# Scan-BIST Based on Transition Probabilities

Irith Pomeranz*
School of ECE
Purdue University
W. Lafayette, IN 47907

## Abstract

We demonstrate that it is possible to generate a deterministic test set that detects all the detectable single stuck-at faults in a full-scan circuit such that each test contains a small number of transitions from 0 to 1 or from 1 to 0 when considering consecutive input values. Using this result we show that built-in test-pattern generation for scan circuits can be based on transition probabilities instead of probabilities of specific bits in the test set being 0 or 1. The resulting approach associates only two parameters with every set of test vectors: an initial value and a transition probability. We demonstrate that this approach is effective in detecting all the detectable single stuck-at faults in benchmark circuits.

**Categories & Subject Descriptors:** B.8.1 Reliability, Testing, and Fault-Tolerance

**General Terms:** Reliability

**Keywords:** built-in self-test, scan design.

## 1. Introduction

The number of transitions from 0 to 1 or from 1 to 0 along an $n$-bit test vector is the number of positions $i$, where $0 \leq i < n-1$, such that the value at position $i+1$ is different from the value at position $i$ of the test vector. For example, the test vector 00001111 has one transition at position 4 while the test vector 01100010 has four transitions at positions 1, 3, 6 and 7. We study the numbers of transitions along tests of deterministic test sets that achieve complete fault coverage of single stuck-at faults (i.e., the test sets detect all the detectable single stuck-at faults). We show that it is possible to generate test sets that achieve complete fault coverage and have very small numbers of transitions from 0 to 1 or from 1 to 0. We then go on to show that a built-in test-pattern generator (*TPG*) programmed to use a small number of different transition probabilities and initial values is able to achieve complete fault coverage for full-scan circuits. We demonstrate this result for benchmark circuits. These results are important for the following reasons.

When a *TPG* drives a scan chain, values of consecutive inputs are generated by the *TPG* in consecutive clock cycles. Thus, it is natural to allow the current value to determine the next value. A transition probability of $p$ implies that if the current value is $\alpha$, with probability $p$ the next value will be $\alpha'$. A test set with a transition probability of $p$ can be implemented as shown in Figure 1. The source of random values in Figure 1 produces a sequence where the probability of a 1 is $p$. This source drives a $T$ flip-flop. The output of the $T$ flip-flop will change whenever the random source produces the value 1. This happens with probability $p$, as required. We design a *TPG* by selecting values for $p$ and corresponding initial values $a$ for the $T$ flip-flop of Figure 1 in order to achieve complete fault coverage for benchmark circuits.

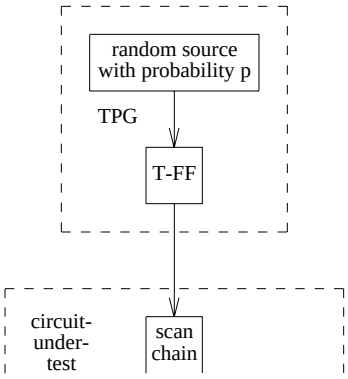


**Figure 1: A *TPG* based on transition probabilities**

Most of the existing test-pattern generation schemes attempt to control input values, and not transitions [1]-[17]. For example, weighted-random pattern generation methods assign a weight to every input, which indicates the probability that the input will be assigned the value 1. The bit-fixing logic in [11] also assigns values to specific bits in order to reproduce deterministic test vectors as part of a pseudo-random sequence. Our results demonstrate that an effective built-in test-pattern generation scheme can control the number of transitions along the tests it generates (or the transition probabilities of these tests) instead of controlling specific values (or the probability that a given value is 1). This implies that a single transition probability and an initial value are sufficient for generating effective test vectors. This alleviates the need to use a weight for every input under weighted-random pattern generation.

