NON-CONCURRENT ON-LINE TESTING VIA SCAN CHAINS

Hussain Al-Asaad and Paolo Moore
Department of Electrical and Computer Engineering
University of California, One Shields Avenue, Davis, CA 95616-5294
Tel: (530) 752-5545
E-mail: halasaad@ece.ucdavis.edu

Abstract — With operational faults becoming the dominant cause of failure modes in modern VLSI, widespread deployment of on-line test technology has become crucial. In this paper, we present a non-concurrent on-line testing technique via scan chains. We discuss the modifications needed in the design so that it can be tested on-line using our technique. We demonstrate our technique on a case study of a pipelined 8x8 multiply and accumulate unit. The case study shows that our technique is characterized by high error coverage, moderate hardware overhead, and negligible time redundancy.

INTRODUCTION

As the ratio of validation/test engineers to design engineers approaching 3 to 1, the verification and testing of the design of a computer chip is becoming the most significant part in the chip's overall production. Before the present age of ever-shrinking geometries, chip real estate was very expensive. Sometimes, there was no room on a single chip for all the desired functionality and therefore, it was not uncommon to have a number of chips delivering the needed functionality. Before silicon was 'cheap', on-line (on-chip) testing would often displace functionality and thus was only used for devices where fail-safe fault-tolerant operation was extremely important. Now that we are in the era of system-on-a-chip, designers are trying to use this newly won silicon by adding new functions that support design-for-testability, design-for-debug, and on-line testing. Furthermore, with transient and intermittent operational faults becoming a dominant failure mode in modern VLSI, widespread deployment of on-line test technology has become crucial. Key benefits of on-line testing include low-latency fault detection and correction, fault effect localization, and fault tolerance [1].

In this paper, we present a new on-line testing method that uses existing scan chains. The paper is organized as follows. We first describe some background on faults and testing methodologies. We then present a general overview of the structure and operation of the circuit under test (CUT), an 8-bit 8x8 multiply and accumulate unit. We then describe the detailed design of the non-concurrent on-line scan chain test module (OST) and its monitoring program that controls its access to the CUT. We finally present an evaluation of the OST implementation and discuss conclusions and possible future work.

BACKGROUND

Faults are physical or logical defects in the design or implementation of a device. Under certain conditions, they lead to *errors*, that is, incorrect system states. Errors induce failures, that is, a deviation from appropriate system behavior. If the failure can lead to an accident, it is a *hazard*. Faults throughout the life of a digital system can be classified into three groups: design, fabrication, and operational faults. Design faults are made by human designers or CAD software (simulators, translators, or layout generators), and occur during the design process. Fabrication defects result from an imperfect manufacturing process. For example, shorts and opens are common defects in manufacturing very large-scale integrated (VLSI) circuits. Operational faults are caused by wearout or environmental disturbances during normal operation of the embedded system. Such disturbances include electromagnetic interference, operator mistakes, and extremes of temperature and vibration. Some design defects and manufacturing faults escape detection and combine with wearout and environmental disturbances to cause problems in the field.

Operational faults are usually classified according to their duration:

- *Permanent* faults remain in existence indefinitely if no corrective action is taken. Many of these are residual design or manufacturing faults.
- *Intermittent* faults appear, disappear, and reappear repeatedly. They are difficult to predict, but their effects are highly correlated. Most intermittent faults are due to marginal design or manufacturing steps. The system works well most of the time, but fails under atypical environmental conditions.
- *Transient* faults appear and disappear quickly, and are not correlated with each other. They are most commonly induced by random environmental disturbances.

On-line testing addresses the detection of operational faults, and is found in computers that support critical or high-availability applications. The goal of on-line testing is to detect fault effects, that is, errors, and take appropriate corrective action. For example, in some critical applications, the system is shut down after an error is detected. In other applications, error detection triggers a reconfiguration mechanism that allows the system to continue its operation, perhaps with some degradation in performance. On-line testing can be performed by external or internal monitoring, using either hardware or software; internal monitoring is referred to as *self-testing*. Monitoring is internal if it takes place on the same substrate as the circuit under test (CUT); nowadays, this usually means inside a single IC—a system-on-a-chip (SOC).

