A SOFTWARE PROGRAMMABLE INFRASTRUCTURE FOR ASSERTION BASED POST SILICON DEBUG

Abner Luís Panho Marciano*, José Augusto Miranda Nacif†*,
Celina Gomes do Val*, Antônio Otávio Fernandes*, Claudionor Nunes Coelho Jr.*
*Computer Science Department, Universidade Federal de Minas Gerais**
Universidade Federal de Viçosa, Florestal Campus†
{*alpm,jnacif,celina,otavio,coelho*}*@dcc.ufmg.br*

ABSTRACT

Even with simulation and formal verification techniques, some bugs reach the silicon, thus, nowadays post-silicon debug techniques are vital for the fabrication of reliable integrated circuits. One approach used is the assertion-based design, in which an assertion processor (AP) can be employed to monitor synthesized property checkers. To reduce area overhead and provide more flexibility through the project stages, in this paper an external AP is implemented using a softcore processor and embedded software. A case study of the architecture using a MIPS32 based processor is also discussed. The area of the synthesized processor with the structure needed to connect itself to the external AP presented an increased of only 0.5%*. Furthermore, the implementation costs of the proposed architecture are analysed.*

1 INTRODUCTION

Verifying integrated circuit designs has always been a growing challenge. If we consider a circuit with n inputs and m possible states, the verification has a complexity of 2^{m+n}. According to Moore's law, every 18 months the density of the existing circuits doubles, what would, in the best case scenario, duplicate the problem's complexity.

As current designs complexity grows, the time needed in the design verification stage increases [1]. Verifying high-complexity modern industrial designs is a challenging task because it is impractical to simulate all possible configurations of the chip. Therefore, the impact on time-to-market and economical issues are significant, making necessary to pursue new techniques to overcome these limitations. Still, these methods can miss some corner cases [2], which may pass unnoticed through manufacturing tests, and lead to bugs in field, such as the famous Intel's Pentium P5 FDIV bug [3].

In order to certify that a design will behave as expected, pre-silicon verification and debug techniques are employed. However, these techniques, which include formal verification and simulation, are inadequate to ensure that the first silicon will be error free. Thus, post-silicon debug techniques [2] are employed to help in the identification of the cause of these failures.

The inclusion of assertion monitors in the designs [1, 4, 5] is a well-known strategy to carry on the debugging process. By adopting this approach, designers are able to verify if certain constraints are met by a selected logic block. Thus, when a certain assertion fails it is possible to trace its origin and point out what is the exact cause of the error.

Extracting data from these assertions can be tricky. To provide a mean to manage and access the information from the assertions, an Assertion Processor (AP) module is commonly attached and synthesized with the design [6]. This approach imposes an area overhead and is not flexible, as once the Assertion Processor is synthesized it is not possible to modify its behavior. In order to address these issues, we propose to move the AP to an external IP, called the AP module (APM).

To make the design process more flexible, we propose a framework that can be used in both prototyping and post-silicon stages. On the first scenario, during prototyping stage, the designer can first check if the circuit under debug (CUD) behaves as expected using a FPGA. On the second scenario, post-silicon, the CUD and the assertions are manufactured in silicon and the APM is implemented in a FPGA. On both approaches we use the same interface between CUD and APM.

Our architecture uses simple assertions from OVL (Open Verification Library) [7], to implement a scan-chain design that is controlled by an external AP that is programmed by software.

This paper is organized as follows: Section 2 shows related work; Section 3 presents the methodology and the architecture developed; Section 4 shows the results; Section 5 presents conclusion and future work.

2 RELATED WORK

Debugging silicon requires the analysis of the chip's internal state. This is a challenging task due to the lack of observability [8], caused by the limited number of pins available to internal signal visualization. Observability can be enhanced by adopting one of two common techniques:

Real-time observability and Time-intrusive, scan-based observability [9].

