IMTC 2005 – Instrumentation and Measurement
Technology Conference
Ottawa, Canada, 17-19 May 2005

Developing Test Environment for Embedded Cores-Based System-on-a-Chip (SOC)

Sunil R. Das[1,2], Dhruv Biswas[1], Emil M. Petriu[1], Mansour. H. Assaf[1] and Mehmet Sahinoglu[2]
[1]School of Information Technology and Engineering, Faculty of Engineering,
University of Ottawa, Ottawa, ON K1N 6N5, Canada
[2]Department of Computer and Information Science,
Troy State University Montgomery, Montgomery, AL 36103, USA

Abstract – *The subject paper proposes developing test environment and test methodologies for digital embedded cores-based system-on-a-chip (SOC). The digital cores used in the study were constructed from ISCAS 85 combinational and ISCAS 89 sequential benchmark circuits. The wrapper that separates the core under test from other cores is assumed to be IEEE P1500-compliant. The test access mechanism plays an important role in transporting the test patterns to the desired core and the core responses to the output pin of the SOC. The faults were injected using a fault simulator that generates tests for the core. The cores and test access mechanism were described using Verilog HDL. The test access mechanism (TAM) provides the connection between the test sources, cores, and test sinks, and is crucial in any SOC design. There are many ways to design a TAM. In this paper, TAM was implemented as a plain signal transport medium, which is shared by all the cores in the system-on-a-chip. Once the compilation of the cores was successful, the fault simulation was carried out. The selection of the individual cores was taken care of by the program running in the background. The outcome was the fault coverage of all the cores being tested.*

Keywords – *Altera Max Plus II, built-in self-testing (BIST), embedded cores-based system-on-a-chip (SOC), sequential circuits, test access mechanism (TAM), test pattern generator (TPG), Verilog HDL, wrapper.*

I. INTRODUCTION

Large-scale integration has added enormous complexity to the process of testing modern digital circuits. Besides, during the past several years, integrated circuit technology evolved from chip-set philosophy to an embedded cores-based system-on-a-chip (SOC) concept, which simply refers to *an IC, designed by stitching together multiple stand-alone VLSI designs to provide full functionality for an application* [1]. Though many aspects of these embedded cores-based systems and SOC are still evolving, they are revolutionizing the electronics industry. These innovations are already on their way to the next generation of cell phones, multimedia devices, and PC graphics chipsets. The cores-based design, justified by the necessity to decrease time-to-market, has created a host of challenges for the design and test community. Unlike traditional test design approaches, SOC makes it impossible to establish the demarcation lines between design and test. For mixed-signal devices and complex digital cores, engineers must use design tools that let them incorporate testability early in the design process. Though the use of cores in an SOC serves a wide range of applications, they are far too complex to be tested by traditional methods. In practice, combinations of test methodologies such as functional test, full-scan test, logic BIST, RAM BIST, Iddq, etc. are normally used for SOCs [1]–[19]. But difficulties surround implementation of the methods for testing cores and SOCs as a whole. A wide range of circuits traditionally comes within the definition of SOCs including microprocessors and microcontrollers. The core test integration is a complex problem – the chip integrator can modify the test and add some design for test (DFT) and built-in self-test (BIST) features, if necessary. Specifically, in the context of embedded cores-based system testing, electrical isolation involving the input and output ports of the core from the chip or other cores is a necessity. The fundamental items of interest in core test is access, control, and isolation, and these are the issues which were addressed by the IEEE Technical Council on Test Technology Working Group P1500 entrusted with the responsibility of developing standard architecture for their solution [1], [3]. The embedded core test requires, in general, hardware components like wrapper around the core, a source and a sink for test patterns (on-chip or off-chip) and an on-chip test access mechanism (TAM) to connect the wrapper to the source or sink. The cores could be without boundary scan or with boundary scan. For design and test reuse, ASIC manufactures have suggested certain characteristics. In general, different DFT and BIST schemes like scan, partial scan, logic BIST and scan-based BIST are used to test various logic blocks within an SOC like microprocessor or microcontroller. However, the main problem is still the resulting area overhead and performance penalties. Structural test methods like scan and BIST are desirable for test reuse, portability, and test integration into the SOC test set [1], [3]. The TAM includes on-chip test generation logic for cores with BIST. The DFT techniques involve adding optimized test logic within cores and at the chip level to enhance testability, and DFT logic helps test pattern generation and application, and assist in the support test environment [2].