A two-state Markov source used in [17] as a *TPG* has a structure similar to Figure 1. Moreover, in extensions of the method of [17], transition probabilities are used for determining the state-transition probabilities of a finite-state machine implementing a Markov source [18]. However, the method of [17] and its extensions require additional logic, called bit-fixing logic, in order to adjust the input probabilities and achieve complete fault coverage. In addition, the scan chain is partitioned into virtual scan chains each with its own set of state-transition probabilities. Thus, these methods use additional logic to drive different virtual scan chains from different sources. The proposed method is simpler in that it does not require partitioning of the scan chain

* Research supported in part by NSF Grant No. CCR-0098091 and in part by SRC Grant No. 2001-TJ-950.


*DAC'04*, June 7-11, 2004, San Diego, California, USA

into virtual scan chains, and it does not require bit-fixing logic to achieve complete fault coverage for benchmark circuits.

The *TPG* used in [19] is also similar in structure to the one shown in Figure 1, with the following differences. The random source in [19] consists of an AND gate driven by $k$ cells of an *LFSR*. This results in low transition probabilities that reduce the power dissipation during scan, which is the goal of [19]. However, since the transition probabilities are limited, not all the detectable circuit faults are detected in [19]. The method proposed here achieves 100% coverage of detectable circuit faults for all the benchmark circuits considered.

We consider circuits with single scan chains. The proposed *TPG* can be easily extended to circuits with multiple scan chains by including a $T$ flip-flop for every scan chain.

The paper is organized as follows. In Section 2 we demonstrate that it is possible to generate deterministic test sets with small numbers of transitions along their tests. In Section 3 we describe the application of this property to built-in test-pattern generation. Experimental results of built-in test-pattern generation are presented in Section 4.

## 2. Test sets with small numbers of transitions

To demonstrate that test sets with small numbers of transitions are effective in achieving complete fault coverage, we perform the experiment described in this section.

We start from a test set $T$ that detects all the detectable circuit faults. For every test $t_i \in T$ we find the number of transitions from 0 to 1 or from 1 to 0. We denote this number by $n_\sigma(t_i)$. We also find the set of positions where the transitions occur. We denote this set by $\sigma(t_i)$. For example, we show in Table 1 a test set $T$ for the combinational logic of ISCAS-89 benchmark circuit $s27$. For every test $t_i \in T$ we show $n_\sigma(t_i)$ and $\sigma(t_i)$.

**Table 1: Test set for** $s27$

| $i$ | $t_i$ | $n_\sigma(t_i)$ | $\sigma(t_i)$ |
|---|---|---|---|
| 0 | 0000011 | 1 | 5 |
| 1 | 1001010 | 5 | 1 3 4 5 6 |
| 2 | 0100110 | 4 | 1 2 4 6 |
| 3 | 0111001 | 3 | 1 4 6 |
| 4 | 1101011 | 4 | 2 3 4 5 |
| 5 | 1010000 | 3 | 1 2 3 |

We consider the tests starting from the ones with the highest numbers of transitions. When a test $t_i$ is considered, if $n_\sigma(t_i) > 0$, we attempt to reduce the number of transitions by modifying the test as follows. For every $j_1 \in \sigma(t_i)$, we find the next position $j_2 \in \sigma(t_i)$ where a transition occurs. If $j_1$ is the last position in $\sigma(t_i)$, we define $j_2 = n$ where $n$ is the number of circuit inputs. We complement positions $j_1, \cdots, j_2-1$ of $t_i$. This eliminates the transition at position $j_1$ without introducing any additional transitions. We check whether all the faults originally detected by $t_i$ are still detected by the test set. If any fault remains undetected, we reverse the change. Otherwise, we accept the change. In this case we recompute $\sigma(t_i)$ and consider $t_i$ again.

