ISSN: 0974-2115
www.jchps.com Journal of Chemical and Pharmaceutical Sciences

A BIST TPG for high fault coverage and low power dissipation by VHDL implementation

[1]Prabhakaran G*, [2]Praveen A
[1]Department of Aeronautical Engineering, Jeppiaar Engineering College, Chennai, India.
[2]VLSI Design, Jeppiaar Engineering College, Chennai, India.
Corresponding Author Email: prabhakaran_govind@rediffmail.com

ABSTRACT

Scan-based Built-In Self-Test (SC-BIST) Test Pattern Generator (TPG) reduces transitions which occur at the inputs of scan during scan-shift operation taking place. Hence decreases switching activity in the circuits under test (CUT). In this paper, the proposed BIST is combination of LT-RTPG and 3-WRBIST. LT-RTPG detect easy-to-detect faults and 3-WRBIST detect faults that remain undetected after LT-RTPG test patterns are applied. On the overview of BIST TPG which does not need to modify mission logics, by which performance degradation takes place. This paper investigates, overhead TPG of a low hardware for SC-BIST that can reduce switching activity in CUTs during BIST. This achieves excellent fault coverage with reasonable lengths of test sequences. The proposed BIST can significantly reduce switching activity during BIST while achieving 100% fault coverage by implementing the circuits in VHDL (VHSIC Hardware Description Language). Reduction in switching activity is larger in large circuits. The proposed BIST can be implemented with low area overhead is observed from the experimental results.

Keywords: Built-In, Component, Formatting, Insert, Styling.

INTRODUCTION

With scan circuitry, BIST circuitry, greatly enhances testability of design. BIST circuitry performs itself of testing up the device, and minimizing the requirement of external equipment for testing. Test generation facilitated by scan circuitry and thereby external tester usage can be reduced. There are two main types of scan circuitry-internal scan and boundary scan. Internal scan is to increase its testability by its internal modification of design circuitry. Boundary scan adds scan circuitry around the periphery of the design to make internal circuitry on a chip accessible via a standard board interface, while scan design modifies circuitry within the original design. This enhances testability of the chip, the chip I/O pads, which includes the interconnections of the chip to other circuitry. BIST structures can test various types of circuitry, from random logic to regular structures such as memory devices. Large, complex circuits often contain difficult-to-test portions of logic. Even the most testable designs, if large, can require extensive test generation time, tester pattern memory, and tester application times—all of which are expensive, yet necessary, to adequately test devices in a classic test scenario. Additionally, because memory faults differ from random logic faults, and memories reside within larger designs, ATPG does not provide an adequate memory testing solution. Memory BIST addresses these issues. BIST provides a memory test solution without sacrificing test quality. In many cases, BIST structures can eliminate or minimize the need for external test pattern generation (and thus, tester pattern memory) and tester application time. In addition, a designer can exercise BIST circuitry within a design, running tests at speed due to the proximity of the BIST circuitry to the memory under test. A designer can also run a memory BIST process from within higher levels of the design.

MEMORY BIST

BIST and memory bist: BIST circuitry varies greatly depending on its application, and yet all variations have a common purpose. BIST structures generate patterns and compare output responses for a dedicated piece of circuitry. Circuitry target types can vary. You can implement BIST on entire designs, design blocks, or structures within design blocks. Additionally, the pattern generation, as well as the output comparison circuitry, can vary. BIST circuitry can generate patterns based on a variety of algorithms, each focused on a particular type of circuitry or fault type. The comparison function has a number of unique implementations, including actual comparators as well as signature. The main objective of Memory BIST circuitry should have RAM and ROM models within a design. Testing RAM and ROM differs from testing random logic. Thus, memory BIST implements circuitry and algorithms effective for testing faults common to RAM and ROM. Memory models consist of three basic blocks: an address decoder, read/write logic, and the memory cell array, as shown in Fig.1. To be most effective, memory BIST must detect faults in all three blocks. These faults include stuck-at, transient, coupling, and neighborhood pattern sensitive faults.

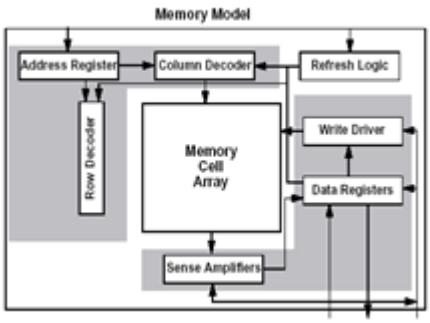


Fig.1.Memory Block Diagram

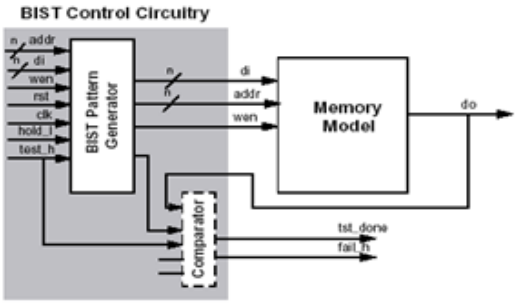


Fig.2.Basic Memory BIST Block Diagram

Simple Memory BIST Architecture: Memory BIST uses one or more algorithms specifically designed for testing memory faults. In any case, memory BIST circuitry generates patterns and detects device failures as shown in Fig.2. This diagram shows an optional comparator that compares the actual memory model's response against a known good memory's response. Instead of a comparator, a compressor (MISR) could provide a response signature used for failure analysis.

Memory BIST Insertion with MBIST Architect: MBIST Architect is the Mentor Graphics memory BIST insertion tool. MBIST Architect creates and interconnects RTL-level BIST logic to test your design's memory. RTL-level BIST synthesis. Insertion of BIST circuitry at the RTL level, moving generation of test circuitry to earlier in the design process. Generation of VHDL output that complies with any standard VHDL simulator or synthesis tool, such as Quick HDL (VHDL), Auto Logic II, and Synopsys' Design Compiler. Generation of Verilog (OVI version 2.0) that complies with any standard Verilog simulator or synthesis tool, such as Verilog-XL simulator, Quick HDL (Verilog), Synopsys' Design Compiler, and Auto Logic II. Generation of default or user-customized BIST architectures. Generation of a finite state machine to produce deterministic patterns using March testing, diagonal, and unique address algorithms, among others. Also, the tool supports different algorithms applied to different ports of multi-port memories. Allows user-specifiable data backgrounds for use in conjunction with March testing to target specific memory faults not proven detected by the standard algorithms.Generation of a single BIST controller for multiple memories, and parallel test data application for fast test application times. Automatic connection of BIST circuitry to memory models. Generation of a test bench, which allows testing of the BIST logic after interconnection with the memory models. Utilizes the DFT library format common to DFT Advisor, Fast Scan and Flex Test —with some additional constructs for describing the memories' read/write cycles.

Test Types and Fault Models: A manufacturing defect is a physical problem that occurs during the manufacturing process, causing device malfunctions of some kind. The purpose of test generation is to create a set of test patterns that detects as many manufacturing defects as possible. For more information on test types and fault models, refer to the Scan and ATPG Process Guide. Memories fail in a number of different ways. The three major parts such as address decode logic, read / write logic and memory cell array, can have individual flaws that cause the device to fail.

Basic Fault Types: The basic types of memory faults include stuck-at, transition, coupling, and neighborhood pattern sensitive. The next several sections discuss each of these fault types in more detail.

a. Stuck-at Faults
b. Transition Faults
c. Coupling Faults
d. Inversion Coupling Fault
e. Idempotent Coupling Fault
f. Neighborhood Pattern Sensitive Faults

3-WRBIST: In this paper, we assume that the sequential CUT has 'm' primary and state inputs, and employs full-scan. Even though the proposed BIST TPG is applicable to scan designs with multiple scan chains, we assume that all 'm' primary and state inputs are driven by a single scan chain unless stated otherwise only for clarity and convenience of illustration. A test cube is a test pattern that has unspecified inputs. The detection probability of a fault is defined as the probability that a randomly generated test pattern detects the fault. In the 3-WRBIST scheme, fault coverage for a random pattern resistant circuit is enhanced by improving detection probabilities of RPRFs; the detection probability of an RPRF is improved by fixing some inputs of the CUT to the values specified in a deterministic test cube for the RPRF. A generator or weight set is a vector that represents weights that are assigned to inputs of the circuit during 3-WRBIST. Inputs that are assigned weight 1 (0) are fixed to 1 (0) and inputs that are

assigned weight 0.5 are driven by outputs of the pseudorandom pattern generator, such as an LFSR and a CA. A generator is calculated from a set of deterministic test cubes for RPRFs.