In this paper, test methodologies are proposed for embedded cores-based system-on-a-chip (SOC) digital systems comprising of wrapper and test access mechanism (TAM). The cores considered in the study were constructed from ISCAS 85 combinational and ISCAS 89 sequential benchmark circuits. The fault model

0-7803-8879-8/05/$20.00 ©2005 IEEE

used is the conventional single stuck-fault model. The wrapper separates the core under test from other cores, which is assumed to be IEEE P1500-compliant. The test access mechanism plays a vital role in transporting the test patterns to the desired core and the core responses to the output pin of the SOC. The faults were injected using a fault simulator that also generates the test sets for the cores. The test access mechanism (TAM) was implemented as a plain signal transport medium, which is shared by all the cores in the SOC. Once the compilation of the cores was done, the fault simulation was carried out with the test patters feeding the cores through the TAM. The selection of the appropriate core is taken care of by the program running in the background. The automatic fault simulator generates tests for the core described at the gate and flip-flop level. There are very few constraints on the core being tested. The simulation process is completely automatic, and requires no intervention from the designer during the test generation process. The subject paper describes the architectures of the wrapper and test access mechanism, applications of the fault simulator, together with models of the SOCs being used, based on application of Altera Max Plus II development environment. Some partial simulation results using the designed cores are also provided to demonstrate the feasibility of the proposed implementations.

II. TEST ACCESS MECHANISM (TAM) AND WRAPPER

The design of test access mechanism (TAM) and test wrapper is of critical importance in terms of system integration since they directly impact hardware overhead, test time, and tester data volume. The main issues in this context are wrapper optimization, core assignment to TAM wires, sizing of the TAMs, and TAM wire routing.

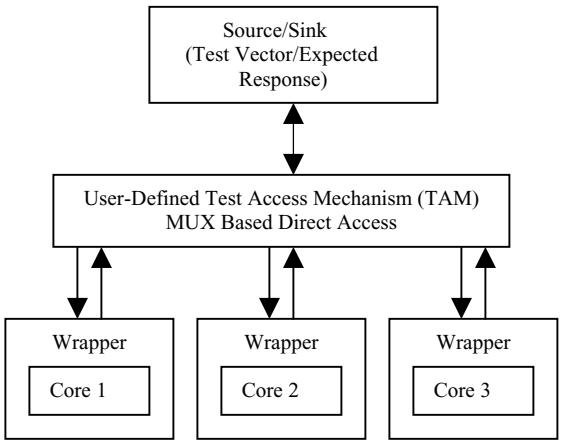

Fig. 1. A multiplexer (MUX)-based direct accesss TAM architecture.

There are two important concepts related to TAM, viz. test pattern source and sink, and core wrapper. The test pattern source is responsible for generating the test vectors or test stimuli for the desired core under test. The test pattern sink compares the fault-free response to the faulty response of the core under test. The test pattern source and sink can be built on-chip or off-chip. In our case, we implemented this by using the fault simulator that generates the test vectors, and after getting the test response, compares it with the fault-free response. We will briefly discuss about fault simulation later. There are several ways to design and implement a TAM. Some of the common TAM architectures are: 1) daisy chained TAMs (that use serial shifting of test data); 2) bussed TAMs (based on use of complex protocols); 3) direct access TAMs (for cores with many inputs and outputs); and 4) multiplexer (MUX)-based direct access TAMs. In this paper, we implemented MUX-based direct access TAM architecture, as shown in Fig. 1. The TAM is used to drive the test vectors from the test source, that is, from the fault simulator to the desired core under test and to transport the test response from the core back to the fault simulator. The selection of the core in the SOC was implemented as part of the TAM architecture. We determined the width of the TAM by the core that has the maximum number of input/output (I/O) pins within the SOC. There are other issues for consideration at this phase, viz. the bandwidth of the TAM versus the cost of extra wires needed for its implementation, total test time depending on the TAM bandwidth, test vectors from the source, and ultimately test data for the individual core. The obvious mechanism to make embedded cores testable from the IC pins is to make the core under test directly accessible from the IC inputs. Though this approach is mostly practiced for embedded memory cores, many block-based ASICs also use this test strategy.