In the example of $s27$ we first consider $t_1$ that has five transitions. Considering $j_1 = 1$ with $j_2 = 3$, we modify $t_1 = 1001010$ into $t_1 = 1111010$. We find that five faults remain undetected and we therefore restore $t_1 = 1001010$. Considering $j_1 = 3$ with $j_2 = 4$, we modify $t_1 = 1001010$ into $t_1 = 1000010$. We find that three faults remain undetected and we therefore restore $t_1 = 1001010$. Considering $j_1 = 4$ with $j_2 = 5$, we modify $t_1 = 1001010$ into $t_1 = 1001110$. We find that three faults remain undetected and we therefore restore $t_1 = 1001010$. Considering $j_1 = 5$ with $j_2 = 6$, we modify $t_1 = 1001010$ into $t_1 = 1001000$. We find that all the faults continue to be detected and we therefore accept the change into $t_1 = 1001000$. We recompute $\sigma(t_1)$ to obtain $\sigma(t_1) = \{1,3,4\}$. We now reconsider all the values $j_1 \in \sigma(t_1)$ and find that no additional changes are possible.

Next, we consider $t_2$ of $s27$ and find that it cannot be modified. Considering $t_4$ next, we first consider $j_1 = 2$ with $j_2 = 3$. We modify $t_4 = 1101011$ into $t_4 = 1111011$ and find that all the faults are still detected. We therefore accept $t_4 = 1111011$ with the new set $\sigma(t_4) = \{4,5\}$. This is also the final test we obtain.

After considering all the tests once, we consider all the tests again in case the new test set allows additional transitions to be removed. In the example of $s27$ we obtain the test set shown in Table 2.

**Table 2: Modified test set for** $s27$

| $i$ | $t_i$ | $n_\sigma(t_i)$ | $\sigma(t_i)$ |
|---|---|---|---|
| 0 | 0000011 | 1 | 5 |
| 1 | 1001000 | 3 | 1 3 4 |
| 2 | 0100110 | 4 | 1 2 4 6 |
| 3 | 0000000 | 0 | |
| 4 | 1111011 | 2 | 4 5 |
| 5 | 1111111 | 0 | |

To further reduce the numbers of transitions in the test set, we attempt to replace the tests that have the largest numbers of transitions by other tests, that have fewer transitions. We define $n_{\sigma,\max} = \max\{n_\sigma(t_i): t_i \in T\}$. For every $t_i \in T$ with $n_\sigma(t_i) = n_{\sigma,\max}$, we add to $T$ the following tests. For every $j_1 \in \sigma(t_i)$, we find the next position $j_2 \in \sigma(t_i)$ as before. We define a test denoted by $t_i^{j_1}$ which is identical to $t_i$ except that positions $j_1, \cdots, j_2-1$ are complemented, and we add it to $T$.

In the example of $s27$, we have $n_{\sigma,\max} = 4$, and we add tests based on $t_2$. The test for $j_1 = 1$ is $t_2^1 = 0000110$, the test for $j_1 = 2$ is $t_2^2 = 0111110$, the test for $j_1 = 4$ is $t_2^4 = 0100000$, and the test for $j_1 = 6$ is $t_2^6 = 0100111$.

After extending the test set, we attempt to modify the tests again as above. The resulting test set is shown in Table 3.

**Table 3: Extended and modified test set for** $s27$

| $i$ | $t_i$ | $n_\sigma(t_i)$ | $\sigma(t_i)$ |
|---|---|---|---|
| 0 | 0000011 | 1 | 5 |
| 1 | 1001000 | 3 | 1 3 4 |
| 2 | 0000000 | 0 | |
| 3 | 0000000 | 0 | |
| 4 | 1111011 | 2 | 4 5 |
| 5 | 1111111 | 0 | |
| 6 | 0000000 | 0 | |
| 7 | 0111111 | 1 | 1 |
| 8 | 0100000 | 2 | 1 2 |
| 9 | 0000000 | 0 | |

We drop unnecessary tests by performing fault simulation with fault dropping and removing tests that do not detect any new faults. The resulting test set is shown in Table 4. At the cost of adding one test, the maximum number of transitions is reduced to three.

**Table 4: Reduced test set for** $s27$

| $i$ | $t_i$ | $n_\sigma(t_i)$ | $\sigma(t_i)$ |
|---|---|---|---|
| 0 | 0000011 | 1 | 5 |
| 1 | 1001000 | 3 | 1 3 4 |
| 2 | 0000000 | 0 | |
| 3 | 1111011 | 2 | 4 5 |
| 4 | 1111111 | 0 | |
| 5 | 0111111 | 1 | 1 |
| 6 | 0100000 | 2 | 1 2 |

We repeat the process of adding tests, modifying the test set and dropping unnecessary tests until no further improvements in the maximum or the total number of transitions are possible.