Consider a sequential circuit that has 'm' primary and state inputs. $C=\{c^0,c^1,c^2,.....c^{d-1}\}$ Denotes a set of test cubes for RPRFs in the CUT, $C^j=(c_0^j,c_1^j,c_2^j,.....c_{m-1}^j)$ where is an m–bit test cube, where $c_i^j\in\{0,1,X\}\forall i$, where 'X' is a don't care. Fig. 1 shows test cube set C, which consists of four test cubes, C^0, C^1, C^2 and C^3. Generator $gen(C)=\langle g_{0,}g_{1,}g_{2,}....g_{m-1}\rangle$ for the circuit with m inputs is denoted as an m -bit tuple, where $g_k\in\{0,1,X,U\}$ and k=0, 1, 2, 3….m-1.

If input p_k is assigned only a 1 (0) or an X in every test cube and assigned a 1 (0) at least in one test cube in C, then input p_k is assigned a 1 (0) in the generator, i.e., $g_k=1(0)$. If input p_k is assigned a 1 in a test cube C^a, i.e., $C^a{}_k=1$, and assigned a U in another test cube c^b, i.e., $c_k^b=0$, then input p_k is assigned a in the generator, i.e., $g_k=U$. Otherwise (p_k is always assigned an X in every generator), input p_k is assigned an in the generator, i.e., $g_k=X$. In summary, g_k is defined as follows:

g_k=1, if $c_k^j=1$ or X in $\forall c^j\in C$ and at least one $c_k^j=1$ 0, if $c_k^j=0$ or X in $\forall c^j\in C$ and at least one $c_k^j=0$ U, if $\exists c_k^a=1$ and $c_k^b=0$ where $c^a,c^b\in C$ X, otherwise (1)

Inputs that are assigned U's in a generator are called conflicting inputs of the generator.

In the test cube C shown in Fig. 3, input p_1 is always assigned only a 0 or an X in every test cube in C (0 in c^1 and c^3 and X in c^0 and c^2). Therefore, g_1 is assigned 0 in generator $gen(C)$. On the other hand, g_3 is assigned a 1 in generator $gen(C)$ since input p_3 is always assigned only a 1 r an X in every test cube (1 in C^0 and C^1 and X in C^2 and C^3). In contrast, since input p_0, which is assigned both 0 and 1 (0 in C^0 and 1 in C^2 and C^3), g_0= U in $gen(C)$.

C	p_5	p_4	p_3	p_2	p_1	p_0	F	dp
c^0	1	X	1	1	X	0	f^0	$1/2^4$
c^1	X	0	1	0	0	X	f^1	$1/2^4$
c^2	1	X	X	0	X	1	f^2	$1/2^3$
c^3	0	0	X	0	0	1	f^3	$1/2^5$
gen(C)	U	0	1	U	0	U		

(a)

C^0	p_5	p_4	p_3	p_2	p_1	p_0
c^0	1	X	1	1	X	0
c^1	X	0	1	0	0	X
$gen(C^0)$	1	0	1	U	0	0

C^1	p_5	p_4	p_3	p_2	p_1	p_0
c^2	1	X	X	0	X	1
c^3	0	0	X	0	0	1
$gen(C^1)$	U	0	X	0	0	1

(b)

Fig.3. Example Test Cube Sets (a) Test Cube Set C (b) Test Cube Set C^0C^1

	g_8	g_7	g_6	g_5	g_4	g_3	g_2	g_1	g_0
$gen(C^0)$	1	1	U	X	X	X	U	0	0
$gen(C^1)$	1	1	0	U	X	X	0	0	0
$gen(C^2)$	0	0	X	X	X	U	U	1	1

Fig.4. Example Generators (Weight Sets).

Assume that test cubes, C^0, C^1, C^2 and C^3, are only test cubes that detect faults f^0,f^1,f^2 and f^3, respectively. Under this assumption, the detection probability of fault f^j is simply given by ½ the number of binary values in c^j. If g_k =1(0), then input p_k can be fixed to a 1 (0) without making any fault F in untestable since input p_k is never assigned a 0 (1) in any test cube. Note that fixing input p_k to a 1 (0) increases the number of binary values by 1 in all test cubes. Hence, fixing p_k to a 1 (0) improves detection of faults which requires a 1 (0) at input p_k for their detection by a factor of 2. When g_k =U, e.g., g_0,g_2 and g_5 of Fig. 1(a), there are test cubes C in that conflict at input p_k, so that fixing input p_k to a binary value v, where $v\in\{1,0\}$, may make faults for which test cubes are assigned $\bar{v}$ at p_k undetectable.

If a circuit contains a large number of RPRFs, then test cubes for RPRFs may be assigned conflicting values in many inputs resulting in a generator where large number of inputs are assigned U's. Hence, only a few inputs can be fixed in such generators without making any fault untestable. If a circuit has a large number of RPRFs, then multiple generators, each of which is calculated from test cubes for a subset of RPRFs in the circuit, may be required to achieve high fault coverage with a reasonable length of random pattern sequence.

In this project, the generator for subset c^i is denoted by $gen(C^i)=\{g_0^i,g_1^i,g_2^i,...g_{m-1}^i\}$, where m is the number of inputs of the CUT. If we partition the four test cubes c^0,c^1,c^2 and c^3 shown in Fig. 1(a) into two groups

$C^0 = \{c^0, c^1\}$ and $C^1 = \{c^2, c^3\}$, then inputs p_0, p_1, p_3, p_4 and p_5 can, respectively, be fixed to 0, 0, 1, 0, and 1 in C^0 and inputs p_0, p_1, p_2 , and p_4 to 1, 0, 0, and 0 in C^1 without any conflict. As a consequence, $gen(C^0) = \{0,0,U,1,0,1\}$ and $gen(C^1) = \{1,0,0,X,0,U\}$. Note that numbers of conflicting inputs, i.e., U's, of both $gen(C^0)$ and $gen(C^1)$ are reduced to 1 and detection probabilities of all four faults f^0, f^1, f^2 and f^3 increase to ½.

Since test cubes are grouped into two subsets C^0 and C^1, the entire test session is also divided into two sub sessions. In the first sub session, inputs p_0, p_1, p_3, p_4 and p_5 are fixed to 0, 0, 1, 0, and 1, respectively, to generate M^0 test patterns by using $gen(C^0)$. In the second sub session, inputs p_0, p_1, p_2 and p_4 are fixed to 1, 0, 0, and 0, respectively, to generate M^1 test patterns by using $gen(C^1)$. In this project, the same M patterns are generated by using every generator $gen(C^i)$, where i= 1, 2,… ..to simplify hardware for the BIST controller. However, different numbers of test patterns can also be generated to reduce test application time at the expense of higher hardware overhead. Adequate numbers of test patterns should be generated by using each generator to detect all faults targeted by the generator. A probability-based procedure to compute a suitable time interval M^i for generator $gen(C^i)$ is described.

Architecture of 3-WRBIST: Fig. 3 shows a set of generators and Fig.5 shows an implementation of the 3-WRBIST for the generators shown in Fig. 3. The shift counter is an (m+1) modulo counter, where m is the number of scan elements in the scan chain (since the generators are 9 bits wide, the shift counter has $\lceil \log_2(9+1) \rceil = 4$ stages). When the content of the shift counter is k, where k= 0, 1,…..8, a value for input p_k is scanned into the input of scan chain. The generator counter selects appropriate generators; when the content of the generator counter is i, test patterns are generated by using $gen(C^i)$, where i= 0,1,2. Pseudo-random pattern sequences generated by an LFSR and a CA are modified (fixed) by controlling the AND and OR gates with overriding signals s_0 and s_1; fixing a random value to a 0 is achieved by setting s_0 to a 1 and s_1 to a 0 and fixing a random value to a 1 is achieved by setting s_1 to a 1 (since a random value can be fixed to a 1 by setting s_1 to a 1 independent of the state of s_0, the state of s_0 is a don't care). Overriding signals s_0 and s_1 are driven by the outputs of T-flip-flops, TF_0 and TF_1. The inputs of TF_0 and TF_1 are in turn driven by the outputs of the decoding logic D_0 and D_1, respectively, which are generated by the outputs of the shift counter and the generator counter as inputs.

