Implementation of Test-per-Cycle BIST Architecture with No Storage Requirement

Viraj Bhusari[1]

[1]M.Tech Scholar,Department of Micro and Nano-electronics, VIT University, Vellore, Tamilnadu, India-632014
virajb34@gmail.com

ABSTRACT: Test-per-clock BIST scheme is used to determine for small test vector and short test time. This paper presents a new test-per-clock BIST method that can perform both pattern generation and response compression concurrently in the same LFSR-based design so as to reduce the area overhead. Furthermore, some internal nets are employed in two ways during test application to help reduce test time and test vector: 1) as the observation points to enhance fault detectability and 2) as the test data provider for reseeding the LFSR. These two ways lead to the benefits that all required patterns can be generated on chip and at-speed testing can be carried out without using any external or internal storage device. Experimental results show that the presented method can achieve 100% fault coverage in very short time for large ISCAS (IWLS) benchmark circuits. When compared with a conventional scan-based design, the area overhead is small considering the features of test-per-clock and no requirement of data storage element.

Key words: Test per clock BIST, CSTP, Testing, DFT, MISR, ATPG

1. INTRODUCTION

Built-In Self-Test (BIST) is a design-for-testability (DFT) technique in which testing (test generation, test application) is accomplished through built-in hardware features. It is lead to significant test complexity reduction, specially attractive for embedded cores. Logic Built-In Self-Test (BIST) is used to define an effective testable design technique in which some on-chip test structure is used to test the digital circuit. Pseudorandom test pattern based on linear feedback shift registers (LFSRs) is commonly used for the basis of BIST due to its simplicity and effectiveness. However in the complex circuit often some hard-to-detect faults are there that are random-pattern resistant so a pseudo-random test scheme usually requires long time to reach satisfactory fault coverage.

To overcome this problem, several techniques have been proposed in the literature. The dual mode BIST technique [2-6][10-13] employs both pseudo-random and deterministic patterns to achieve complete fault coverage in a short time, where for the deterministic patterns required on chip ROM it reduces tester memory and test equipment cost or provided from an external tester. However this may induce high chip area overhead. To address this problem, reseeding techniques [2-6] are proposed to compress the required test data into some small-size seeds. The seeds can then be decompressed by an LFSR or similar logic during test application time. Since the required test patterns to cover faults can be derived from the seeds, only a small volume of test vector need be stored. However, an internal or external storage device is still required for the reseeding techniques.

Techniques using circuit responses to generate the required deterministic patterns are also proposed. The circular self-test scheme [7] replaces each primary IO with a special BIST cell and connects them together with the internal scan cells to form a long circular self-test path (CSTP). Both pattern generation and signature analysis can be done on the self-test path, and thus only small area overhead is needed. However the fault coverage may be degraded if some states required to detect some faults cannot be reached by the self-test path. To address this issue the Circular BIST proposed in [8] adds some state skipping logic in jump to states that detect hard-to-detect faults so as to achieve full fault coverage promptly. However, in the case that many hard-to-detect faults

exist, a large state-skipping logic is needed to detect all faults. To address this problem, a few observation points are inserted to enhance the detectability of hard-to-detect faults. Although the observation point insertion requires circuit modification, the previous work in [8] shows that it is efficient to alleviate the increasing area overhead. The deterministic circular self-test path (DCSTP) in [9] adds some jumping logic that can change the contents of the scan register to any deterministic state, by which all the desired patterns can be generated efficiently with a small area overhead.

In contrast to utilizing only the output responses for test generation, the authors in [10] propose to connect some internal nets of a CUT to its inputs so as to provide the required logic values of the next pattern directly. To ensure high-enough fault coverage, different sets of net connections and corresponding initial stored patterns are thus required, which may lead to high area overhead. In [11], another mixed-mode BIST method is proposed, where all deterministic patterns can be generated on chip in real-time, thereby obviating the need of a storage device. By using a set of pseudo-random patterns together with the required deterministic patterns generated by appropriately connecting some internal nets, this BIST scheme can achieve high fault coverage in a very short time. In [12], an efficient net identification algorithm is proposed to identify a minimal set of internal nets for the test architecture in [11]. However the methods in [10-11] still require many sets of internal nets to generate all deterministic patterns. In [13] a new reseeding technique is proposed which employs the internal net responses as the control signals to generate required seeds. In this way, only a few internal nets are needed for complete fault coverage.

In general the BIST methods can be classified into test-per scan [2-4] and test-per-clock [5-13] methods according to their test application schemes. The test-per-scan method serially loads each test pattern into scan chains bit-by-bit. The resulting test responses are captured into the scan chains and scanned out for examination. The test-per-clock method applies one test pattern in one test cycle. All output responses are captured and loaded to a parallel response monitor at each test cycle. Though the area overhead required for the response monitor may be larger, the test-per-clock schemes do have the advantage of much shorter test time since one test pattern can be applied for each clock cycle.