There are five primary parameters to consider in the design of an on-line testing scheme:

- *Error coverage* (*EC*): This is defined as the fraction of all modeled errors that are detected, usually expressed in percent. Critical and highly available systems require very good error detection or *error coverage* to minimize the impact of errors that lead to system failure.
- *Error latency* (*EL*): This is the difference between the first time the error is activated and the first time it is detected. *EL* is affected by the time taken to perform a test and by how often tests are executed. A related parameter is *fault latency* (*FL*), defined as the difference between the onset of the fault and its detection. Clearly, $FL \geq EL$, so when *EL* is difficult to determine, *FL* is often used instead.
- *Space redundancy* (*SR*): This is the extra hardware or firmware needed to perform on-line testing.
- *Time redundancy* (*TR*): This is the extra time needed to perform on-line testing.
- *Power overhead* (*PR*): This is the extra power needed to perform the on-line testing.

An ideal on-line testing scheme would have 100% error coverage, error latency of 1 clock cycle, no space redundancy, no time redundancy, and no power overhead. It would require no redesign of the CUT, and impose no functional or structural restrictions on the CUT. Most on-line testing methods meet some of these constraints without addressing others. Consideration of all the parameters discussed above in the design of an on-line testing scheme can create conflicting goals. High coverage can require high *EL*, *SR* and/or *TR*. Schemes with immediate detection $(EL = 1)$ minimize time redundancy, but require more hardware. On the other hand, schemes with delayed detection $(EL > 1)$ reduce the time and space redundancy at the expense of increased error latency. Several proposed on-line testing techniques that use delayed detection assume equiprobable input combinations and try to establish a probabilistic bound on the error latency [2]. This results in certain faults remaining undetected for a long time because tests for them rarely if ever appear at the inputs of the CUT.

To cover all of the fault types described earlier, two different modes of on-line testing are employed: *concurrent testing* which takes place during normal system operation, and *non-concurrent testing* which takes place during idle times or while normal operation is temporarily suspended. These operating modes must often be overlapped to provide a comprehensive on-line testing strategy at acceptable cost.

Concurrent testing: Concurrent testing continuously checks for errors due to transient or intermittent faults whose effects disappear quickly. However, concurrent testing is not by itself particularly useful for diagnosing the source of errors, so it is often combined with diagnostic software. It may also be combined with non-concurrent testing to detect or diagnose complex faults of all types.

A common method of providing hardware support for concurrent testing, especially for detecting control errors, is a watchdog timer [3]. This is a counter that must be reset by the system on a repetitive basis to indicate that the system is functioning properly. A watchdog timer is based on the

assumption that the system is fault-free—or at least alive—if it is able to perform the simple task of resetting the timer at appropriate intervals, which implies that control flow is correctly traversing timer reset points. Proper system sequencing can be monitored to very high precision by guarding watchdog timer reset operations with software-based acceptance tests that check signatures computed while control flow traverses various checkpoints. More complex hardware watchdogs can be constructed that implement this last approach in hardware [3].

A key element of concurrent testing for data errors is redundancy. For example, *duplication with comparison* (DWC) [4] can detect any single error at the expense of 100% space redundancy. DWC requires two copies of the CUT, which operate in tandem with identical inputs. Their outputs are compared and any discrepancy indicates an error. In many applications, the high hardware overhead of DWC is unacceptable. Moreover, it is difficult to prevent minor variations in timing between duplicated modules from invalidating comparisons. A possible lower-cost alternative is time redundancy. Critical operations can be executed more than once, at diverse time points, and their results compared. This technique is called *double-execution* or *retry*. Transient faults are likely to affect only one instance of the operation and can thus be detected. *Recomputing with shifted operands* (RESO) [4] achieves almost the same error coverage of DWC with 100% time redundancy but very little space redundancy. However, the practicality of double execution and RESO in on-line testing for general logic circuits has not been demonstrated. A third form of redundancy which is very widely used is information redundancy, that is, the addition of redundant (coded) information such as a parity check bit [4]. Such codes are particularly effective for detecting memory and data transmission errors, since memories and networks are susceptible to transient errors, however, coding methods can also detect errors in data computed during critical operations.