The shift counter is required by all scan-based BIST techniques and not particular to the proposed 3-WRBIST scheme. All BIST controllers need a pattern counter that counts the number of test patterns applied. The generator counter can be implemented from $\lceil \log_2 G \rceil$ MSB (most significant bit) stages of the existing pattern counter, where G is the number of generators, and no additional hardware is required for the generator counter. Hence, hardware overhead for implementing a 3-WRBIST is incurred only by the decoding logic and the fixing logic, which includes two toggle flip-flops (flip-flops), an AND and an OR gate. Since the fixing logic can be implemented with very little hardware, overall hardware overhead for implementing the serial fixing 3-WRBIST is determined by hardware overhead for the decoding logic.

Consider generating test patterns by the TPG shown in Fig. 3. Assume that if $g_k^i = X (i = 0,1,2, and k = 0,1,2....8)$, both D_0 and D_1 are set to 0's and hence the T flip-flops hold their previous states in cycles when a scan value for input p_k is scanned in. Also assume that T flip-flop TF_0 is initialized to a 1 and T flip-flop TF_1 is initialized to a 0 before each scan shift operation starts. As shown in Fig. 3, scan flip-flops are placed in the scan chain in a descending order of their subscript numbers. Hence, the value for input is scanned in first and the value for input p_8 is scanned in last.

Test patterns are generated by using generator $gen(C^0)$ (GEN_COUNT = 0) first. Since $g_0^0 = 0$ (see Fig. 2) when a value for scan input p_0 is scanned in s_0, (the output of TF_0) should be set to a 1 to set the output of the AND gate to a 0 (to fix the value for scan input p_0 to a 0) and s_1 (the output of TF_1) should be set to a 0 to propagate the 0 at the output of the AND gate to the output of the OR gate. Since the initial state of TF_0 is a 1 and the initial state of TF_1 is a 0, both D_0 and D_1 are set to 0's and TF_0 and TF_1 hold their initial states. At the next scan shift cycle (SCAN_COUNT=1), since $g_1^0 = 0$, both D_0 and D_1 are assigned a 0 to make TF_0 and TF_1 hold their

previous states $TF_0 = 1$ and $TF_1 = 0$. Since $g_2^0 = U$, both $s_0 and s_1$ should be set to a 0 to scan in a random value, which is generated by the LFSR, for p_2. Hence, D_0 is set to a 1 to toggle the state of TF_0 to a 0 when (SCAN_COUNT=2). The next three inputs p_3, p_4, and p_5 are assigned X in generator $gen(C^0)$, i.e., $g_3^0 = g_4^0 = g_5^0 = X$. Hence, TF_0 and TF_1 hold their previous states, i.e. $TF_0 = TF_1 = 0$, for the next three cycles (SCAN_COUNT=3, 4, and 5). Since $g_6^0 = U$, both TF_0 and TF_1 hold their previous states when (SCAN_COUNT=6). Next, since $g_7^0 = 1$ is set to a 1 to fix the value for scan input p_7 to a 1. This requires D_1 to be set to a 1 to toggle the state of TF_1 to a 1. Finally, when (SCAN_COUNT=8), since $g_8^0 = 1$, both D_0 and D_1 are assigned a 0 and both TF_0 and TF_1 hold their previous states. This sequence is repeatedly generated at D_0 and D_1 until all test patterns are applied to the scan inputs. D_0 and D_1 values for generators $gen(C^1)$ and $gen(C^2)$ can be determined in similar manners.

Random patterns generated by the LFSR can be fixed by controlling the AND/OR gates directly by the decoding logic without the two flip-flops. However, this scheme will incur larger hardware overhead for the decoding logic and also cause more transitions in the CUT during BIST (see Section III) than the scheme with the flip-flops. Fig.3.4 (a) shows TF_0, and TF_1, D_0, D_1 values for the scheme with T flip-flops that is implemented for the three generators $gen(C^0), gen(C^1)$ and $gen(C^2)$ shown in Fig. 3.1. The column under the label I^- shows the initial states of TF_0 and TF_1. D_0 is assigned 1's at five locations (p_2 in $gen(C^0)$, p_5 and p_6 in $gen(C^1)$, and p_2 and p_a in $gen(C^2)$) and D_1 is assigned 1's at four locations (p_7 in $gen(C^0)$ p_7 in $gen(C^1)$, and p_0 and p_2 in $gen(C^2)$). Hence, the on-set of the function for D_0 has five min-terms and the on-set of the function for D_1 has four min-terms. Hardware overhead required to implement a decoding logic is in general determined by the number of min-terms in the on-set (or off-set) of the function of the decoding logic.

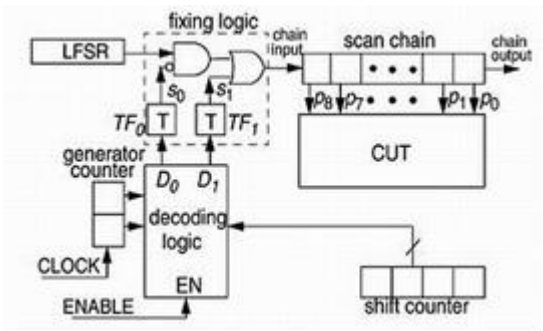


Fig.5. 3 Weight WRBIST

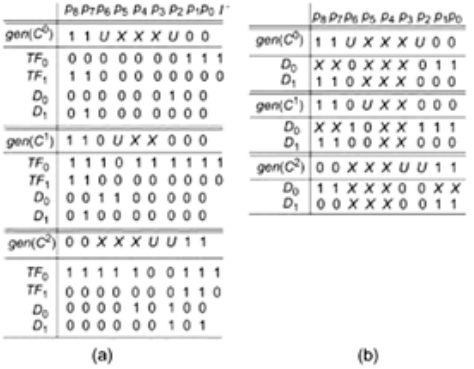


Fig.6. D_0 and D_1 values: (a) with Toggle Flip Flops TF_0 and TF_1 and (b) without Toggle Flip Flops

Now consider a version of 3-WRBIST that is implemented without the T flip-flops between the outputs of the decoding logic and the inputs of the AND and OR gate. D_0 and D_1 values for all three generators $gen(C^0), gen(C^1) and gen(C^2)$ are shown in Fig.6(b) for comparisons with the scheme with T flip-flops. For every input p_k that is assigned a 0 in a generator $gen(C^i)$, i.e., $g_k^i = 0$, its corresponding D_0 value should be a 1 and D_1 value be a 0. If p_k is assigned a 1 in a generator $gen(C^i)$, i.e., $g_k^i = 1$, then the corresponding D_1 value should be a 1 (the D_0 value is a don't care). The on-set (off-set) of the function for D_0 has eight (five) min-terms and the on-set (off-set) of the function for D_1 has six (13) min-terms. Hence, the function for the decoding logic of this scheme requires more min-terms in its on-set and off-set than that for the decoding logic of the scheme with T flip-flops. Significant reduction in the number of min-terms can be achieved by inserting T flip-flops for large designs.

TPG to Generate Suitable Test Cubes: Each generator is computed from a set of deterministic test cubes for RPRFs. A test cube set is dynamically constructed by continuously adding test cubes into the test cube set. The test cube set C^i, which is currently under construction, i.e., into which test cubes are currently added, are called the current test cube set. Test cubes are added into the current test cube set until placing any more test cube into C^i makes the number of conflicting inputs, i.e., U's, in the generator greater than a predefined threshold. Whenever a

test cube is placed into C^i, generator $gen(C^i)$ is updated according to (1). Upon the completion of generating a test cube set C^i, a new current test cube C^{i+1} set is created and the test cubes generated later are placed into the new test cube set C^{i+1}.