In this paper, we consider the test-per-clock BIST scheme. We shall present a BIST method that employs an LFSR-based multiple-input-signature-register (MISR) to perform both pattern generation and response compression concurrently in the same registers hence no additional response monitor is required. In addition, in our scheme some internal nets are employed in two ways during test application to help reduce test time and test vector. The first one employs some internal nets as additional observation points to enhance fault detectability, and the second one uses some internal nets as the test data provider for reseeding the LFSR. These two ways lead to the benefits that all required patterns can be generated on chip and at-speed testing can be carried out without using any external or internal storage device. Efficient algorithms to identify appropriate internal nets as observation points are presented. A novel net selection logic unit is also developed that can select appropriate internal nets to generate all required 6seeds. To avoid any degradation on the circuit performance, all internal nets in the critical paths will be not allowed in our BIST technique. Experimental results show that our method can achieve 100% fault coverage in a short time for large ISCAS (IWLS) benchmark circuits.

2. PROPOSED BIST ARCHITECTURE

As shown in Fig. 1, the proposed BIST architecture consists of a MISR, a number of multiplexers (MUX) each of which provides one input to the MISR, XOR-tree-based network, a net-selection logic unit and an on-chip BIST control unit.

Figure1. BIST Architecture

Figure2. Proposed BIST Architecture

Similar to other test-per-clock BIST methods [5-13], we insert a scan cell for each primary input and connect them with the original scan cells of the circuit under test (CUT) to form the MISR such that the length of the MISR is equal to the sum of the numbers of primary inputs and pseudo-primary inputs to the CUT. For simplicity we will call this number as #PPI. Only #PPI scan flip flops in the MISR are needed for our BIST to perform both pattern generation & response analysis. All responses in the primary outputs and pseudo primary outputs of the CUT are compacted by using an XOR-tree-based network before entering the MISR. The total number of these outputs is denoted as #PPO. During test application, the required seeds for the MISR can be generated by simply resetting the MISR or through the net-selection logic unit based on the current responses of some pre-determined internal nets. The 2-to-1 multiplexers shown in Figure 1 are used to switch between the reseeding mode and the pseudo-random test mode. In the reseeding mode, some internal responses are selected by the net-selection logic unit to change the states of MISR so as to generate a new seed. The generation of each seed may require several cycles, called reseeding cycles, to complete. We denote the number of reseeding cycles for one seed by #reseed-cycles.

A large value of #reseed-cycles can reduce the number of used multiplexers, denoted by #MUX which is equal to #PPI / #reseed-cycles. When larger #reseed-cycles is used, more reseeding cycles are needed to generate a new seed. It will decrease the probability to find a feasible set of internal nets for reseeding. In our experiences, setting #reseed-cycles = 3 is feasible for all employed ISCAS benchmarks to generate all required seeds.

When one seed is generated, the pseudo-random test mode is activated to derive a number of pseudo-random patterns. At each cycle of this mode the output responses of the circuit are compacted into the MISR and then used as the test pattern in the next

cycle. In addition to the circuit outputs, #OBP internal nets are chosen as observation points to increase the observability of hard-to-detect faults. Each observation point is directly connected to one of #MUX multiplexers for compaction. The other (#MUX – #OBP) multiplexers are used to compact the circuit output responses. An XOR-tree-based network is added in the front of the multiplexers to compact #PPO pseudo primary outputs. This XOR-tree compactor contains 2-input XOR-gates. The area overhead of the XOR-tree-based network and the MISR in our scheme is much smaller than a conventional, separate response monitor in the test-per-clock scheme.

The control unit shown in the left side of Fig. 1 is employed to control the switching between reseeding and pseudo-random test modes as well as to control the selection of different sets of internal nets for reseeding. The control unit consists of three counters, namely the seed counter, the reseeding-cycle counter and the pseudo-random-cycle counter. The seed counter indicates the index of the current seed, while the reseeding cycle counter and the pseudo-random-cycle counter count the numbers of cycles for generating a new seed and the associated pseudo-random test sequence, respectively.

For increasing the diagnosiability, extra patterns are needed in testing stage are often needed. Similar to the previous work in [13], our design can support this requirement by using an extra diagnosis mode with a few extra logic. For example, we can add some bypass MUX in the outputs of net selection logic unit. When diagnosis mode is activated, we can bypass the net-selection logic and shift-in diagnosis patterns into the MISR via the bypass MUX, then capture and shift-out responses as like the original functionality of scan-based design.