Resent research emphasize that transient errors pose a major barrier to robust system design. A system's susceptibility to such errors increases in advanced technologies, making the incorporation of effective protection mechanisms into chip designs essential. A new design paradigm proposed in [5] reuses design-for-testability and debug resources to eliminate such errors.

Non-concurrent testing: This form of testing is either event-triggered (sporadic), time-triggered (periodic), or activated via a test scheduling algorithm. Event-triggered testing is initiated by key events or state changes in the life of a system, such as start-up or shutdown, and its goal is to detect permanent faults. It is usually advisable to detect and repair permanent faults as soon as possible. Event-triggered tests resemble manufacturing tests. Any such test can be applied on-line, as long as the required testing resources are available. Typically the hardware is partitioned into components, each of which is exercised by tests specific to that component.

Time-triggered testing is activated at predetermined times in the operation of the system. It is often done periodically to detect permanent faults using the same types of tests applied by event-triggered testing. This approach is especially useful in systems that run for extended periods, where no significant events occur that can trigger testing. Periodic testing is also essential for detecting intermittent faults. Such faults typically behave as permanent faults for short time intervals. Since they usually represent conditions that must be corrected, diagnostic resolution is important. Periodic testing can identify latent design or manufacturing flaws that only appear under the right environmental conditions. Note that time-triggered tests are frequently partitioned and interleaved, so that only part of the test is applied during each test period.

As noted above, for critical or highly available systems, it is essential to have a comprehensive approach to on-line testing that covers all expected permanent, intermittent, and transient faults. In recent years, built-in self-test (BIST) has emerged as an important method for testing manufacturing faults, and it is increasingly promoted for on-line testing as well. BIST is a design-for-testability technique that places the testing functions physically with the CUT, as illustrated in Figure 1. In normal operating mode, the CUT receives its

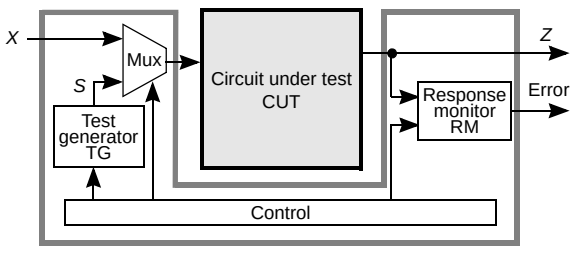

Figure 1 Generic BIST scheme.

inputs *X* from other modules and performs the function for which it was designed. In test mode, a test pattern generator circuit TG applies a sequence of test patterns *S* to the CUT, and the test responses are evaluated by a response monitor RM. In the most common type of BIST, test responses are compacted in RM to form (fault) *signatures*. The response signatures are compared with reference signatures generated or stored on-chip, and the error signal indicates any discrepancies detected.

BIST can be used for non-concurrent, on-line testing of the logic and memory parts of a system [6]. It can readily be configured for event-triggered testing, in which case, the BIST control can be tied to the system reset so that testing occurs during system start-up or shutdown. BIST can also be designed for periodic testing with low fault latency. This requires incorporating a testing process into the CUT that guarantees the detection of all target faults within a fixed time. We next describe our on-line BIST technique using scan chains.

ON-LINE TESTING VIA SCAN CHAINS

Circuit under test: The CUT for our case study is an 8-bit 8x8 multiply and accumulate unit (MAC). MAC units are commonly used in many important DSP filtering and frequency transforming functions. The CUT uses distributed arithmetic to perform multiplication based on a look-up table scheme. It computes the arithmetic sum of products or vector dot product much needed in matrix manipulation calculations.

