL co-verification environment is divided in three specific parts:

1) *The untimed functional processes*: Those processes are associated to the high level software routines called by the co-verification environment. It is not linked to any cycle accurate event, only through high-level logic conditions required by the control of the “virtual tester” that respect the test procedure of the WaferIC™. The same procedures and logic transaction must take place in the future WaferIC™ control software that will be implemented. Thus the routines implemented in the verification environment are truly reusable for the control software.

2) *Cycle accurate processes*: Meanwhile, the “J” and “N” processes (see Figure 4) must deal with the low level time synchronisation between cycles and signals. Those processes

are not reusable. They are placed in the systemC architecture as an interface with the device under verification.

3) *RTL descriptions of the WaferICTM* : Modelsim was leveraged for concurrent SystemC/VHDL simulations. The use of “spy” variables provided by Modelsim made it possible the recopy data from inside to outside modules [13]. Thus, the proposed architecture offers as much data observability for hardware verification, as provided in “e” or System Verilog [14].

V. RESULTS

Our verification environment can simulate all the functional features of the WaferICTM. For demonstration purposes, only a small subset of the entire regression test is shown in this paper. Figure 5 presents the simulation results corresponding to a test case taken from the regression script of the verification plan. This test case uploads JTAG bit streams in each Unit-cell of the WaferICTM in order to configure a specific interconnection net list. All cells are daisy chained together (see fig. 3) as is required by the IEEE 1149.1 communication protocol. Our test cases and RTL descriptions are generic according to the number of Unit Cells in one reticle, n^2 , thus it is easy to change the value of n and observe the effect on memory consumption and simulation time.

As expected, memory used to co-simulate the test cases grows quadratically with n , the size of the Unit Cell array. Meanwhile, the simulation time complexity of one test case grows as $O(n^4)$ as shown in figure 5. Effectively, time complexity depends on both the length of the bit stream and the number of Unit Cells, which each grows as n^2 . A solution to dramatically reduce the simulation time consists of increasing the number of JTAG ports with the number of Unit Cell, in which case the time complexity could remain $O(n^2)$.

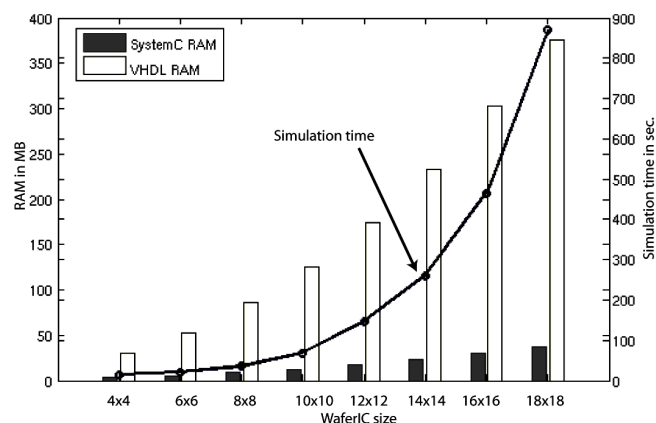


Figure 5: Simulation time and RAM used vs the size of the Reticle for one generic test case.

In spite of such time complexity, our results shows that a typical high-end desktop computer and commercially available software can handle the simulation task. Indeed, the simulations have been carried out on a Linux machine, with ModelSim SE 6.2c. The graph on figure 5 shows that the simulation time and the memory used by the selected typical test case in the verification environment is tractable for a Intel

DualCore running at 2.4 GHz with 3.289 GB RAM desktop computer. Therefore, the reported results demonstrate that our verification environment can handle the required verification tasks to simulate all the features of the WaferICTM.

VI. CONCLUSIONS

An innovative and promising technology for fast system prototyping of digital systems that uses Wafer-Scale Integration has been presented and validated with a model coded in SystemC-VHDL and the ModelSim simulator. Moreover, the Unit-Cell that is the cornerstone of the proposed regular architecture and its associated configurable interconnection network have been simulated, validated and verified with an unified co-verification approach. The successful simulation reported in this paper demonstrates that the level of complexity of the proposed hardware/software co-verification environment is manageable for a small team of designers.

ACKNOWLEDGMENTS

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada and Gestion TechnoCap Inc. for their financial support and CMC Microsystems for providing design tools, support and associated technologies.

REFERENCES

- [1] R. Norman, O. Valorge, Y. Blaquièrre, E. Lepercq, Y. Basile-Bellavance, Y. El-Alaoui, R. Prytula, Y. Savaria, “An Active Reconfigurable Circuit Board”, NEWCAS 2008, Montreal, Canada, June 2008.
- [2] R. Norman, U.S. Patent Application Number 11/611,263.
- [3] A. Ghosh, M. Bershteyn, R. Casley, C. Chien, A. Jain, M. Lipsie, D. Tarrodaychik, O. Yamamoto, “A hardware-software co-simulator for embedded system design and debugging”, Asia and South Pacific Design Automation Conference, Chiba, Japan, Sept 1995.
- [4] P. K. Parker, “The boundary-scan handbook: analog and digital”, 2nd ed., Kluwer Academic Publishers Boston/ Dordrecht/ London, 1998, ch. 1.4.
- [5] Y. Nakamura, “Software Verification for System on a Chip using a C/C++ Simulator and FPGA Emulator”, International Symposium on VLSI Design, Automation and Test, April 2006.
- [6] T. Wu, P. Wang, D. Jin, L. Zeng, “Hardware/software co-verification scheme for MSTP ASIC”, ICSICT '06. 8th International Conference on Solid-State and Integrated Circuit Technology, 2006.
- [7] E. Lepercq, Y. Savaria, Y. Blaquièrre, “An Interconnection Network for A Novel Reconfigurable Circuit Board”, NEWCAS, Montreal, Canada, June 2008.
- [8] Y. Blaquièrre, Y. Savaria, J. El Fouladi, “Digital Measurement Technique for Capacitance Variation Detection on Integrated Circuit I/Os”, ICECS'07, December 11-14, 2007.
- [9] SystemC Version 2 User's Guide, <http://www.systemc.org>
- [10] Functional Specification for SystemC 2.0.1- Version 2.0-P, <http://www.systemc.org>
- [11] A. Hassan, J. Rajski, V.K. Agarwal, “Testing and Diagnosis of Interconnects using Boundary Scan Architecture”, International Test Conference, 1988
- [12] F. Thoent, F. Catthoor, “Modeling, Verification and Exploration of Task-Level Concurrency in Real-Time Embedded Systems”. Kluwer Academic Publishers. 438p., 2000.
- [13] Mentor Graphics, “ModelSim SE User's Manual”, August 2006, ch. 21.
- [14] J. Bergeron, “Writing Testbenches: Functional Verification of HDL Models”, Springer, 2nd edition, 2003, ch 4.5.