Results of the process above for the combinational logic of ISCAS-89 benchmark circuits are shown in Table 5. After the circuit name we show the number of inputs. Under column *tests* we show the numbers of tests before the test set is modified, and after the test set is modified. Under column *max trans* we show the maximum number of transitions for any test in the test set before the test set is modified, after the test set is modified without adding tests, and after the test set is modified with the addition of tests.

**Table 5: Numbers of transitions in deterministic test sets**

| | | tests | | max tran | | |
|---|---|---|---|---|---|---|
| circuit | inp | orig | mod | orig | mod1 | mod2 |
| s208 | 19 | 27 | 34 | 13 | 8 | 4 |
| s298 | 17 | 24 | 26 | 12 | 8 | 6 |
| s344 | 24 | 15 | 19 | 11 | 7 | 5 |
| s382 | 24 | 25 | 40 | 18 | 10 | 5 |
| s420 | 35 | 43 | 59 | 16 | 9 | 5 |
| s510 | 25 | 54 | 55 | 15 | 7 | 7 |
| s526 | 24 | 50 | 57 | 16 | 10 | 8 |
| s641 | 54 | 22 | 31 | 28 | 15 | 11 |
| s820 | 23 | 94 | 94 | 14 | 8 | 8 |
| s953 | 45 | 76 | 76 | 29 | 9 | 9 |
| s1423 | 91 | 26 | 71 | 47 | 24 | 12 |
| s5378 | 214 | 100 | 310 | 112 | 37 | 12 |
| s9234 | 247 | 111 | 288 | 98 | 43 | 16 |
| s13207 | 700 | 235 | 268 | 351 | 161 | 87 |
| s15850 | 611 | 97 | 321 | 192 | 63 | 21 |
| s38417 | 1664 | 87 | 94 | 741 | 183 | 162 |

Table 5 demonstrates that significant reductions in the numbers of transitions are possible at the cost of an increase in the number of tests. The final maximum number of transitions is significantly smaller than the number of circuit inputs, which is an upper bound on the number of transitions.

## 3. Built-in test-pattern generation

In this section, we describe the selection of parameters for a *TPG* with the structure shown in Figure 1. The parameters to be selected are transition probabilities and corresponding initial values for the $T$ flip-flop.

We denote the set of parameters by $P$. Every entry $(p,a) \in P$ consists of a transition probability $p$ and an initial value $a$ for the $T$ flip-flop. We discuss the logic required for implementing several pairs of parameters at the end of Section 4.

For a given pair of parameters $(p,a)$ we allow the *TPG* to produce $N$ tests and apply them to the circuit. Here, $N$ is a preselected constant. Thus, given a set of parameters $P$ and a value for $N$, test application to an $n$-input circuit with a single scan chain proceeds as follows.

**Procedure 1:** Test application based on $P$

For every $(p,a) \in P$:

Initialize the $T$ flip-flop to $a$.

Drive the $T$ flip-flop with a sequence where the probability of a 1 is $p$.

Repeat $N$ times:

Shift $n$ values produced by the $T$ flip-flop into the scan chain and apply the resulting test vector to the circuit.

We consider values of $p$ that are multiples of $1/\psi$ for $\psi$ which is a power of two, i.e., we consider $p = 1/\psi, 2/\psi, \cdots, (\psi-1)/\psi$. In our experiments, $\psi = 32$, 64 or 128. We note that a higher value of $\psi$ provides better control over the average number of transitions. For example, the average number of transitions with $p = 1/32$ and $p = 2/32$ is expected to be the same as the average number of transitions with $p = 2/64$ and $p = 4/64$, respectively. However, with $\psi = 64$ we also have an intermediate value obtained for $p = 3/64$. With $\psi = 128$ we can obtain in addition the intermediate values for $p = 5/128$ and $p = 7/128$.