The arithmetic sum of products that defines the response of linear, time-invariant networks can be expressed as:

$$y(n) = \sum_{k=1}^{K} A_k x_k(n)$$

where $y(n)$ is the response of network at time n, $x_k(n)$ is the k^{th} input variable at time n, and A_k is the weighting factor of k^{th} input variable that is constant (time-invariant) for all n.

In filtering applications, the constants (A_k) are the filter coefficients and the variables (x_k) are the prior samples of a single data source (for example, an analog to digital converter). In frequency transforming – whether the discrete Fourier or the fast Fourier transform – the constants are the sine/cosine basis functions and the variables are a block of samples from a single data source. Examples of multiple data sources may be found in image processing [7].

The MAC is fed a matrix row of eight 8-bit samples in a 64-bit word *d* (corresponding to *x* in the above equation) and it outputs a 12-bit *result* (corresponding to *y* in the above equation) after 4 clock cycles. It has three control signals: a *clk* signal, a *reset* signal and a *load* signal. Two ROMs (*rom_a* and *rom_b*) serve as the distributed arithmetic look-up tables (DALUTs) containing the matrix coefficients A_k. The block diagram for the MAC is presented in Figure 2.

On-line scan chain test module (OST): Scan chain testing is used to test sequential logic circuits by partitioning them into a series of combinational logic blocks separated by scan chain registers. This segmenting allows for controllability of inputs and observability of outputs of the combinational logic blocks that otherwise would be inaccessible within the internals of the CUT. Our OST that we describe next is fundamentally a form of BIST.

The OST performs an on-line scan test of a selected CUT. The test vectors are fed into the OST from the data bus of an external memory module. The OST then scans the test vector bit-by-bit into specialized scan registers in the CUT. After the complete test vector is scanned into the chip and the OST is granted control of the CUT by an OST monitor program running on the operating

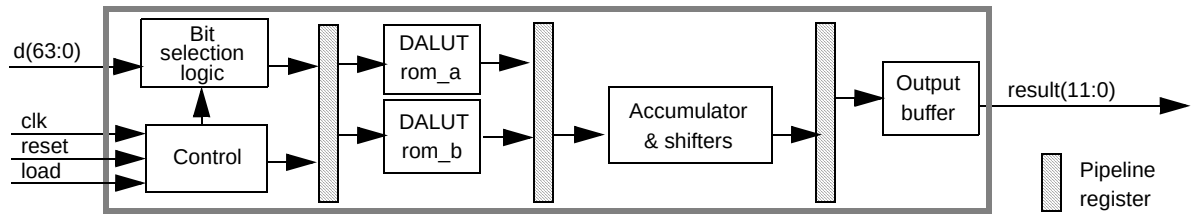

Figure 2 Block diagram for pipelined MAC unit.