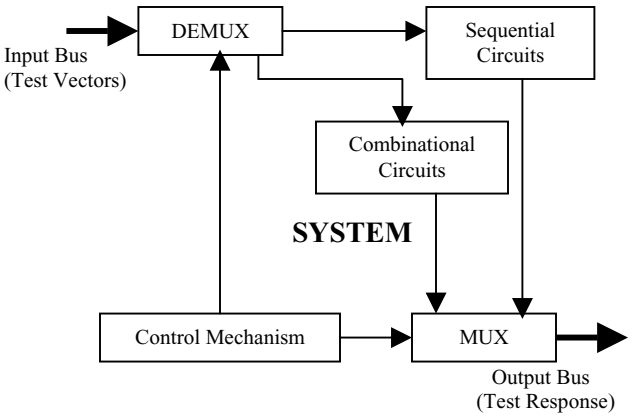

Fig. 2. Proposed SOC test architecture.

The demultiplexer (DEMUX), multiplexer (MUX) and control mechanism are coded such that they provide a dedicated test mode for each core in an SOC. The control signal connects the input bus to the specific core, and the test vectors are passed on to the core through the bus. Once

the fault simulation is done, the test response is channeled through the output bus. After all the cores in the SOC were

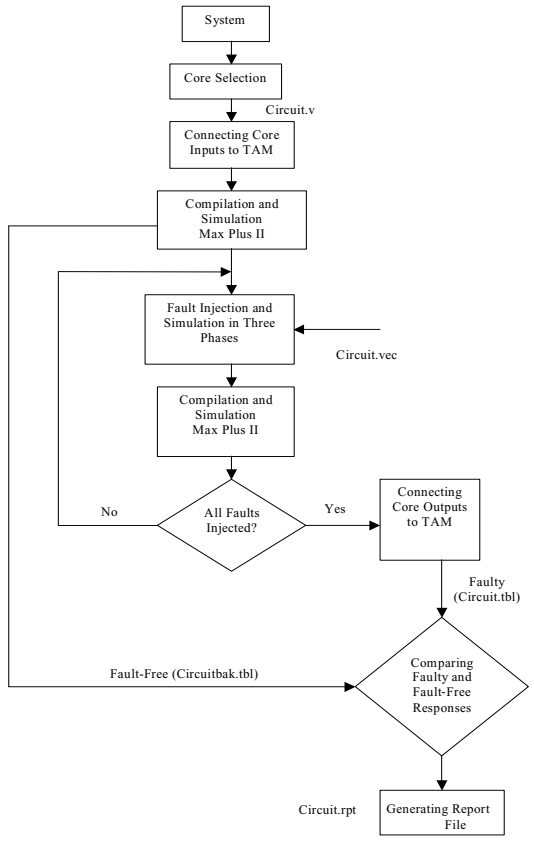

Fig. 3. Flow chart of SOC test.

tested, the process was repeated for other SOCs. The width of the input (and output) bus is determined by the maximum number of input (and output) pins contained in a core out of all the SOCs. The input and output bus uses only the bits required by the core being tested at a time, the rest of them being set to high impedance (Z). The advantage of this approach is clear; an embedded core is tested as if it were a stand-alone device. Iyengar and Chakrabarty [2] proposed methods for determining the TAM width. They developed a new wrapper/TAM co-optimization methodology that have addressed TAM optimization and wrapper design as independent problems and overcomes the limitations of previous TAM design approaches. The design approach they presented improves upon previous approaches by minimizing core test time, and reducing the TAM width required for the core. They also considered the conflict that appears while sharing the test bus for test data transport. The advanced microcontroller bus architecture (AMBA) is another approach where all test sets are scheduled in a sequence on a single bus. Varma and Bhatia [13] also proposed a bus architecture for TAM. In multiplexing architecture, the tests are performed in sequence. In a distributed architecture, all test are done concurrently on their dedicated TAMs, while daisy-chain architecture works like a distributed architecture but uses an additional clocked buffer. Another important TAM architecture proposed is the scalable bus-based architecture called Test-Rail. During recent times, integrated TAM design and test scheduling has also been attempted. We attempted to implement the bus-based TAM that is flexible and scalable and is hence worth exploring. However, there were problems related to optimization of test time under our proposed architecture.