We select the set of parameters $P$ for a circuit as follows. Given $N$ and $\psi$, we consider for every value of $p$, $p = 1/\psi, 2/\psi, \cdots, (\psi-1)/\psi$, both values of $a$, $a = 0$ and $a = 1$. We simulate the circuit under $N$ test vectors produced for every pair $(p,a)$ with fault dropping. We store *effective* pairs, i.e., pairs that result in the detection of at least one circuit fault, in the set $P$. This process is given as Procedure 2 next.

**Procedure 2:** Finding a set of parameters $P$ for given $N$ and $\psi$

(1) Let $F$ be the set of target faults. Set $P = \phi$.

(2) For $p = 1/\psi, 2/\psi, \cdots, (\psi-1)/\psi$ and for $a = 0,1$:

Generate $N$ test vectors with parameters $(p,a)$. Simulate $F$ under these test vectors with fault dropping. If any fault is detected, add $(p,a)$ to $P$.

The set $P$ obtained by Procedure 2 may contain more pairs of parameters than necessary. To remove such pairs, we perform reverse order simulation of the parameters in $P$. This is done using Procedure 3 given next.

**Procedure 3:** Reducing the size of $P$

(1) Let $F$ be the set of target faults. Let $P = \{(p_1,a_1),(p_2,a_2), \cdots, (p_k,a_k)\}$.

(2) For $i = k, k-1, \cdots, 1$:

Generate $N$ test vectors with parameters $(p_i,a_i)$. Simulate $F$ under these test vectors with fault dropping. If no faults are detected, remove $(p_i,a_i)$ from $P$.

Procedures 1, 2 and 3 require fault simulation of preselected numbers of vectors. This determines the computational complexity.

## 4. Experimental results

We applied Procedure 2 followed by Procedure 3 to ISCAS-89 benchmark circuits that are random pattern resistant. We also considered multi-level implementations of Berkeley PLAs that are random pattern resistant. Our definition of a random pattern resistant circuit requires that incomplete fault coverage would be achieved for the circuit after applying 100,000 random patterns.

We used $N = 1024, 2048, \cdots, 65536$ and $\psi = 32, 64, 128$. We report on the smallest value of $N$ and the smallest value of $\psi$ (in this order) that result in the highest fault coverage. In several cases we provide results for a larger value of $N$ or $\psi$ in order to demonstrate the tradeoff between the number of parameter pairs in $P$, the number of tests, $N$ and $\psi$.

The results are shown in Table 6 as follows. After the circuit name we show the values of $N$ and $\psi$. We then show the fault efficiency (the percentage of detected faults out of the set of detectable faults). Under column $|P|$ we show the number of parameter pairs in the set $P$ after reverse order simulation. Under column *max p* we show the maximum value of $p$ for any pair $(p,a) \in P$. Under column *tests* we show the total number of tests applied to the circuit to achieve the reported fault efficiency (the number of tests is equal to $N|P|$). For comparison, we show in the last column of Table 6 the fault efficiency achieved by the same number of pure-random patterns for some of the circuits considered.

Table 6 demonstrates that complete fault coverage is achieved for all the circuits considered. By increasing the number of tests $N$ and/or the probability parameter $\psi$ it is possible to reduce the number of parameter pairs required to achieve complete fault coverage. A higher value of $N$ sometimes allows a lower value of $\psi$ to be used.