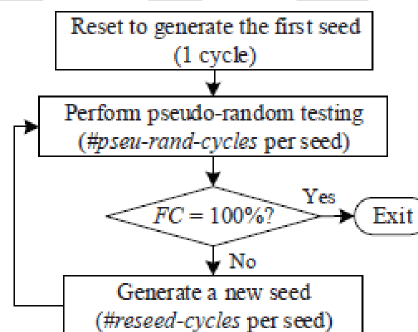

Figure3. Test Generation Flow of our scheme

Fig. 3 shows the test generation flow of our scheme. At the first test cycle, the initial seed is generated by resetting the MISR. The pseudo-random test mode is then executed for a number of cycles denoted by #pseu-rand-cycles which includes (#pseu-rand-cycles-1) cycles to generate a set of pseudorandom patterns and one cycle to compact the responses of the last generated pattern. If the fault coverage FC is not 100%, we activate the reseeding mode to generate a new seed in the following #reseed-cycles cycles and then continue to perform pseudo-random testing again. These steps are repeated until all faults are detected. Assume the total number of seeds for 100% fault coverage is #Seed, the total number of test cycles is 1+(#Seed–1)×#reseed cycles+#Seed×#pseudo-rand-cycles and the total number of applied patterns is #Seed×#pseudo-rand-cycles. In our BIST, the total number of required nets, the area overhead of the net-selection logic unit, and the total test time are all highly dependent on the total number of required seeds #Seed. Next, we present an efficient algorithm which aims to minimize the total numbers of seeds and the required nets without decreasing the fault coverage.

3. SEED IDENTIFICATION AND OBSERVATION POINT

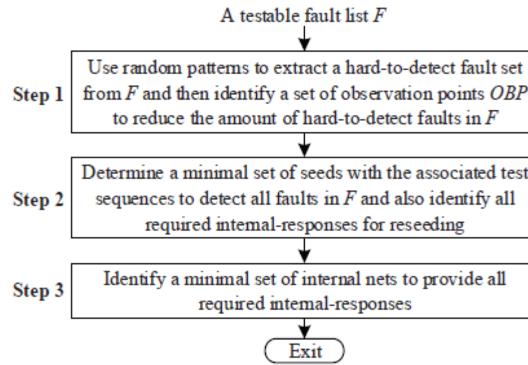

Figure4. Algorithm for observation points

Fig. 4 gives a brief outlines about overall procedure to determine the required observation points, the required seeds and the internal nets for reseeding. In Step 1, some internal nets are identified as the observation points to reduce the number of hard-to-detect faults. This step also determines the structure of the XOR-tree based network based on the identified observation points for the next process. Step 2 determines a set of seeds which together with their derived test sequences can detect all faults in the input fault list F. The required internal responses for reseeding are also recorded in this step. Finally, Step 3 further explores the sharing relations between different internal nets to reduce the amount of required internal nets for reseeding. We will go in detail from above steps (Fig. 4)

3.1 OBSERVATION POINT DETERMINATION

This step identifies a set of observation points to enhance the detectability of hard-to-detect faults. First, a set of random patterns are generated to help extract the hard-to-detect faults. Note that all random patterns mentioned here are used only for the observation point identification process and are not used during the test application time by our BIST design. The generated random patterns are input to a fault simulation process to identify a hard-to-detect fault list from F. The fault effects of the hard-to-detect faults on each internal net are also recorded during the fault simulation process. According to the fault effect information, we select the internal net which can detect most hard-to-detect faults as an observation point, then drop all detected faults from the hard-to-detect fault list and then repeat to select the next observation point until all hard –to-detect faults are detected or no any hard-to-detect faults can be detected. Note that some hard-to-detect faults may not be activated by any identified random patterns and hence have no fault effects on any internal net. For this kind of faults we do not select any observation points. All identified observation points are added to our BIST structure and the XOR-tree-based network is decided accordingly.

3.2 SEED IDENTIFICATION

This step attempts to minimize the total number of seeds so as to reduce the number of internal nets to be connected and the area overhead of the net-selection logic. First, an all-0 or all-1 pattern is selected to be the first seed SC dependent on which one detects more faults. We add SC to the seed set S and use it to derive a set of pseudo-random patterns. A user-defined input parameter #pseudo-rand-cycles to limit the maximum number of pseudo-random patterns generated from a pre-computed seed. All faults detected by SC or the derived pseudo-random test patterns are dropped from F. If some faults are still undetected, a new seed is identified. Otherwise, this step is terminated. The initial state of the next seed SN can be derived from the responses of the last pseudo-random pattern generated from the previous seed and the polynomial function of MISR. For each unspecified bit (X-bit) in SN, we

extract the corresponding linear equation of each X-bit and present a test embedding process to assign values to the X-bits for detecting more faults.