Since each generator requires a different sequence of control bits at the output of the decoding logic, hardware overhead for the decoding logic is determined also by the number of generators. A special automatic test pattern generation (ATPG) is used to generate deterministic test cubes for RPRFs that are suitable to minimize the number of generators. In order to minimize the number of generators (placing more test cubes into each test cube set will result in smaller number of generators), the proposed ATPG generates each test cube taking all test cubes existing in the current test cube set into consideration. At the heart of the ATPG technique are three cost functions: controllability, observability, and test generation cost function, which are obtained by modifying the traditional SCOAP-like cost functions. The test generation process of the ATPG, which is based on PODEM, is guided by the cost functions to generate suitable test cubes.

The controllability cost of input p_k, $C_{v(p_k)}$, is defined by considering the generator of the current test cube set as follows:

$C_{v(p_k)} =$ 0, if $g_k^i = U$

$C_{v(p_k)} =$ 0, if $g_k^i = v$

$C_{v(p_k)} =$ w, if $g_k^i = \bar{v}$ (where w >>1) (2)

$C_{v(p_k)} = 1$, if $g_k^i = X$ where v is a binary value 0 or 1.

The purpose of the controllability cost is to estimate the number of conflicting inputs that would be caused by adding into the current test cube set a test cube where input p_k is assigned a binary value v. If $g_k = U$, the current test cube set already contains test cubes that conflict at input p_k (at least one test cube in the current test cube is assigned a 1 at p_k and another test cube is assigned a 0 at p_k). Hence, assigning any binary value to p_k does not cause any more adverse effect. Hence, $C_{v(p_k)} = 0$. When $g_k = 1(0)$, adding a test cube whose input p_k is assigned a 1 (0) causes no conflict with any test cube in the current test cube set. Hence, $C_{v(p_k)} = 0$. If $g_k = 1(0)$, all test cubes existing in the current test cube set are assigned only 1 (0) or X at input p_k. Hence, adding a test cube whose input p_k is assigned the opposite value 0 (1) causes a conflict at input p_k with other test cubes in the current test cube set. Hence, $C_{v(p_k)} = w$ high cost is given. Finally, if $g_k = X$ (input p_k is not specified in any test cube in the current test cube set), adding a test cube that is assigned v at input p_k may increase one min-term in the on-set for the function of D_v. Hence, a small cost 1 is given.

The controllability costs for internal circuit line in the circuit, which are computed in a similar manner to the testability measures used, are given by if otherwise

$$C_v(l) = \text{Min}\{C_c(l_a)\}, \text{ if } v = c \oplus i \qquad C_v(l) = \sum_{l_a} C_{\bar{c}}(l_a), \quad \text{otherwise} \qquad (3)$$

where l_a and l are, respectively, the inputs and the output of a gate with controlling value c and inversion i. The controlling value of a gate is the binary value that, when applied to any input of a gate, determines the output value of that gate independent of the values applied at the other inputs of the gate. The controllability cost functions guide the ATPG to select the back trace paths that require the minimum cost (number of conflicting inputs), whenever there is a choice of several paths to back trace from the target line to the inputs.

The observability cost functions are recursively computed from primary outputs to primary inputs. The observability cost of line l is given by

O(l) = min $O(l_0)$ if l is a stem with branches l_0

O(l) = $\sum_{l_a} C_{\bar{c}}(l_a) + O(l_0)$ otherwise (4)

where in the latter case l_0 is the output of gate with input l and l_0 are all inputs of l_0 other than l. The observability cost functions are used to guide the objective selection.

The proposed ATPG will now be described for the stuck-at fault model. In order to generate a test cube to detect a stuck-at- $\bar{v}$ (s-a-$\bar{v}$) at line l, first the fault should be activated by setting line l to v. The cost to activate 1s -a - $\bar{v}$ is $C_v(l)$. Then, the activated fault effect should be propagated to one or more outputs. The cost to propagate

the activated fault effect at line l is 0(l). Hence, the test generation cost to generate a test cube for ls -a - $\bar{v}$ is defined as the sum of two cost functions

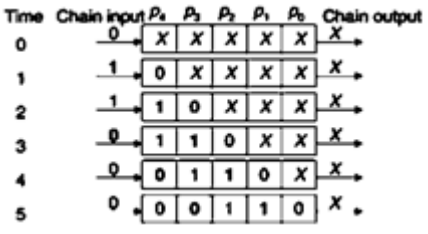
$$Tv(l) = C_v(l) + O(l) \quad (5)$$

The test generation cost is used to select a best target fault from the fault list. Since test cubes generated by the proposed ATPG are often over-specified, a few bits that are assigned binary values by the proposed ATPG can be relaxed to don't cares while ensuring the detection of targeted faults. Test cubes with fewer specified inputs have fewer conflicting inputs with test cubes already in the current test cube set so that more test cubes can be placed in the current test cube set. Whenever a test cube is generated, inputs that are assigned binary values are ordered according to the cost incurred by assigning each input to its binary value. The binary value assigned to each of these inputs is flipped in this order. If all the targeted faults can still be detected even after an input is flipped to its opposite value, the value assigned to the input is relaxed to a don't care.

If a circuit has any re-convergent fan out, an input assignment required to satisfy some objectives may conflict with that required to satisfy other objectives, causing the proposed ATPG to select an objective or back trace path with a high cost. Hence, in circuits with re-convergent fan outs, the actual cost of the test cube generated for a fault may be much higher than the cost of the fault given by the estimate test generation cost function shown. To prevent adding such test cubes to the current test cube set, if the actual cost of a generated test cube is higher by a certain number (say, 100) than the estimate test generation cost of the fault, the generated test cube is discarded. Test generation is then carried out for alternative target faults until a test cube is found for a fault whose actual cost is close to the estimate test generation cost of the fault or all faults in the fault list are tried. Even in the worst case where all faults in the fault list need to be tried, generating test cubes is required only for a few faults. The estimate test generation cost of a fault is always an optimistic approximate of the actual cost for the fault in the sense that the actual cost of any test cube for the fault cannot be less than the estimate test generation cost. Hence, if the estimate test generation cost of a fault is greater than the actual cost of the test cube that has the minimum actual cost among the test cubes that have been generated but discarded due to high actual cost, then the actual cost of any test cube for the fault cannot be smaller than the current minimum actual cost. Hence, we do not need to generate a test cube for the fault. If test cubes for all faults in the fault list have very high actual cost, then the test cube that has the minimum actual cost is chosen to be a new member of the current test cube set.

SWITCHING ACTIVITY

Minimizing Switching Activity During BIST: The BIST TPG proposed in this paper reduces switching activity in the CUT during BIST by reducing the number of transitions at scan inputs during scan shift cycles. The sequential CUT is implemented in CMOS, and employs full-scan. If scan input p_i is assigned v , where $v \in \{0,1\}$, at time t-1 and assigned the opposite value $\bar{v}$ at time t, then a transition occurs at p_i at time t. The transition that occurs at scan input p_i can propagate into internal circuit lines causing more transitions. During scan shift cycles, the response to the previous scan test pattern is also scanned out of the scan chain. Hence, transitions at scan inputs can be caused by both test patterns and responses. Since it is very difficult to generate test patterns by a random pattern generator that cause minimal number of transitions while they are scanned into the scan chain and whose responses also cause minimal number of transitions while they are scanned out of the scan chain, we focus on minimizing the number of transitions caused only by test patterns that are scanned in. Even though we focus on minimizing the number of transitions caused only by test patterns, our extensive experiments show that the proposed TPG can still reduce switching activity significantly during BIST. Since circuit responses typically have higher correlation among neighborhood scan outputs than test patterns, responses cause fewer transitions than test patterns while being scanned out.