Time-intrusive observability can be reached by reusing test structures like JTAG and scan chains commonly present in industrial designs. By using these structures, engineers can debug the circuit by halting the system and carrying out a set of signals. This is the main idea behind the IEEE 1149.1 standard [10], where boundary-scan registers are used to store signals while the data is extracted through a test access port (TAP). On real-time approach, internal signals are monitored and captured in real time. To achieve this observability, a trace-based debug architecture is generally used. This debug method consists on storing the state of a selected set of signals on a memory, called trace-buffer, while the design is running at full speed. Later the data stored on this buffer can be serially extracted from the chip.

Current design flows usually rely on assertion-based and property checking methodologies, as stated by H. Foster in [5]. In these methodologies, designers include assertions that describe design's properties that must hold during simulation or be proved by formal verification techniques. If one of these assertions fails, an error between designer's intention and implementation is found.

In order to implement these techniques, the Open Verification Library (OVL) provides a rich set of Verilog and VHDL assertion modules, called monitors. The Open Verification Library (OVL), a vendor- and language-independent assertion library, is a powerful tool on the design process. It enables designers to elaborate the assertion specification once, and then use it on multiple verification processes, such as simulations and formal verification tools.

On a simple approach, a dedicated connection to the monitor would be needed, what in large scale systems could result in unnecessary area overhead. A solution for this problem was presented by [6], where OVL modules were combined with the IEEE 1149.1 standard, creating a scan-based architecture that can be used in silicon and or to emulate the design in an FPGA, at full clock speed.

3 METHODOLOGY

As presented in [5], the monitors provided by OVL are modules that checks invariant properties. It supplies simple monitors such as the *assert_never*, that reports a failure when its target must never evaluate TRUE, to more fine grained ones, as the *assert_win_change*, that verifies a value change on an event window.

However, the OVL modules can just be used on the simulation/emulation phases of the design, as it is not synthesizable. To surpass this problem, [6] presented modified version of the OVL so it could be arranged on a scan-chain approach. In order to modify this structure, extra pins were needed, three inputs and two inputs. Figure 1 depicts an standard OVL module and its modified version. Table 1 describes the new module's pins.

This work presents a silicon debug framework composed by a Wrapper Interface (WI) and an external monitoring system. The proposed architecture is shown in Figure 2.

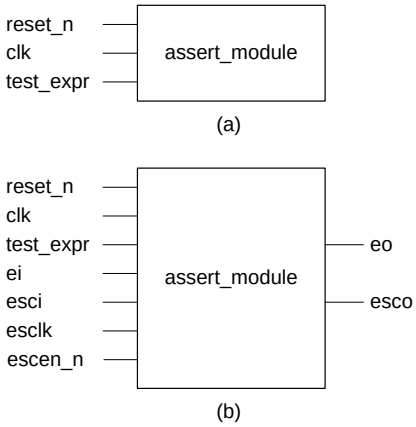

Figure 1. (a) Typical OVL assertion; (b) OVL assertion modified for scan-chain architecture.

Table 1. Signal descriptions for Figure 1(b)

Signal	Description	IO
reset_n	Reset Active Low	Input
clk	System Clock	Input
test_expr	Any HDL test expression	Input
ei	Error Input	Input
esci	Error Scan Input	Input
eo	Error Output	Output
esco	Error Scan Output	Output
esclk	Error Clock	Input
escen_n	Error Scan Enable Active Low	Input

The WI module wraps the circuit under debug (CUD). On WI's internal interface, are connected the assertion scan-chains and, optionally, manage signals like clock, reset and halt, providing more controllability. Also, WI offers access to shared memories on the IC, that can be used as a mean to inject instructions and fetch general data. Outside, WI provides a simple and narrow bus which will interface with the external monitor.