First, this process employs a commercial ATPG tool to generate a partially-specified pattern set P for all the remaining faults by using all the specified bits in SN as the input constraints for test generation. The patterns which can detect most faults are first selected from P as PS. If PS can be successfully embedded to SN by solving the corresponding set of linear equations for SN, SN together with all MISR states are updated. Otherwise, PS is removed from P and another pattern in P is considered until one pattern can be embedded to SC or P becomes empty.

After test embedding, appropriate internal net responses are identified to generate SN and all the remaining X-bits in SN are filled to reduce the required internal nets. A special data structure, called configuration matrix (CM) as presented in [11-13] is used in our method to keep track of the current response of all internal nets under the current MISR state, and the validity of the connections between the internal nets and the multiplexer inputs for generating a new seed. Each seed SN corresponds to an individual CM. Similar to the method in [13], the CM for SN is initialized by the net responses at the last pseudo-random pattern generated by the previous seed SN-1. Then, the CM is updated to keep all valid connections for generating SN. Unlike the method in [13] which generates SN at one cycle and updates CM one time for SN, the CM-updating step here will be executed for #reseed-cycles times until SN are generated. This difference is to reduce multiplexer. However if a large #reseed-cycles is used, we may fail to find any valid connections for the new seed. To avoid this problem, we have tried various values for #reseed-cycles, and found that by setting #reseed-cycles = 3, a set of valid connections for the required seeds in all employed cases can always be identified.

To enlarge the solution space, not only the responses of internal nets but also their inversions are recorded in CM. When feasible net responses for generating SN are identified, a fully-specified seed SN can be obtained. Then, we add the SN to S and then generate pseudo-random patterns again. If all faults in F are detected, this flow completes. Otherwise, more seeds are determined until all the remaining faults are detected.

4. CONCLUSION

This paper proposes a novel BIST scheme that uses a few internal nets to provide extra observation and generate all required seeds for complete testing. This method successfully addresses the drawback of large area overhead which generally exists in the test-per-clock BIST schemes. The results on large ISCAS and IWLS benchmark circuits show that much smaller area overhead and no storage are needed for our BIST method to achieve 100% fault coverage.

REFERENCES:

[1] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*, Morgan Kaufmann, 2006.

[2] B. Koenemann, “LFSR-coded test patterns for scan designs,” *Europe TestConf.*, 1991, pp.237-242.

[3] H.-S. Kim and S. Kang, “Increasing encoding efficiency of LFSR reseeding-based test compression,” *IEEE Trans. on Computer-Aided Design of Integr. Circuits and Syst.*, 25(5), pp. 913-917, 2006.

[4] V. Tenentes, X. Kavousianos and E. Kalligeros, “Single and variable state-skip LFSRs: bridging the gap between test data compression and test set embedding for IP cores,” *IEEE Trans. on Computer-Aided Design of Integr. Circuits and Syst.*, 29(10), pp. 1640-1644, 2010.

[5] E. Kalligeros, D. Bakalis, X. Kavousianos and D. Nikolos, “Reseeding based test set embedding with reduced test sequences,” *Int’l Symp. On Quality Electronic Design*, 2005, pp. 226-231.

[6] T. Liu, J. Kuang, Z. You and S. Cai, "An effective deterministic test generation for test-per-clock testing," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 29, No. 5, pp. 25-33, May 2014.

[7] A. Krasniewski and S. Pilarski, "Circular self-test path: A low-cost BIST technique for VLSI circuits," *IEEE Trans. on Computer-Aided Design of Integr. Circuits and Syst.*, 8(1), pp. 46-55, 1989.

[8] N. A. Touba, "Circular BIST with State Skipping," *IEEE Trans. on Very Large Scale Integr. Syst.*, 10(5), pp. 668- 672, 2002.

[9] K. Wen, Y. Hu, and X. Li, "Deterministic Circular Self-Test Path,"*Tsinghua Science and Technology*, 12(1), pp. 20- 25, 2007.

[10] K. Jishun, O. Xiong, and Y. Zhiqiang, "A novel BIST scheme using test vectors applied by circuit-under-test itself," *Asian Test Symp.*, 2008, pp.75-80.

[11] W.-C. Lien and K.-J. Lee, "A complete logic BIST technology with no storage requirement," *Asian Test Symp.*, 2010, pp. 129-13.

[12] W.-C. Lien, T.-Y. Hsieh, and K.-J. Lee, "Routing-efficient implementation of an internal-response-based BIST architecture," *Int'l Symp. on VLSI Design, Autom. and Test*, 2012, pp. 1-4.

[13] W.-C. Lien, K.-J. Lee, T.-Y. Hsieh, and K. Chakrabarty, "Efficient LFSR Reseeding Based on Internal-Response Feedback," *Journal of Electronics Testing".*

IJERGS