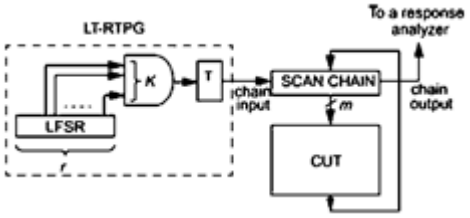


Fig.7. Transition at Scan Chain Input **Fig.8. LT-RTPG**

A transition of scan shift cycle t, which has the input of the scan chain that is caused by scanning in a value. As it is opposite to the value which was scanned in at the last scan shift cycle t-1, causes transitions continuously at scan inputs and the value travels through the scan chain for the next scan shift cycles. Fig. 4.1 describes scanning a scan test pattern 01100 into a scan chain that has five scan flip-flops. Since a 0 is scanned into

the scan chain at time t=0, the 1 at time t=1 causes a transition at the input of the scan chain. That is continuously generate transitions at the scan flip-flops it passes through until it arrives at its final destination p_1 at time t=5. But, 1 that is scanned into the scan chain at cycle t=2 causes no transition at the input of the scan chain and arrives at its final destination p_2 at time t=5 without causing any transition at the scan flip-flops it passes through. This shows that transitions that occur in the entire scan chain can be reduced by reducing transitions at the input of the scan chain. Since transitions at scan inputs propagate into internal circuit lines causing more transitions, reducing transitions at the input of scan chain can eventually reduce switching activity in the entire circuit.

LT-RTPG: The LT-RTPG proposed, reduces switching activity during BIST by reducing transitions at scan inputs during scan shift operations. An example LT-RTPG is shown in Fig. 4.2. The LT-RTPG is comprised of an -stage LFSR, a -input AND gate, and a toggle flip-flop (T flip-flop). The AND gate is connected to output of the LFSR stages. If large K is used, large sets of neighboring state inputs will be assigned identical values in most test patterns, resulting in the decrease fault coverage or the increase in test sequence length. Hence, in this project, LT-RTPGs with only K=2 or 3 are used. Since a T flip-flop holds previous values until the input of the T flip-flop is assigned a 1, the same value v, where $v \in \{0,1\}$, is repeatedly scanned until the output of the AND gate value becomes 1. Hence, during scan shift operations, the adjacent scan flip-flops are assigned identical values in almost all test patterns and scan inputs have fewer transitions. Since most switching activity during scan BIST occurs during scan shift operations (a capture cycle occurs at every m+1 cycles), the LT-RTPG can reduce heat dissipation during overall scan testing. Various properties of the LT-RTPG are studied and a detailed methodology for its design is presented.

It has been observed that many faults that escape random patterns are highly correlated with each other and can be detected by continuously complementing values of a few inputs from a parent test vector. This observation is exploited to improve fault coverage for circuits that have large numbers of RPRFs. We have also observed that tests for faults that escape LT-RTPG test sequences share many common input assignments. This implies that RPRFs that escape LT-RTPG test sequences can be effectively detected by fixing selected inputs to binary values specified in deterministic test cubes for these RPRFs and applying random patterns to the rest of inputs. This technique is used in the 3-WRBIST to achieve high fault coverage for random pattern resistant circuits. In this paper we investigate that augmenting the LT-RTPG with the serial fixing 3-WRBIST proposed, can attain high fault coverage without excessive switching activity or large area overhead even for circuits that have large numbers of RPRFs.

Property of 3-WRBIST to Reduce Switching Activity: If a large set of scan inputs that are consecutively located in the scan chain are assigned identical values (X is identical to any binary value 0 or 1) in a generator, then the T flip-flops TF_0 and TF_1 of 3-WRBIST (see Fig. 3.3) stay at the same state for many scan shift cycles. While TF_1 holds a 1, the output of the OR gate in the fixing logic is set to a 1 and 1's are continuously scanned into the scan chain and no transitions occur at the input of the scan chain. Likewise, while TF_0 holds a 1, random pattern values generated by the LFSR are blocked at the AND gate and no transition occurs at the input of scan chain provided that the other T flip-flop TF_1 does not toggle. Hence, in order to significantly reduce the number of transitions at the input of scan chain, either TF_0 or TF_1 should be assigned a 1and stays at the 1 for long periods of scan shift cycles.

Typically, the majority of scan inputs are assigned X's (don't cares) in every generator. Since all the faults that are targeted by a generator can be detected independent of the scan values applied to the scan inputs that are assigned X's in the generator, the values of D_0 and D_1 for those scan inputs are assigned such that the number of minterms in the on-sets of the functions for D_0 and D_1 is minimized (to minimize hardware overhead for the decoding logic) and either TF_0 or TF_1 stays at 1 for long periods of scan shift cycles (to minimize the number of transitions at the input of the scan chain). In order to minimize the number of minterms in the on-sets of functions for D_0 and D_1, D_0 and D_1 for the scan inputs that are assigned X's in a generator should be assigned 0. Note that $TF_0(TF_1)$ does not toggle (holds its previous state) when the state of $D_0(D_1)$ is 0.

Assume that test patterns are currently generated by generator $gen(C^{i})$. Also, assume that scan input p_a is assigned an X while its predecessor p_{a-1} is assigned a care bit (0, 1, or U) in $gen(C^{i})$. Let p_b be the first scan input in the scan chain that is assigned a care bit $gen(C^{i})$ in after p_a, i.e. b>a, In other words, consecutive scan

inputs p_a, $p_{a+1,}$,..... p_{b-1} which are located between $p_{a+1,}$ and p_b, are assigned X's in $gen(C^{i)}$. $TF_0(k) and TF_1(k)$, respectively, denote the states of TF_0 and TF_1 in a scan shift cycle when a value for scan input p_k is scanned into the scan chain. In this project, the decoding logic is designed such that the states of TF_0 and for the scan inputs $p_{a+1,}p_{a+2,}.....p_{b-1}$ are always the same as those of TF_0 and TF_1 for input p_a, i.e., D_0 and D_1values for the inputs $p_{a+1,}p_{a+2,}.....p_{b-1}$ are always 0 to minimize the number of minterms in the on-sets of the function for the decoding logic.

D_0 and D_1 values for input p_a are determined by considering values assigned at p_a and p_b in generator $gen(C^i)$. If either $TF_0(a-1) or TF_1(a-1)$ is 1, i.e., $p_{a-1,}$ is assigned a binary value 0 or 1 in $gen(C^{i)}$, then both D_0 and $D_1$for p_a are assigned 0 to minimize the number of min-terms in the on-sets of the decoding logic function. Note that this minimizes also the number of transitions since when either $TF_0 or TF$ is set to 1 for scan shift cycles for inputs $p_a, p_{a+1,}p_{a+2,}.....p_{b-1}$, no transitions occur during the scan shift cycles for these scan inputs. On the other hand, if both $TF_0(a-1) and TF_1(a-1)$ are 0, i.e., $g_{a-1}^i = U$, then we check g_b^i, i.e., the value assigned at p_b in $gen(C^{i)}$, to determine the values of D_0 and D_1 for scan input p_a (recall that both $D_0 and D_1$ values for scan inputs $p_{a+1,}p_{a+2,}.....p_{b-1}$ are always assigned 0). If $g_b^i = 1(0)$, then we assign $D_1(D_0)$ for input p_a to 1 to make $TF_0(a)$ =0 and $TF_1(a)$=1, ($TF_1(a)$=0 and $TF_0(a)$ =1). This adds one min-term in the on-set of $D_1(D_0)$ for all the scan inputs $p_a, p_{a+1,}p_{a+2,}.....p_{b-1}$. In this case, a transition can occur at the input of the scan chain only in the scan shift cycles when a value for scan input p_a is scanned in among all scan shift cycles for inputs $p_a, p_{a+1,}p_{a+2,}.....p_{b-1}$. As the last case, if both $g_{a-1}^i = U$ and g_b^i, i.e., both inputs that flank the consecutive scan inputs $p_a, p_{a+1,}p_{a+2,}.....p_b$ are assigned U in $gen(C^i)$, then we determine the values of D_0 and D_1 for scan input p_a based on the number of scan inputs between p_b and p_a, i.e., b-a. If b-a >R, where R is a predefined natural number, then we arbitrarily select either $D_0 or D_1$ and assign it to a 1 for input p_a to suppress transitions at the input of scan chains. Otherwise, we assign both D_0 and D_1 for scan input p_a to 0 to minimize the number of min-terms in the on-sets of functions for $D_0 and D_1$. If b-a is large, then transitions can occur at the input of the scan chain in many scan shift cycles. Hence adding one more min-term in the on-set of functions for the decoding logic to suppress large number of transitions is worthy.