**Table 6: Results of built-in test-pattern generation**

| circuit | N | ψ | f.e. | \|P\| | max p | tests | rand |
|---|---|---|---|---|---|---|---|
| s420 | 2048 | 128 | 100 | 6 | 26/128 | 12288 | 91.07 |
| s420 | 4096 | 32 | 100 | 2 | 5/32 | 8192 | 91.07 |
| s526 | 1024 | 32 | 100 | 6 | 23/32 | 6144 | 98.93 |
| s526 | 2048 | 32 | 100 | 5 | 23/32 | 10240 | 99.14 |
| s526 | 2048 | 128 | 100 | 4 | 69/128 | 8192 | 99.14 |
| s641 | 32768 | 64 | 100 | 4 | 26/64 | 131072 | 98.43 |
| s5378 | 4096 | 64 | 100 | 14 | 50/64 | 57344 | 99.93 |
| s5378 | 4096 | 128 | 100 | 10 | 77/128 | 40960 | 99.68 |
| s5378 | 8192 | 64 | 100 | 9 | 36/64 | 73728 | 99.98 |
| s5378 | 8192 | 128 | 100 | 8 | 61/128 | 65536 | 99.98 |
| s9234 | 65536 | 32 | 100 | 11 | 19/32 | 720896 | 96.61 |
| s9234 | 65536 | 64 | 100 | 10 | 34/64 | 655360 | 96.61 |
| s13207 | 4096 | 64 | 100 | 23 | 38/64 | 94208 | - |
| s13207 | 4096 | 128 | 100 | 21 | 75/128 | 86016 | - |
| s13207 | 8192 | 32 | 100 | 14 | 15/32 | 114688 | - |
| s13207 | 8192 | 64 | 100 | 8 | 34/64 | 65536 | - |
| s15850 | 65536 | 64 | 100 | 17 | 48/64 | 1114112 | - |
| s38584 | 32768 | 128 | 100 | 16 | 44/128 | 524288 | - |
| s38584 | 65536 | 128 | 100 | 12 | 37/128 | 786432 | - |
| rckl | 4096 | 64 | 100 | 4 | 7/64 | 16384 | 45.78 |
| x1dn | 4096 | 32 | 100 | 3 | 8/32 | 12288 | 92.93 |
| x9dn | 1024 | 32 | 100 | 5 | 6/32 | 5120 | 90.67 |

Next, we describe the hardware required by the proposed built-in test generation method. We consider $s420$ with $N = 4096$ and $\psi = 32$. These values result in two pairs of parameters, $(p,a) = (2/32,0)$ and $(p,a) = (5/32,1)$. The *TPG* consists in this case of a counter $CNT_N$ for $N$, and a counter $CNT_P$ that determines the current values of the parameters $(p,a)$. Depending on the value of $CNT_P$, the $T$ flip-flop in Figure 1 is loaded with the appropriate value $a$, and the random source of Figure 1 produces a sequence where the probability of a one is $p$. To obtain $p = 2/32$ or $1/16$, it is possible to use the AND function of four cells of a linear-feedback shift-register (*LFSR*). To obtain $p = 5/32$, it is possible to use the following. (1) The AND function of three cells of an *LFSR* will implement a function which is one with probability 1/8. (2) The AND function of five cells of an *LFSR* will implement a function which is one with probability 1/32. (3) The OR function of the two functions in 1 and 2 will implement a function which is one with probability 1/32+1/8 = 5/32, as required. The same *LFSR* can be used for all the functions required. A multiplexer controlled by $CNT_P$ can select between the appropriate functions. Another multiplexer controlled by $CNT_P$ can determine the initial value of the $T$ flip-flop based on $a$. Compared with other approaches, the proposed method has a very low hardware overhead while allowing all the detectable faults to be detected. For example, the method of [17], which is very efficient, requires in addition to hardware similar to that described above also bit-fixing logic to modify the values of selected bits. It also requires the partitioning of the scan chain into multiple virtual scan chains (up to 64 virtual scan chains), causing the hardware to be duplicated according to the number of virtual scan chains.

## 5. Concluding remarks

We described a procedure for reducing the numbers of transitions from 0 to 1 or from 1 to 0 when considering consecutive input values in a deterministic test set that achieves complete coverage of single stuck-at faults. The results of this procedure indicated that relatively small numbers of transitions are sufficient for achieving complete fault coverage. This pointed to the ability to base built-in test-pattern generation for scan circuits on transition probabilities instead of probabilities of specific bits being 0 or 1. The advantage of such an approach is that it associates only two parameters with every set of test patterns: an initial value and a transition probability. Based on the transition probability, the next value to be shifted into the scan chain is either equal to the previous value or its complement. We demonstrated that such an approach to scan-BIST is effective in achieving complete fault coverage for benchmark circuits.