The wrapper is a thin shell around the core, which allows the core to be tested as a stand-alone entity by shielding it off from its environment. Also, the wrapper allows the environment (glue logic and bus architecture) to be tested separately without the core's intervention. The P1500 Scalable Task Force defines the behavior of a standard wrapper to be used with every core to be P1500-compliant and as an interface between the wrapper and test access mechanism. The objective of the wrapper is to facilitate core test, interconnect test, and isolate functions by providing switching between different test and diagnostic modes as well as the normal functional mode. The wrapper of a core may interfere with three types of input/output signals: 1) static control signals for wrapper modes; 2) digital data and dynamic control signals (to be routed through wrapper cells); and 3) exceptional signals (to bypass wrapper cells), such as clocks, high-speed, asynchronous, analog, and other special-purpose signals.

The standard wrapper behavior can be implemented and provided by the core vendors, or it can be added to the core at the design stage of an SOC. The interface between the wrapper and test access mechanism is also standardized, but a specific test access mechanism is not defined for SOC since the implementation depends on the test strategy used by the SOC designer. With the design, simulation, and test vector files of intended functionality of the core, it is the core provider's responsibility to perform the following: 1) augment the functionality of the wrapper into the core; and 2) provide an RTL simulation model of its behavior, or provide a file of an RTL, or gate level design with synthesis scripts and a text description of its operation, which if implemented with the core design according to the text description, will perform the P1500 functions. The P1500 functions ensure the following:

- An isolation such that during the core test mode any activity in the core and wrapper will not adversely affect any other section of the chip.
- Access to the core design to use the testability features, test vector application, and test response observation.
- Access to interconnects and user-defined logic (UDL) between the cores, or between a core and primary I/Os, such that any testing of interconnects and UDL will not adversely affect the core circuitry.

The IEEE Working Group P1500 has developed two standards: *IEEE 1500-Compliant Core* and *IEEE 1500-Ready Core.* The *IEEE 1500-Compliant Core* refers to a core that incorporates an IEEE 1500 wrapper function, and comes with a CTL program. The *IEEE 1500-Ready Core* refers to a core that does not have a complete IEEE 1500 wrapper but does have an IEEE 1500 CTL program on the basis of which the core could be made 1500-compliant. It was assumed that the combinational and sequential circuits we used as cores in our SOCs were IEEE 1500-complaint cores.

III. CORE TEST GENERATION AND TEST ENVIRONMENT

The cores we used for our purpose were constructed from ISCAS 85 combinational and ISCAS 89 sequential benchmark circuits. The problems related to testing sequential circuits are much more complicated than those for the combinational circuits. The basic reason is the presence of storage elements in the realization of sequential circuits. The testing time is also generally higher in the case of sequential circuits. The test generation was focused at the gate level based on single stuck-fault model. The first step in our SOC test generation procedures is to obtain specified sets of test vectors using high-level descriptions of circuit cores. We endeavored to improve upon the process by using automatic test generation for high fault coverage. Figure 2 gives our proposed SOC test architecture, while a flow chart of the test procedure is furnished in Fig. 3.

All the SOCs used in the paper were designed to operate in Altera Max Plus II development environment. Not only the SOC cores, the test access mechanism (TAM) was also designed to work with Max Plus II and described in Verilog HDL. The process of fault injection for the different SOC cores was carried out by the designed fault simulator [11]. The fault simulator was written in C programming language, and injects faults in three distinct steps in a core. It first injects faults at the input lines, then at the output lines, which were followed by fault injection at the internal wires. The nature of faults is single stuck-faults. Thus each line can have only two types of stuck-faults: stuck-at-1 and stuck-at-0. A line with a stuck-at-1 fault will always have a logical value 1 irrespective of the correct logical output of the gate driving it. We used here simulation based fault injection scheme to evaluate the dependability of the system based on the percentage of fault coverage. Each line is injected either with a stuck-at-1 or with a stuck-at-0 fault. The fault simulator makes a fault list of all stuck-faults on the inputs and outputs of gates and functional blocks. The simulator then simulates the circuit core with the selected faults by using the given test vectors. It keeps a record of the detection data (time and output of detection) for each fault and also stores the true-value response. Each vector set is run through the simulator by using the list of undetected faults up to that point. At the end of a run, the simulator can store the internal states of the circuit core in a checkpoint data set for use by the next vector set. This data set is provided to the test program and the fault coverage is finally calculated.