For example, consider the set of generators shown in Fig. 4(a). Consecutive scan inputs $p_4, p_5 and p_6$ are assigned X's and input p_3, which proceeds p_4 in the scan chain, is assigned a U in $gen(C^2)$. Since p_3 is assigned a U and hence both $TF_0(3) = TF_1(3) = 0$, we check input p_7, which is the first scan input that is assigned a care bit (0, 1, or U) in $gen(C^2)$ after the consecutive scan inputs $p_4, p_5 and p_6$, which are assigned X's in $gen(C^2)$. Since p_7 is assigned a 0 in $gen(C^2)$, D_0 for p_4 is assigned a 1 to toggle the state of TF_0 in the scan shift cycles for input p_4. On the other hand, in generator $gen(C^2)$, consecutive inputs $p_3, p_4, and p_5$ are assigned X's and inputs $p_2 and p_6$ which flank the inputs $p_3, p_4, and p_5$ are assigned both U. Hence, the D_0 and D_1 values for the scan input p_3 are determined by considering the predefined number R and the number of consecutive scan inputs that are assigned X's between $p_2 and p_6$, i.e., 3. If R<3 is used, then D_0 for input p_3 is assigned a 1 to toggle the state of TF_0 to 1 to suppress transitions. If R≥3 is used, then both D_0 and D_1 for input p_3 are assigned 0's to minimize the number of minterms in the on-sets of functions for the decoding logic.

Test Pattern Generator: The proposed BIST is consists of two TPGs an LT-RTPG and a 3-WRBIST (Fig.9). The multiplexer, which drives the input of scan chain, selects a test pattern source between the LT-RTPG and the 3-WRBIST. In the first test session, test patterns generated by the LT-RTPG are selected and scanned into the scan chain to detect easy-to-detect faults. In the second session, test patterns that are generated by the 3-WRBIST are selected to detect the faults that remain undetected after the first session. Considering the fact that an LT-RTPG can be implemented with very little hardware overhead (only one flip-flop and one AND gate in addition to an LFSR),

overall hardware overhead to implement the proposed TPG is determined by hardware overhead for the decoding logic of the 3-WRBIST.

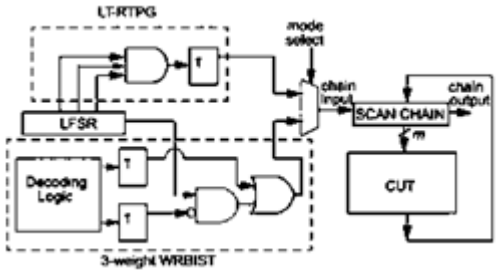


Fig.9. BIST TPG

An outline of the overall procedure to design an optimized BIST TPG by the proposed method is described in the following:

1) Apply a sequence of test patterns generated by the LT-RTPG to the circuit and drop all detected faults.

2) i ← 0

3) Initialize the current test cube set, $C^i \leftarrow \Phi$, and generator $gen(C^i) \leftarrow${X, X, X,…..X}, j←0 and unmark all faults in the fault list.

4) If there are no more faults in the fault list, then exit. Select an unmarked fault f^j that has the minimum test generation cost and generate a test cube c^j for the fault by the proposed ATPG.

5) Add the test cube to the current test cube set, $C^i \leftarrow C^i \cup c^j . j+1$.

6) Update generator $gen(C^i)$ according to the definition described in Section II-A. If the number of conflicting inputs is smaller than or equal to $U_{\max}$ ($U_{\max}$ is a positive integer, then mark all faults detected by test cube c^j and go to Step 4).

7) $C^i \leftarrow C^i - c^j$ update generator $gen(C^i)$ and $2^{U_{\max}}$ generate 3-WRBIST patterns by using $gen(C^i)$. Run fault simulation to drop the faults that are detected by the generated 3-WRBIST patterns. i← i+1 and go to Step 3.

RESULTS AND DISCUSSION

TABLE.1 : COMPARISON WITH LFSR GENERATED PATTERN

	LFSR			Proposed														
					K = 2							K = 3						
CKT Name	# pat	stuck at FE %	# Aver. Trans.	# LT-RTG pat	LT-RTG FE%	# gen	tot. # pat.	Switching LT-RTG	Switching 3W-TPG	Trans dly FE%	run time (sec)	LT-RTG FE%	# gen	tot. # pat.	Switching LT-RTG	Switching 3W-TPG	Trans dly FE%	run time (sec)
s1423	4096	99.08	339	2048	97.94	2	2304	.64	.72	99.75	4	96.04	4	2560	.45	.76	99.75	4
s5378	32768	99.75	1221	16384	96.13	4	20480	.73	.59	99.77	4	93.68	7	23552	.58	.69	99.50	5
s9234	132072	95.56	2717	16384	86.14	15	31744	.61	.49	99.55	627	82.35	18	34816	.38	.51	99.59	280
s13207	132072	99.76	4585	16384	89.27	9	25600	.73	.43	99.97	37	84.40	11	27648	.55	.41	99.91	83
s15850	528288	97.63	4773	16384	93.49	9	25600	.65	.54	99.72	165	90.79	14	30720	.44	.47	99.72	182
s38417	528288	99.31	12476	65536	93.86	29	124928	.67	.47	99.82	1399	93.65	31	129024	.44	.41	99.92	1557
s38584	528288	99.18	11211	32768	93.51	9	51200	.64	.45	99.96	601	91.49	16	65536	.41	.54	99.96	772
average	269410	98.61			92.91		40265	.67	.55	99.79		90.34		44857	.46	.54	99.76	

Table.1. compares results obtained by applying test sequences generated by regular LFSRs (LFSR sequences, for short) and results obtained by applying test sequences generated by the proposed TPGs (the proposed TPG sequences, for short). Columns under the heading LFSR give results of LFSR sequences while columns under the heading proposed give results of proposed TPG sequences. The column # pat under the heading LFSR shows the number of test patterns generated by the LFSR and the column FE% shows fault efficiency achieved by the LFSR sequence. Fault efficiency is defined as 100 X the number of detected faults /) the number of total faults the number of untestable faults. The average number of transitions per cycle is given in the column # Aver. Trans. This number includes the number of transitions caused not only by test patterns being scanned in but also by responses being scanned out. The number of transitions at signal lines are weighted by the number of fan-out in each stem. Hence, if a fan out stem that drives ‘a’ branches has a transition, then the transition is counted as ’a’ rather than 1. Total CPU time spent to synthesize the proposed TPG is shown in the columns run time (sec) in seconds.

For each circuit, we implemented two different proposed TPGs each of which is comprised of a different LT-RTPGs: one comprised of 2-input AND gate and the other comprised of 3-input AND gate (Fig.9.). Results

obtained by the proposed TPG with a 2-input (3-input) AND gate LT-RTPG are shown in the columns under the heading K=2. The same number of test patterns were generated by both 2-input AND gate LT-RTPG and 3-input AND gate LT-RTPG for each circuit (to drop easy-to-detect faults). The number of test patterns generated by the LT-RTPG is given in the column LT-RTG # pat. Fault efficiency achieved by the LT-RTPG sequence is shown in the columns LT-RTG FE%.

100% single stuck-at fault efficiency was achieved by the proposed TPG sequence for every benchmark circuit. In addition to stuck-at fault efficiency, we computed transition delay fault efficiency achieved by the same sequence of test patterns. Transition delay fault efficiency is given in the columns Trans dly FE%. To compute transition delay fault efficiency, test patterns generated by the proposed TPG were applied to scan chains by the skewed-load approach (or the launch-off-shift). It is interesting to see that even though the proposed ATPG is not designed for transition delay fault model, transition delay fault efficiency achieved by the proposed TPG is very close to 100% for every circuit. Columns tot. # pat. Give the total number of test patterns, which includes both LT-RTPG patterns and 3-WRBIST patterns, generated by the proposed TPG to achieve 100% stuck-at fault efficiency.