On the external monitoring system, present on a FPGA, lies a Xilinx MicroBlaze [11] softcore processor, which is connected to a Processor Local Bus (PLB) [12] as a master and communicates to its slaves. A slave PLB compliant interface was created to interface with the bus from the CUD. Furthermore, an UART module, provided by Xilinx, was also attached to the PLB as a slave, to provide an interface with an embedded software which runs on the softcore. This software, called GreenBug, supplies a simple interface where engineers can easily control the processor and monitor the assertions. GreenBug also provides access to the IC's memory, making possible to pass and retrieve data on a debug basis.

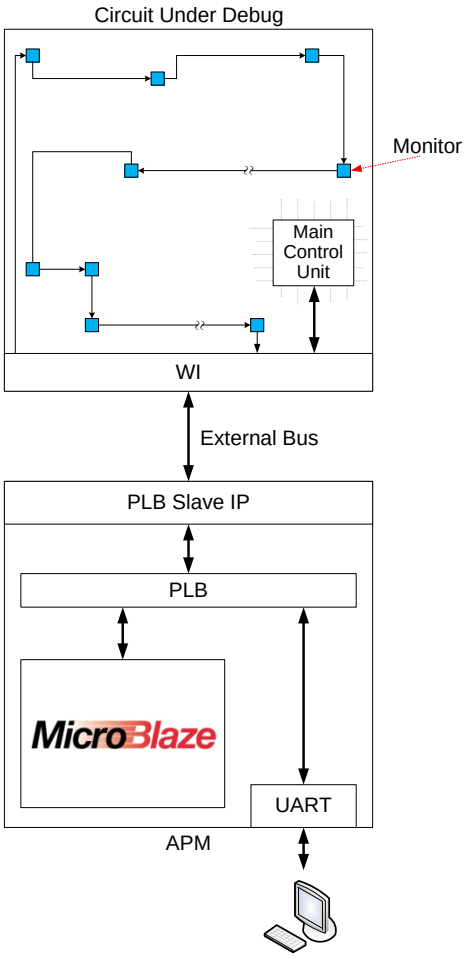

Figure 2. Proposed Architecture

4 CASE STUDY

To quantitatively demonstrate the benefits of the proposed architecture, a case study of the presented architecture was done. A single core MIPS32 processor, implemented in Verilog, was used, and relevant features like 5 stage pipe-line, instruction and data caches employed. We chose this 32-bit RISC architecture due to its extensively use in embedded solutions such as routers, gateways, some SGI computers and video-game consoles [13].

To assess the debugging process, 45 monitors were chained across the processor modules. Table 2 describes the types of assertions used.

The MIPS core was wrapped within the WI, also providing access to its central control unit and its shared memory unit. Thus, the design would be ready for post-silicon and prototyping stages.

When an assertion fails, an interruption is dispatched by the APM PLB Slave module, and the softcore enter the dump mode. Once dump mode is reached, the processor is stalled, and GreenBug automatically starts dumping all assertion data from the target's scan-chain in a non-destructive manner. It is also possible to enter dump mode through a command on GreenBug.

Table 2. OVL-based monitors used

Assertion	Ensures
assert_next	Proper cycle timing between two events
assert_implication	A specified consequent expression is TRUE if the specified antecedent expression is TRUE
assert_never	Property must never evaluate true
assert_range	Legal value within a valid range
assert_change	Value change on an event window
assert_unchange	Value must not change on an event window
assert_zero_one_hot	Value of a specified expression is zero or one-hot

4.1 Results

The project was synthesized using the Xilinx ISE Embedded Edition 12.2 [14] to Xilinx Virtex5 XC5VLX110T. Virtex 5 is a series of high-performance logic with advanced serial connectivity FPGA's. XC5VLX110T FPGA provides 110.000 gates.

We compare two approaches: a prototyping FPGA based one, where all the design is contained in a single FPGA (APM+MIPS+WI); a first-silicon, with the circuit under debug in silicon and the APM in a separated FPGA. Table 3 summarizes the synthesis results. The obtained outcome evidences the APM overall low cost in both stages, making possible the use of far more simpler FPGAs.