IV. IMPLEMENTATION RESULTS

To demonstrate the feasibility of our developed test environment and test methodologies for embedded cores-based SOCs, independent simulations were conducted on the cores of the various SOCs that we constructed based on ISCAS 85 combinational and ISCAS 89 sequential benchmark circuits. We first provide here some simulation data on an SOC that was constructed from the sequential circuits in ISCAS 89 benchmark circuits list. Next, simulation experience on an SOC comprised of the combinational circuits in ISCAS 85 benchmark circuits list is given. Finally, we consider SOCs comprised of both sequential and combinational circuits from ISCAS 89 and ISCAS 85 benchmark circuits lists, respectively. In this category, we experimented with four mixed SOCs. All the simulation results were obtained using C programming language on an IBM-compliant personal computer with AMD Athlon (TM) processor having 256 MB of RAM. In our implementations, we used system function of C to call Altera Max Plus II for compilation and simulation of each core in our SOCs. The individual cores were described using Verilog HDL gate-level descriptions.

Figure 4 shows the plot (bar chart) of the number of injected faults, detected faults, and fault coverage for our first SOC (SOC_seq) using pseudorandom test vectors. In Fig. 5 is shown the graph of the number of test vectors, testing time, and fault coverage for our second SOC (SOC_comb), also using pseudorandom test vectors. Figure 6, on the other hand, shows a plot of the number of test vectors, testing time, and fault coverage in combinational circuit SOC (SOC_comb) using deterministic compacted test sets (MinTest) [20], [21].

Tables 1−4 show results for fault coverage, number of injected faults, test vectors, and detected faults of the four mixed SOCs: mixed_SOC_1, mixed_SOC_2, mixed_SOC_3, and mixed_SOC_4, respectively. The experimental results, though not comprehensive enough, give some idea on the performance of the proposed implementations in terms of TAM realization and core test methods in general. A more detailed experimental data, either in graphical or in tabular form, however, could not be included due to space constraints.

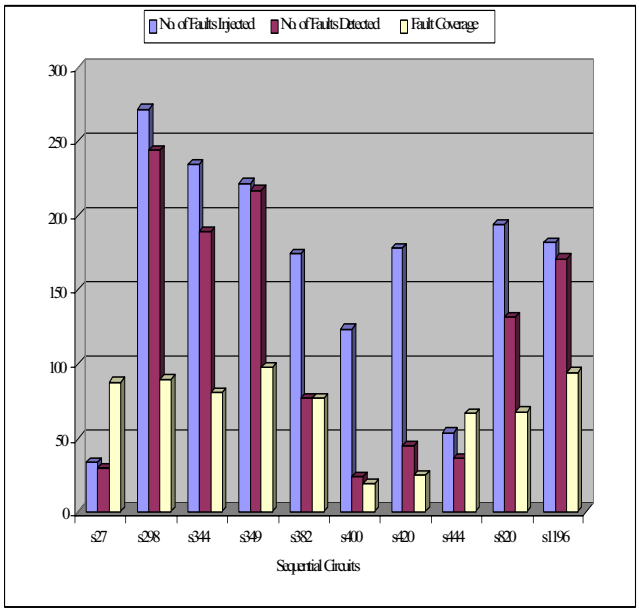

Fig. 4. Faults injected, detected, and fault coverage
in sequential circuit SOC (SOC_seq)
using pseudorandom test vectors.

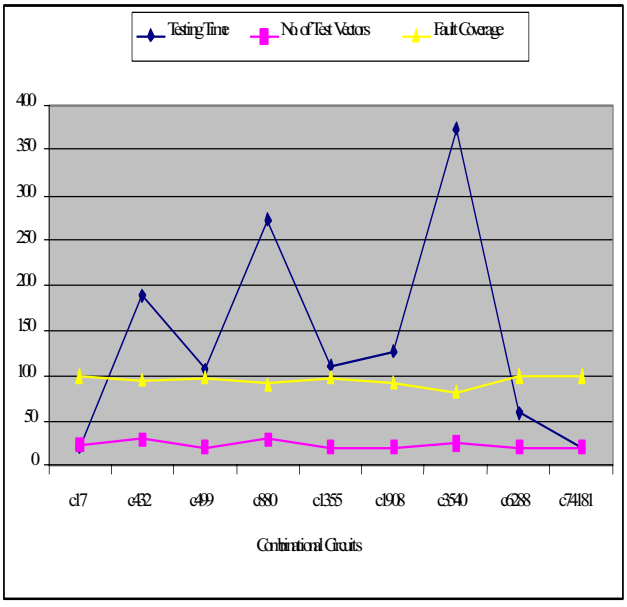

Fig. 5. Number of test vectors, testing time, and fault coverage
in combinational circuit SOC (SOC_comb)
using pseudorandom test vectors.