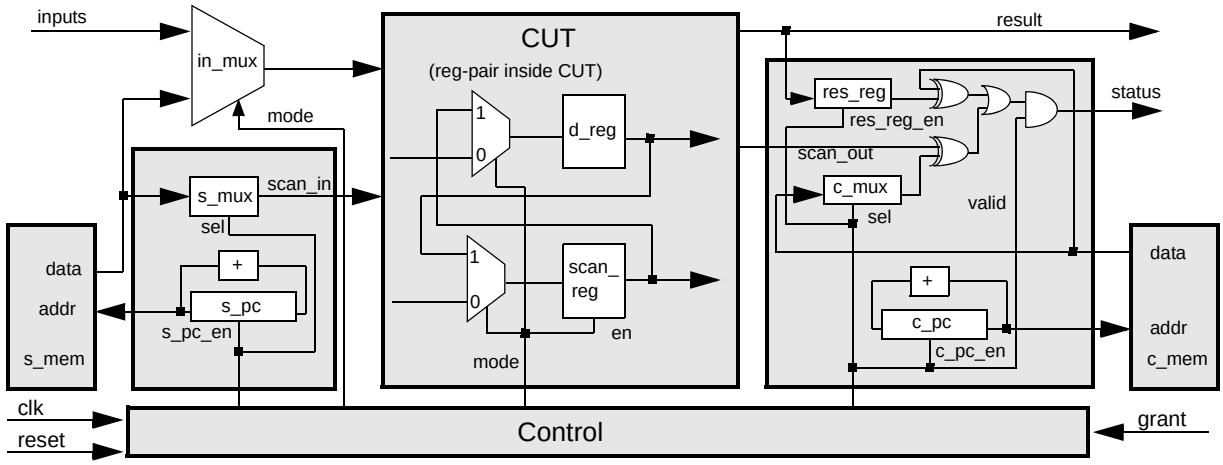

Figure 3 Block diagram of on-line scan chain testing.

system, a 1-cycle mode switch occurs. During the mode switch, the data bits in the scan registers are swapped with the data bits in the CUT's normal operation registers. This effectively takes the CUT 'off-line' for a single cycle so that the scanned-in test vector can drive the data/control paths. The state of the CUT is saved during this swap so that after the 1-cycle scan testing operation, the state is reinstated. In addition, the OST drives all input lines except the clock into the CUT. Immediately after mode switch, all output generated by the CUT during the mode switch is saved in result registers for later comparison with expected data patterns stored and read out from another external memory module. Thus, the OST borrows the CUT from normal operation during test mode and 'returns' it when the test mode is completed. After that, the OST compares the experimental values with expected values and asserts a status line if a mismatch occurs. It is up to an OST monitor program to take any appropriate action(s). A block diagram of the OST is presented in Figure 3.

The OST is organized into two functional sections. The scan functional section is responsible for reading out all inputs to the CUT (combinational inputs and scan chain test vectors) from the scan memory module and shifting the test vector into the scan chain of the CUT. The second section, the compare section, is responsible for reading the expected or 'golden' values from the compare memory, comparing it to the experimental values from the CUT generated during the mode context switch and driving the status line high if a mismatch occurs.

For every register that would normally appear in the architected design of the CUT, a *reg_pair* is substituted. A *reg_pair* is two registers that each has an input side multiplexer whose selection bit is driven by the *mode* signal. These multiplexers drive the data lines of their respective registers. When *mode* = 1, the values in the *d_reg* and *scan_reg* are swapped so that the machine state is saved in the *scan_reg* and the data/control paths of the CUT are driven by the 'scanned-in' test vector now residing in the normal operational *d_reg*. This allows for a context switch of only one cycle.

The OST is serviced by two ROM memory modules. The 64Kx256 *s_mem* module provides all the input stimulus and test vectors needed for the CUT. It returns, on the 256-bit *s_mem_data* bus, the 73-bit scan chain test vector and CUT inputs indexed by the 16-bit memory address generated by the *s_pc* program counter. In like manner, the 64Kx128 *c_mem* module contains all the expected results correlated with both the scan chain test vector and CUT outputs. It returns, on the 128-bit *c_mem_data* bus, the data word indexed by the 16-bit memory address generated by the *c_pc* program counter.

Each of the data buses from the memory modules is fed into multiplexer. The multiplexer has the same number of inputs as there are bits in the scan chain, 73 in the case of the MAC. A subset of *s_mem_data*, appropriately sized to the length of the scan chain for the CUT, is fed into the *s_mux*. The output of the *s_mux* supplies the OST output signal *scan_in* which feeds the scan chain into the CUT. The remaining bits of *s_mem_data* (used to supply the off-chip stimulus during the testing of the CUT) are fed into the CUT input multiplexers.

In like manner, a scan chain length subset of *c_mem_data* is fed into the *c_mux* multiplexer, whose output is later used in the comparison logic. Both multiplexers use the same *sel* signal generated by the control unit.