## References


[1] P. H. Bardell, W. H. McAnney and J. Savir, *Built-In Test for VLSI Pseudorandom Techniques*, Wiley, 1987.

[2] E. B. Eichelberger, E. Lindbloom, J. A. Waicukauski and T. W. Williams, *Structured Logic Testing*, Prentice-Hall, 1991.

[3] V. D. Agrawal, C. R. Kime and K. K. Saluja, "A Tutorial on Built-In Self-Test, Part 1: Principles", IEEE Design & Test of Computers, March 1993, pp. 73-82.

[4] V. D. Agrawal, C. R. Kime and K. K. Saluja, "A Tutorial on Built-In Self-Test, Part 2: Applications", IEEE Design & Test of Computers, June 1993, pp. 69-77.

[5] H.-J. Wunderlich, "Multiple Distributions for Biased Random Test Patterns", in Proc. Intl. Test Conf., 1988, pp. 236-244.

[6] J. Savir and W. H. McAnney, "A Multiple Seed Linear Feedback Shift Register", in Proc. 1990 Intl. Test Conf., Sept. 1990, pp. 657-659.

[7] B. Koenemann, "LFSR-Coded Test Patterns for Scan Designs", in Proc. European Test Conf., March 1991, pp. 237-242.

[8] S. Pateras and J. Rajski, "Generation of Correlated Random Patterns for the Complete Testing of Synthesized Multi-Level Circuits", in Proc. Design Autom. Conf., June 1991, pp. 347-352.

[9] S. Hellebrand, S. Tarnick, J. Rajski and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers", in Proc. 1992 Intl. Test Conf., Sept. 1992, pp. 120-129.

[10] I. Pomeranz and S. M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set", IEEE Trans. on Computer-Aided Design, July 1993, pp. 1050-1058.

[11] N. A. Touba and E. J. McCluskey, "Altering a Pseudo-Random Bit Sequence for Scan-Based BIST", in Proc. Intl. Test Conf., Oct. 1996, pp. 167-175.

[12] L.-R. Huang, J.-Y. Jou and S.-Y. Kuo, "Gauss-Elimination-Based Generation of Multiple Seed-Polynomial Pairs for LFSR", IEEE Trans. on Computer-Aided Design, Sep. 1997, pp. 1015-1024.

[13] C. Fagot, O. Gascuel, P. Girard and C. Landrault, "On Calculating Efficient LFSR Seeds for Built-In Self Test", in Proc. European Test Workshop, 1999, pp. 7-14.

[14] K.-H. Tsai, R. Tompson, J. Rajski and M. Marek-Sadowska, "STAR-ATPG: A High Speed Test Pattern Generator for Large Scan Designs", in Proc. Intl. Test Conf., 1999, pp. 1021-1030.

[15] S. Hellebrand, H.-G. Liang and H.-J. Wunderlich, "A Mixed Mode BIST Scheme based on Reseeding of Folding Counters", in Proc. 2000 Intl. Test Conf., Oct. 2000, pp. 778-784.

[16] L. Li and Y. Min, "An Efficient BIST Design Using LFSR-ROM Architecture", in Proc. 9th Asian Test Symp., Nov. 2000, pp. 386-390.

[17] N. Z. Basturkmen, S. M. Reddy and I. Pomeranz, "Pseudo Random Patterns Using Markov Sources for Scan BIST", in Proc. 2002 Intl. Test Conf., Oct. 2002, pp. 1013-1021.

[18] C. Yu, W. Li, S. M. Reddy and I. Pomeranz, "An Improved Markov Source Design for Scan BIST", in Proc. Intl. On-Line Testing Symp., July 2003.

[19] S. Wang and S. K. Gupta, "LT-RTPG: A New Test-Per-Scan BIST TPG for Low Heat Dissipation", in Proc. Intl. Test Conf., Sept. 1999, pp. 85-94.