The columns # gen show the number of generators that were generated by the proposed ATPG. The results on the number of test patterns clearly demonstrate that the proposed TPG can achieve high fault coverage with significantly fewer test patterns than the LFSR. The average number of transitions per test cycle is shown in the columns switching as a fraction of average number of transitions caused by LFSR sequences. Both LT-RTPG and 3-WRBIST test sequences significantly reduced switching activity in every circuit. Test sequences that are generated by the 2-input (3-input) AND gate LT-RTPG caused on average 33% (54%) fewer transitions than LFSR generated test sequences. The 3-WRBIST generated sequences also significantly reduced the number of transitions (by average 47%). Note that larger reduction in the number of transitions is achieved for large circuits. These results clearly demonstrate that the proposed TPG can achieve high fault coverage and also efficiently reduce excessive switching activity that may occur during BIST. Hardware overhead for the proposed TPG is presented in Table II.

Table.2. compares the proposed method with recent prior work. Like the proposed TPG, the TPGs proposed achieve 100% fault efficiency for every ISCAS'89 benchmark circuit.

Table. 2: Comparison with Prior Work

CKT	*Proposed (K = 2)			[36]		[12]		[37]			[38]		
Name	# pat	GE	% AP Red	# pat	GE	# pat	GE	# pat	% AP Red	FE %	# pat	% AP Red	FC %
s5378	20.5K	111.0	33	14K	103.5	-	-	-	-	-	-	-	-
s9234	31.7K	408.5	48	46K	748.5	21.3K	181.5	72.8K	59	95.47	37.0K	32	85.64
s13207	25.6K	344.5	46	18K	157.5	20.4K	283.5	40.7K	53	97.84	54.5K	36	96.80
s15850	25.6K	285.5	39	33K	478.5	13.9K	327.5	37.8K	58	97.69	29.5K	31	91.71
s38417	124.9K	1203.5	48	138K	2680.5	48.5K	1011.5	82.0K	61	95.96	80.5K	44	95.37
s38584	51.2K	398.5	47	21K	240.0	33.2K	382.5	82.1K	61	99.23	49.0K	22	95.24

*100 % fault efficiency is achieved by the proposed TPG for every circuit

Table.3: Experimental design on industrial design (3WEIGHT WRBIST)

Name	# FFs	# gates	grid cnt	# Init pat	Init FC%	# 3W pat	Final FC%	U	3-W Area (%)
D1	63292	876K	4006479	32K	88.93	32K	94.46	11	33655(0.84)
D1	63292	876K	4006479	512K	92.33	130K	98.03	11	48478(1.21)
D2	13960	244K	1480326	64K	92.14	100K	99.37	10	32129(2.17)
D3	753	6.9K	65323	1K	84.38	1K	100	3	3185 (4.88)

The columns % AP reduction show reduction in the average number of transitions against regular LFSRs. Fault efficiency and coverage achieved are shown in the columns FE% and FC%. Table.3. shows experimental results for three industrial designs. The column grid cnt gives the size of each design in the number of grids (the grid count does not include area occupied by embedded memories) and the column # FFs gives the number of scan flip-flops in the design. The column # Init pat gives the number of test patterns that were generated to detect easy-to-detect faults and the column In it FC % gives fault coverage achieved by the initial test sequence. The number of test patterns generated by the 3-weight BIST to detect the faults remaining undetected by the initial test sequence is reported in the column # 3W pat and final fault coverage achieved is reported in the column Final FC%. The column is the maximum number of conflicting bits allowed in each test pattern. The column 3-W Area reports area overhead for the 3-weight BIST in grid count and also in percentage to the grind count of benchmark circuit (shown in the parenthesis in the same column). Area overhead for 3-WRBIST for large designs such as and is only about 2% or even less. Note that the grid count reported in the column grid cnt does not include area occupied by embedded memories. Hence real area overhead for 3-WRBIST will be even less. The experimental results clearly show that the proposed TPG can be implemented for large industrial designs with low hardware overhead. Since a simulation tool that can calculate the number of transitions during scan testing for large industrial designs is currently not available to us, the number of transitions is not reported.

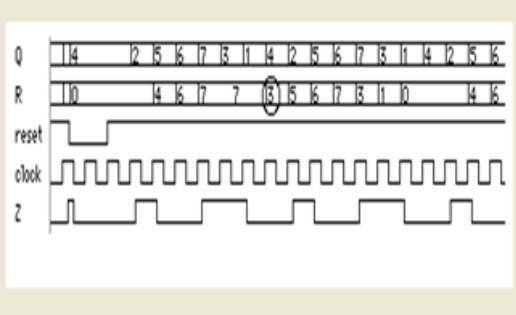


Fig.10. BIST (WITHOUT FAULT)

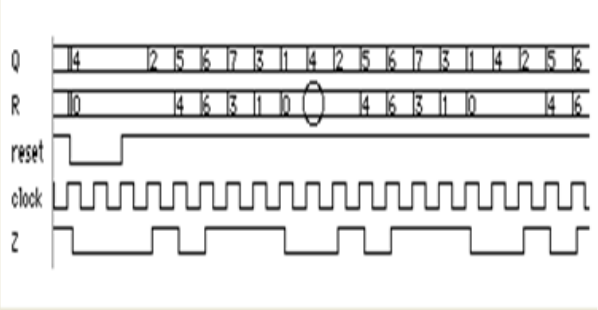


Fig.11. BIST (WITH FAULT)

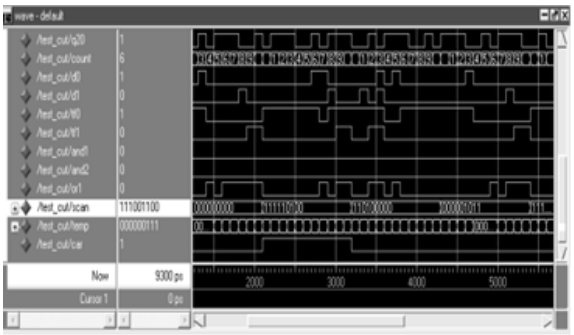


Fig.12. 3WRBIST SWITCHING ACTIVITY

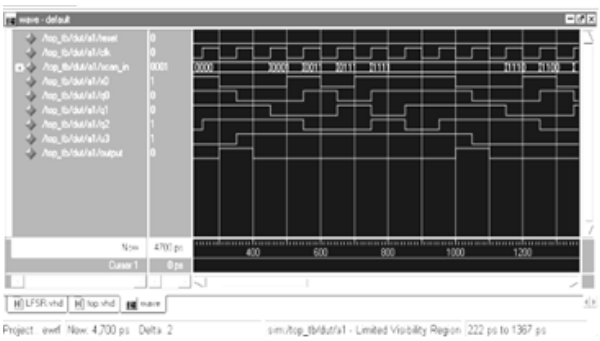


Fig.13. LT_RTPG

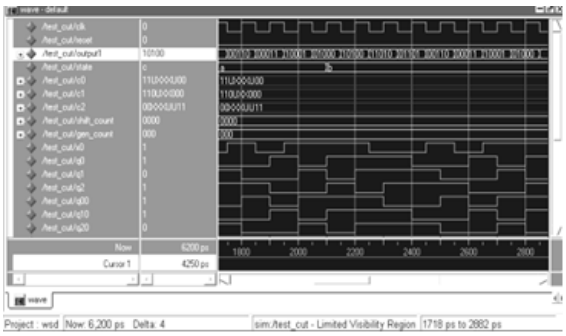


Fig.14. 3-WRBIST

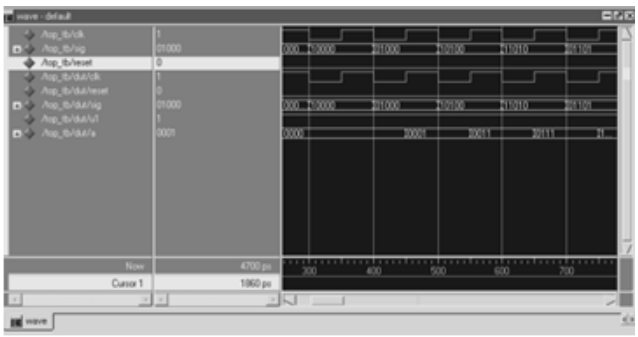


Fig.15. LT_RTPG

CONCLUSIONS

This paper presents, during BIST, a SC-BIST of low hardware overhead TPG which can reduce switching activity in CUTs. This also with a recommendable length of test sequence achieve very high fault coverage. Unacceptably long test sequences are often required to attain high fault coverage with pseudorandom test patterns for circuits that have many random pattern resistant faults. The main objective of advanced BIST techniques has been the design of TPGs which can achieve high fault coverage at highly appreciable test lengths of such circuits. While this objective still remains important, reducing heat dissipation during test application is also becoming an important objective. Excessive switching activity during test application can cause several problems. The proposed TPG reduces the number of transitions that occur at scan inputs during scan shifting by scanning in the test patterns where neighboring bits are highly correlated.