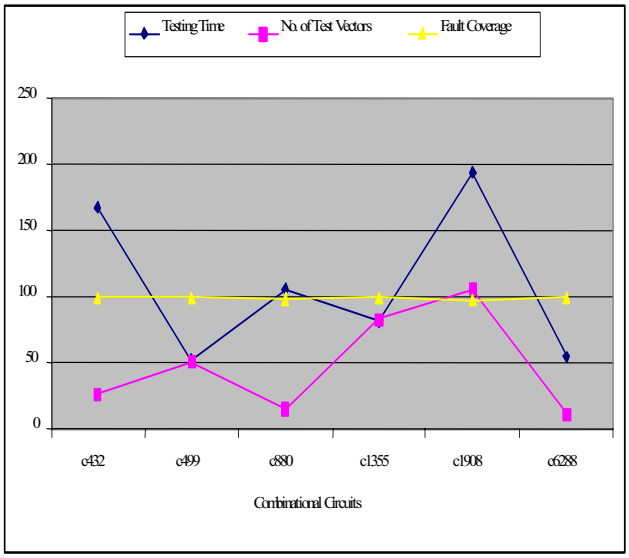

Fig. 6. Number of test vectors, testing time, and fault coverage
in combinational circuit SOC (SOC_comb)
using deterministic compacted test sets (MinTest).

Table 1. Fault Coverage, Number of Injected Faults, Test Vectors, and Detected Faults for Mixed_SOC_1

Circuit name	No. of test vectors used for simulation	No. of faults injected	No. of faults detected	Fault coverage (%)
s27	14,999	34	30	88.23
s298	14,999	272	244	89.7
s344	14,999	234	189	80.77
s349	14,999	222	217	97.75
c17	23	22	22	100
c1355	20	146	142	97.26
c1908	20	119	107	92.241

Table 2. Fault Coverage, Number of Injected Faults, Test Vectors, and Detected Faults for Mixed_SOC_2

Circuit name	No. of test vectors used for simulation	No. of faults injected	No. of faults detected	Fault coverage (%)
s420	14,999	178	38	25.28
s444	14,999	54	36	66.67
s820	19,999	194	132	68.04
c880	30	172	156	90.7
c3540	25	144	119	81.25
c6288	20	126	126	100
c74181	20	44	44	100

Table 3. Fault Coverage, Number of Injected Faults, Test Vectors, and Detected Faults for Mixed_SOC_3

Circuit name	No. of test vectors used for simulation	No. of faults injected	No. of faults detected	Fault coverage (%)
s444	14,999	194	132	68.04
s820	14,999	182	171	93.96
c880	30	172	156	90.7
c3540	20	144	119	81.25

Table 4. Fault Coverage, Number of Injected Faults, Test Vectors, and Detected Faults for Mixed_SOC_4

Circuit name	No. of test vectors used for simulation	No. of faults injected	No. of faults detected	Fault coverage (%)
s349	14,999	222	217	97.75
s382	14,999	174	77	76.55
c432	30	86	81	94.19
c499	20	146	142	97.26
c74181	20	44	44	100

V. CONCLUSIONS

Embedded cores-based design paradigm has evolved from the necessity to increase design productivity and decrease time-to-market, but as a result has created numerous challenging problems for the test design community. Keeping in view the many formidable issues that arise in testing these cores-based SOCs, the present paper provides approaches to developing test environment and test methodologies for digital SOCs. The test access mechanism which plays an important role in transporting the test patterns to the desired cores and the resulting responses to the output pins of the SOCs was implemented as a plain signal transport medium, being shared by all the cores in a given SOC. The faults were injected using fault simulator that generates tests for the core under test. The selection of the individual cores was taken care of by the program running in the background. The entire test process runs without any intervention from outside.

REFERENCES

[1] R. Rajsuman, *System-on-a-Chip: Design and Test*. Boston, MA: Artech House, 2000.