The output of *c_mux* (a single bit selected from bits 0-72 of the *c_mem_data*) is XORed with the OST input *scan_out* generated by the scan chain in CUT after a mode context switch. The remaining bits of *c_mem_data* (containing the expected 'golden' values for the MAC output) are XORed with the contents of *res_reg*. These two 'intermediate' XORed results are then ORed together and fed into a two-input AND gate along with the *valid* signal. The *valid* signal is asserted high for 73 clock cycles after a mode context switch signifying that the compare operation is indeed valid rather than garbage comparisons. The output of this AND gate is the active high OST output *status*. The signal *status* is driven high when an error has been detected. It is left to the OST monitoring program in the operating system to monitor this assertion and perform any subsequent actions.

The control unit has three inputs: a clock signal, *clk*, a reset signal, *reset* and an active high *grant* signal. The grant and reset signals are generated from an OST monitor running in the operating system controlling the CUT. When either a key event occurs or a preset period of time elapses, the OST monitor asserts the grant signal to alert the OST that the CUT can be 'borrowed' for scan testing.

The control logic loop consists of basically two 73 clock cycle blocks, the scan block and the compare block. These two blocks are separated by a single cycle mode context switch, which is triggered if the grant signal is asserted after the input data (off-chip stimulus vector and scan chain test vector) is fully loaded in during the scan block. After the mode context switch in which *mode* goes from a value of 0 to 1 and back to 0, the second 73 clock cycle block, the compare block, begins. Also, during the mode context switch, the *res_reg_en* signal is asserted so that the normal operational output of the CUT is captured for later comparison.

During the first cycle of the scan block, the *c_pc* is incremented via assertion of the *c_pc_en* signal so that the *c_mem* memory module can drive the *c_mem_data* bus with the next values needed during the compare block. At the last cycle of the scan block, the *grant* signal is polled. Only when the signal is asserted does the mode context switch occur. Also, the write enable signal, *en*, for the *s_reg* registers in the scan chain is deasserted after the scan chain test vector is fully loaded. This prevents erroneous unwanted shifting during *grant* signal polling or during the mode context switch.

During the mode context switch, the CUT is the only module performing a useful function. The OST waits during this cycle. During the other parts of the control loop, the two 73 clock cycle blocks, the control unit synchronizes and cycles the selection bits of the two memory data bus selection multiplexers through the values 0 to 72, selecting the appropriate bits from their respective memory module data buses. This cycling is temporarily suspended during polling of the grant signal and the mode context switch.

After the mode context switch, the compare block begins with the assertion of the valid signal. Also, the scan chain write enable signal *en* is asserted allowing the scan chain to be loaded. Though the scan chain is capable of being loaded, the *s_pc* has not yet fetched the next test vector. During the first cycle of the compare block, the *s_pc* is incremented via assertion of the *s_pc_en* signal so that the *s_mem_data* bus can have the next values needed during the next scan block in the next iteration of the control loop.

Operating system OST monitor: An OST monitor program running in the operating system controls the assertion of the grant signal that signifies the CUT's availability for testing, and drives the reset signal for the OST. It is the responsibility of the monitor to take any subsequent action if the OST status signal is ever asserted.

The OST monitor can be programmed to assert the grant signal when key events occur (event-triggered OST operation). An obvious choice for an event that appears in the key event list is when the CUT is idle. In addition, if certain regions of the program running on the CUT have a high incidence of errors associated with it, the OST monitor can be programmed to assert the grant signal during appropriate times in this error-prone operating region of the CUT.

The OST monitor can also be programmed to periodically assert the grant signal allowing the OST access to the CUT (time-triggered OST operation). This would assure that the OST tests the CUT once every predetermined interval.

EVALUATION AND CONCLUSIONS

To evaluate our on-line scan testing technique, we need to examine the following basic parameters: error coverage, error latency, time redundancy, space redundancy, and power overhead.