Table 3. Synthesis results for APM

Logic Utilization	Prototyping[1]		First-Silicon[2]	
	Used	Total	Used	Total
Number of Slices	3174	18%	1675	9%
Number of Slice Registers	4793	6%	2983	4%
Number of Slice LUTS	5936	8%	2833	4%

[1] APM+MIPS+WI
[2] APM only

Table 4 compares the MIPS with and without the WI. Therefore, its possible to infer WI's cost, which raised in just 0.5% in an overall analysis.

Table 4. Synthesis results for MIPS

Logic Utilization	With WI		Without WI	
	Used	Total	Used	Total
Number of Slices	1669	9%	1583	9%
Number of Slice Registers	1909	2%	1891	2%
Number of Slice LUTS	3498	5%	3274	4%

5 CONCLUSION AND FUTURE WORK

In this paper we designed and implemented an external assertion processor as well as a software based infrastructure to aid post silicon debug. We discussed the assertion-based and property checking based debugging methodolo-

gies. Further, a case study of the proposed architecture using a MIPS32 based processor was done. The synthesis results showed promissing results, with minor area occupation ($\sim 9\%$) by the AP on a FPGA system, and insignificant area overhead after wrapping the processor core with the module used to interface with the AP.

For future work we expect to improve the architecture proposed, in order to implement a fully reconfigurable assertion architecture, where the engineer can disable or enable sets of assertions on a design using software.

6. REFERENCES

[1] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for socs," in *Proceedings of the 43rd annual Design Automation Conference*, ser. DAC '06. New York, NY, USA: ACM, 2006, pp. 7–12. [Online]. Available: http://doi.acm.org/10.1145/1146909.1146916

[2] M. Abramovici, "In-system silicon validation and debug," *Design Test of Computers, IEEE*, vol. 25, no. 3, pp. 216 –223, may-june 2008.

[3] M. L. M. Subramaniam, "Abstraction for analytic verification of concurrent software systems," in *Symposium on Abstraction, Reformulation and Approximation*, ser. SARA '98, 1998, pp. 85–94.

[4] A. Gharehbaghi, M. Babagoli, and S. Hessabi, "Assertion-based debug infrastructure for soc designs," in *Microelectronics, 2007. ICM 2007. Internatonal Conference on*, dec. 2007, pp. 137 –140.

[5] H. Foster, D. Lacey, and A. Krolnik, *Assertion-Based Design*, 2nd ed. Norwell, MA, USA: Kluwer Academic Publishers, 2003.

[6] J. A. M. Nacif, F. M. de Paula, H. Foster, C. J. N. C. Jr., and A. O. Fernandes, "The chip is ready. am i done? on-chip verification using assertion processors." in *International Conference on Very Large Scale Integration of System-on-Chip (VLSI-SoC)*, 2003, pp. 111–116.

[7] Accellera, "Open verification library technical subcommittee," Available at: http://www.accellera.org/activities/ovl/, 2011.

[8] E. Hung and S. J. E. Wilton, "On evaluating signal selection algorithms for post-silicon debug," in *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, march 2011, pp. 1 –7.

[9] B. Vermeulen and S. Goel, "Design for debug: catching design errors in digital chips," *Design Test of Computers, IEEE*, vol. 19, no. 3, pp. 35 –43, may/jun 2002.

[10] "Supplement to ieee std 1149.1-1990, ieee standard test access port and boundary-scan architecture," *IEEE Std 1149.1b-1994*, p. i, 1995.

[11] Xilinx, "Microblaze processor reference guide (embedded development kit edk 13.1)," Available at: http://www.xilinx.com/support/documentation/, 2011.

[12] ——, "Logicore ip processor local bus (plb) v4.6 (v1.05a)," Available at: http://www.xilinx.com/support/documentation/, 2011.

[13] M. T. Inc., "Mips technologies website," Available at: http://www.mips.com, 2011.

[14] Xilinx, "Embedded development kit edk 12.2," Available at: http://www.xilinx.com/products/design-tools/ise-design-suite, 2011.