[2] K. Chakrabarty, V. Iyengar, and A. Chandra, *Test Resource Partitioning for System-on-a-Chip*. Boston, MA: Kluwer, 2002.

[3] K. Chakrabarty (Editor), *SOC (System-on-a-Chip) Testing for Plug and Play Test Automation*. Boston, MA: Kluwer, 2002.

[4] S. Mourad and Y. Zorian, *Principles of Testing Electronic Systems*. New York: Wiley, 2000.

[5] T. W. Williams and K. P. Parker, "Testing logic networks and design for testability", *Computer*, vol. 21, pp. 9–21, Oct. 1979.

[6] S. R. Das, C. V. Ramamoorthy, M. H. Assaf, E. M. Petriu, and W. –B. Jone, "Fault tolerance in systems design in VLSI using data compression under constraints of failure probabilities", *IEEE Trans. Instrum. Meas.*, vol. 50, pp. 1725–1747, Dec. 2001.

[7] S. R. Das, M. H. Assaf, E. M. Petriu, and W. –B. Jone, "Fault simulation and response compaction in full scan circuits using HOPE", *IEEE Instrum. Meas. Tech. Conf.*, vol. 1, 2002, pp. 607–612.

[8] S. R. Das, C. Jin, L. Jin, M. H. Assaf, E. M. Petriu, W. –B. Jone, and M. Sahinoglu, "Implementation of a testing environment for digital IP cores", *Proc. IEEE Instrum. Meas. Tech. Conf.*, vol. 2, 2004, pp. 1472–1477.

[9] M. H. Assaf, S. R. Das, E. M. Petriu, L. Jin, C. Jin, D. Biswas, V. Groza, and M. Sahinoglu, "Hardware and software co-design in space compaction of digital circuits", *Proc. IEEE Instrum. Meas. Tech. Conf.*, vol. 2, 2004, pp. 1503–1508.

[10] M. H. Assaf, "Digital Core Output Test Data Compression Architecture Based on Switching Theory Concepts", Ph.D. dissertation, School of Information Technology and Engineering, University of Ottawa, Ottawa, ON, Canada, 2003.

[11] C. Jin, "Test Implementation of Embedded Cores-Based Sequential Circuits Using Verilog HDL Under Altera Max Plus II Development Environment", M.A.Sc. thesis, Department of Systems Science, University of Ottawa, Ottawa, ON, Canada, 2003.

[12] Y. Zorian, "Test requirements for embedded core-based systems and IEEE P-1500", *Proc. Int. Test Conf.*, 1997, pp. 191–199.

[13] P. Varma and S. Bhatia, "A structured test reuse methodology for core-based system chip", *Proc. Int. Test Conf.*, 1997, pp. 294–302.

[14] I. Ghosh, S. Dey, and N. K. Jha, "A fast and low cost testing technique for core-based system-on-chip", Proc. Des. Automat. Conf., 1998, pp. 542–547.

[15] P. Harod, "Testing re-usable IP: A case study", *Proc. Int. Test Conf.*, 1999, pp. 493–498.

[16] Y. Zorian, E. J. Marinissen, and S. Dey, "Testing embedded-core-based system chips", *Computer*, vol. 32, pp. 52–60, June 1999.

[17] E. J.Marinissen, S. K. Goel, and M. Lousberg, "Wrapper design for embedded core test", *Proc. Int. Test Conf.*, 2000, pp. 911–920.

[18] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "Efficient test access mechanism optimization for system-on-chip", *IEEE Trans. Computer-Aided Des.*, vol. 22, pp. 635–643, May 2003.

[19] V. Iyengar, K.Chakrabarty, and E. J. Marinissen, "Test access mechanism optimization, test scheduling, and tester data volume reduction for system-on-chip", *IEEE Trans. Comput.*, vol. C-52, pp. 1619–1632, Dec. 2003.

[20] I. Hamzaoglu and J. H. Patel, "New techniques for deterministic test pattern generation", *J. Electron. Test. Appl.*, vol. 15, pp. 63–73, Aug. 1999.

[21] I. Hamzaoglu and J. H. Patel, "Compact two-pattern test set generation for combinational and full scan circuits", *Proc. Int. Test Conf.*, 1998, pp. 944–953.

This research was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant A 4750.