Error coverage is an artifact of the test vector generator program and the error model used to test for particular faults. So, we can easily achieve high error coverage by selecting a good test set. Of course, we need to select the smallest test set size since the test set and the CUT response to it need to be stored in the memory modules needed by our OST. So, a smaller test size implies less space redundancy and power overhead.

Error latency depends both on the frequency (scheduling) of the testing and the duration of it. So, if tests are done more frequently, the error latency will be decreased.

The specialized *reg_pair* module was designed so that the OST would only borrow the CUT for one clock cycle. The *reg_pair* allowed the CUT machine state to be saved so that the operation in progress could be reinstated rather restarted after the testing. After the scan chain was shifted 73 clock cycles into the CUT and a one clock cycle mode context switch, it took 73 clock cycles for the comparison operation. So, although we need 147 total clock cycles for one test vector to be processed, only one clock cycle is taken from normal operation. So, the time redundancy is negligible.

A *reg_pair* is substituted for every normal operational register originally architected in the CUT. This adds an extra register and two 2-to-1 1-bit multiplexers for every original register. In the case of the MAC, an additional 73 registers and 146 multiplexers were added. For the OST implementation, the hardware needed is 69 registers, 67 logic gates and 2 ROM memory modules (64Kx256 and 64Kx128). Although the amount of extra hardware seems moderate in our case study, it decreases as the size of the CUT increases.

With the extra hardware needed for the OST operation, we definitely need extra power. The power overhead is directly related to the space redundancy. However, the peak power of the overall system (CUT and OST) can be the same as the original system (CUT alone) if the CUT power is cut-off when the OST is working and vice versa.

We have demonstrated the feasibility of bringing scan chain testing on-line. The particular implementation in this demonstration provided a simple straightforward design. It had high error coverage, moderate hardware overhead, and negligible time redundancy.

Possible future improvements on the implementation of non-concurrent on-line scan chain testing include a pipelined design to overlap the compare block of a test vector with the scanning in of the next test vector. This will decrease the time necessary for all test vectors to be executed by 50%. In addition, multiple scan chains can be used to decrease the time needed for shifting in the test vectors at the expense of additional cost of I/O pins. Finally, the OST can be used to diagnose and locate the errors. The test vectors need to be grouped in such a way the OST monitor program is able to associate the program counter values of the OST with particular logic regions of the CUT.

REFERENECES

[1] R. Karri and M. Nicolaidis, “Online VLSI testing”, *IEEE Design & Test of Computers*, Vol. 15, No. 4, pp. 12 - 16, Oct.-Dec. 1998.

[2] K. K. Saluja, R. Sharma, and C. R. Kime, “A concurrent testing technique for digital circuits”, *IEEE Transactions on Computer-Aided Design*, Vol. 7, pp. 1250-1259, Dec. 1988.

[3] A. Mahmood and E. McCluskey, “Concurrent error detection using watchdog processors—A survey”, *IEEE Transactions on Computers*, Vol. C-37, pp. 160-174, Feb. 1988.

[4] B. W. Johnson, *Design and Analysis of Fault Tolerant Digital Systems*, Addison-Wesley, Reading, Massachusetts, 1989.

[5] S. Mitra et al., “Robust system design with built-in soft-error resilience”, *IEEE Computer*, Vol. 38, pp. 43-52, Feb. 2005.

[6] B. T. Murray and J. P. Hayes, “Testing ICs: Getting to the core of the problem”, *IEEE Computer*, Vol. 29, pp. 32-45, Nov. 1996.

[7] Xilinx Inc., “The role of distributed arithmetic in FPGA-based signal processing”, http://www.xilinx.com/appnotes/theory1.pdf.

[8] H. Al-Asaad, B. T. Murray, and J. P. Hayes, “On-line BIST for embedded systems”, *IEEE Design and Test of Computers*, Vol. 15, No. 4, pp. 17-24, November 1998.

[9] J. Savir, “On-line and off-line test of airborne digital systems: a reliability study”, *Proc. International Test Conference*, 2000, pp. 35-44.