REFERENCES

C.Y. Tsui , M. Pedram, C.-A. Chen and A. M. Despain, Low power state assignment targeting two-and multi-level logic implementation, Proc. IEEE Int. Conf. Comput.-Aided Des., 1994, 82 -87.

Design of A, 7490 Like Decade Counter Integrated Circuit,Using Gaas Mesfet Dcfl Family, For Requencies Up To 1 Ghz. 2006

Deyhimy, Ira: Gallium Arsenide Joins the Giants, IEEE Spectrum, February 1995.

E. M. Sentovich , K. J. Singh , L. Lavagno , C. Moon , R. Murgai, A. Saklanha , H. Savoj , P. R. Stephan , R. K. Brayton and A. Sangiovanni-Vincentelli SIS: A system for sequential circuit synthesis, 1992.

F. Corno , M. Rebaudengo , M. S. Reorda , G. Squillero and M. Violante, Low power BIST via non-linear hybrid celluar automata, Proc. VLSI Test. Symp., 2000, 29 -34.

H.-C. Tsai , K.-T. Cheng, C.-J. Lin and S. Bhawmik, Efficient test-point selection for scan-based BIST, IEEE Trans. Very Large Scale Integr. (VLSI) Syst., 6(4), 1998, 667 -676.

Huber, John P. & Rosneck, Mark W, Successful ASIC Design the First Time Through, Van Nostrand Reinhold, 1991.

I. Pomeranz and S. Reddy, 3-weight pseudo-random test generation based on a deterministic test set for combinational and sequential circuits, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 2, 1993, 1050 -1058.

J. Hartmann and G. Kemnitz, How to do weighted random testing for BIST, Proc. IEEE Int. Conf. Comput.-Aided Design, 1993, 568 -571.

J. Savir, Skewed-load transition test: Part I, calculus, Proc. IEEE Int. Test Conf., 1992, 705 -713.

J. Waicukauski , E. Lindbloom, E. Eichelberger and O. Forlenza, A method for generating weighted random test patterns, IEEE Trans. Comput., 33(2), 1989, 149 -161.

K.-H. Tsai, J. Rajski and M. Marek-Sadowska, Star test: The theory and its applications, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 19(9), 2000, 1052 -1064.

L. H. Goldstein and E. L. Thigpen, SCOAP: Sandia controllability/observability analysis program, Proc. IEEE-ACM Des. Autom. Conf., 1980, 190 -196.

L. Li and K. Chakrabarty, Test set embedding for deterministic BIST using a reconfigurable interconnect network, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 23(9), 2004, 1289 -1305.

Long, Stephen I. & Butner, Steven E, Gallium Arsenide Digital Integrated Circuit Design, McGraw-Hill, 1990.

M. Chatterjee and D. K. Pradhan, A new pattern biasing technique for BIST, Proc. VLSITS, 1995, 417 -425.

N. A. Touba and E. J. McCluskey, Altering a pseudo-random bit sequence for scan-based BIST, Proc. IEEE Int. Test Conf., 1996, 167 -175.

N. H. E. Weste and K. Eshraghian Principles of CMOS VLSI Design: A Systems Perspective, 1992 :Addison-Wesley

N. Tamarapalli and J. Rajski, Constructive multi-phase test point insertion for scan-based BIST, Proc. IEEE Int. Test Conf., 1996, 649 -658.

N. Z. Basturkmen , S. M. Reddy and I. Pomeranz, A low power pseudo-random BIST technique, J. Electron. Test.: Theory Appl., 19(6), 2003, 637 -644.

N. Z. Basturkmen , S. M. Reddy and I. Pomeranz, Pseudo random patterns using markov sources for scan BIST, Proc. IEEE Int. Test Conf., 2002, 1013 -1021.

N. Zacharia , J. Rajski and J. Tyszer, Decompression of test data using variable-length seed LFSRs", Proc. IEEE 13th VLSI Test Symp, 1995, 426 -433.

N.C. Lai , S.-J. Wang and Y.H. Fu, Low-power BIST with a smoother and scan-chain reorder under optimal cluster size, IEEE Trans. Comput, Aided Des. Integr. Circuits Syst, 25(11), 2006, 2586 -2594.

P. Girard , L. Guiller , C. Landrault and S. Pravossoudovitch, A test vector inhibiting technique for low energy BIST design, Proc. VLSI Test. Symp., 1999, 407 -412.

P. Goel, An implicit enumeration algorithm to generate tests for combinational logic circuits, IEEE Trans. Comput., C-30, 3, 1981, 215 -222.

R. M. Chou , K. K. Saluja and V. D. Agrawal, Scheduling tests for VLSI systems under power constraints, IEEE Trans. Very Large Scale Integr. (VLSI) Syst, 5(2), 1997, 175 -185.

S. Gerstendorfer and H.-J. Wunderlich, Minimized power consumption for scan-based BIST, Proc. IEEE Int. Test Conf., 1999, 77 -84.

S. Hellebrand , J. Rajski , S. Tarnick, S. Venkataraman and B. Courtois, Built-In test for circuits with scan based on reseeding of multiple-polynomial linear feedback shift registers, IEEE Trans. Comput., 44(2), 1995, 223 -233.

S. Hellebrand , S. Tarnick and J. Rajski, Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift registers, Proc. IEEE Int. Test Conf., 1992, 120 -129.

S. W. Golomb Shift Register Sequences, 1982 :Aegean Park

S. Wang Minimizing Heat Dissipation During Test Application, 1998

S. Wang and S. K. Gupta, DS-LFSR: A BIST TPG for low heat dissipation, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 21(7|), 2002, 842 -851.

S. Wang and S. K. Gupta, LT-RTPG: A new test-per-scan BIST TPG for low heat dissipation, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 25(8), 2006, 1565 -1574.

S. Wang, Generation of low power dissipation and high fault coverage patterns for scan-based BIST, Proc. IEEE Int. Test Conf., 2002, 834 -843.

S. Wang, Low hardware overhead scan based 3-weight weighted random BIST, Proc. IEEE Int. Test Conf., 2001, 868 -877.

T. H. Cormen , C. E. Leiserson and R. L. Rivest Introduction to Algorithm, 1990 :MIT Press

T. Schuele and A. P. Stroele, Test scheduling for minimal energy consumption under power constrainits, Proc. VLSI Test. Symp., 2001, 312 -318.

Texas Instruments Inc, Designing with TTL Integrated Circuits, McGraw-Hill, 1971.

Texas Instruments Inc, The TTL Databook, 1980.

V. Dabholkar , S. Chakravarty , I. Pomeranz and S. Reddy, Techniques for minimizing power dissipation in scan and combinational circuits during test application, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 17(12), 1998, 1325 -1333.

Vitesse Semiconductor Corp, Foundry Design Manual, Version 6.0, 1993.

W. Li , C. Yu , S. M. Reddy and I. Pomeranz, A scan BIST generation method using a markov source and partial BIST bit-fixing, Proc. IEEE-ACM Design Autom. Conf, 2003, 554 -559.

Y. Savaria , B. Lague and B. Kaminska, A pragmatic approach to the design of self-testing circuits, Proc. IEEE Int. Test Conf, 1989, 745 -754.

Y. Zorian, A distributed BIST control scheme for complex VLSI devices, Proc. VLSI Testing Symp., 1993, 4 -9.