2005

On Finding Consecutive Test Vectors in a Random Sequence for Energy-Aware BIST Design

Sheng Zhang
University of Nebraska - Lincoln, szhang@cse.unl.edu

Sharad C. Seth
University of Nebraska - Lincoln, seth@cse.unl.edu

Bhargab B. Bhattacharya
Indian Statistical Institute, Kolkata, India, bhargab@isical.ac.in

On Finding Consecutive Test Vectors in a Random Sequence for Energy-Aware BIST Design

Sheng Zhang, Sharad C. Seth
Department of Computer Sci. and Eng.
University of Nebraska-Lincoln
Lincoln, NE 68588-0115, USA
{szhang, seth}@cse.unl.edu

Bhargab B. Bhattacharya
ACM Unit
Indian Statistical Institute
Calcutta – 700 108, India
bhargab@isical.ac.in

Abstract

During pseudorandom testing, a significant amount of energy and test application time is wasted for generating and for applying "useless" test vectors that do not contribute to fault dropping. For low-power testing, modification logic/ROM may be used to skip the LFSR states that generate useless test patterns. The overhead of extra logic increases rapidly with the number of such jumps. Since identification of useless patterns strongly depends on the order in which incremental fault simulation is performed, an elegant solution to this problem would be to find a minimum set of segments in the LFSR sequence, where each segment corresponds to a consecutive subsequence of useful test patterns. This is formulated as consecutive test cover (CTC) problem, where the objective is to optimize a cost function combining the number of segments and the number of useful test patterns. The proposed heuristic algorithm to solve the CTC problem includes a "gap" parameter to allow a controllable number of useless patterns. Experiments on ISCAS-89 benchmark circuits reveal considerable reduction in the number of segments without any degradation of modeled fault coverage.

1. Introduction

Power/energy minimization during testing has become important for deep sub-micron technology because of higher clock rates and device densities [1]. A recent survey by Girard nicely summarizes the research techniques to minimize power for both external testing and built-in self-test (BIST) [2]. Existing techniques for power/energy minimization use toggle suppression [6], low-power test pattern generators (TPG) [4], BIST [5, 7, 11], scan testing based on Golomb coding [3], or new scan-path architectures [12, 13].

In BIST applications, a TPG is usually implemented as linear feedback shift register (LFSR). To achieve high fault coverage, a very long pseudorandom test sequence (*TS*) is required. Thus, a significant amount of energy and test application time is wasted for generating and for applying such a long sequence. Energy-aware BIST design should therefore consider power/energy consumption as a key issue in addition to the usual constraints on fault coverage, test application time, and area overhead.

Earlier schemes for LFSR modification include changing the characteristic polynomial of an LFSR to generate pre-computed deterministic test patterns [9], or replacing the LFSR with a finite-state machine [10]. Other proposals for energy-aware BIST have focused on the distinction between "useful" *vs*. "useless" pseudorandom patterns [5, 6, 11]. The former class is defined by patterns that contribute to modeled fault coverage *when applied in the order of test pattern generation*. The set of useful patterns is typically a very small subset of *TS* but they may be distributed throughout the sequence. For example, in s38417 only 620 patterns are found to be useful out of 20,000 applied random tests [5]. Girard et al. use vector-inhibiting technique to filter out useless patterns [11]. In [5], the LFSR is modified with a mapping logic to generate only the useful test patterns. There are two problems with this solution: (i) the overhead of mapping logic increases directly with the number of such jumps, and (ii) skipping of useless test vectors may cause degradation of non-modeled fault coverage. All of the above schemes do not address the problem of area overhead or the power consumed in the modification logic, or that of selecting an optimal set of useful vectors.

2. Main Results

In this paper, we consider BIST structures that employ the technique of modifying the LFSR to block useless test patterns. Our main focus is on reducing energy and test application time, but with minimum mapping logic overhead and with no loss of modeled fault coverage. A set of consecutive patterns generated by an LFSR is called a *segment*. We introduce the *consecutive test cover* (CTC) problem in a linearly-ordered bipartite graph to formulate the objective of reducing the number of segments (which lessens area/energy overhead of the modification logic for the LFSR) as well as the number of useful test patterns

Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design (VLSID'05)
1063-9667/05 $20.00 © 2005 **IEEE**

(which reduces overall energy consumption and test application time). This is clearly different from the classical LFSR reseeding problem or test compaction problem. A heuristic algorithm is proposed to solve the CTC problem. It includes a parameter called *gap* that can be controlled for further reduction of the number of segments at the expense of inclusion of some useless test patterns, which are often considered desirable for coverage of non-modeled faults. Experimental results on ISCAS-89 benchmark circuits demonstrate the effectiveness of the proposed scheme. We also provide comparative data to place our results in the context of earlier works [5, 11].

3. The Consecutive Test Cover Problem

The proposed algorithm assumes a test-per-scan scheme. The LFSR runs autonomously after being loaded with an initial seed, and generates the test sequence. For each pattern in the sequence, incremental fault simulation is performed to determine its fault coverage.

Modification to the LFSR state-table is required only to skip the useless patterns, i.e., for jumping to a non-consecutive state in the normal sequence. For generating patterns within a segment, the LFSR may be allowed to run freely according to its state table, without any additional mapping logic. The overhead may be cut down by reducing the number of segments; however, if the segments are constrained to include only the useful vectors, then their number may be too large. If we relax this constraint, significant reduction in the number of segments is possible while only paying a modest penalty in the total number of patterns over the useful patterns. The following example motivates us to define the *consecutive test cover* (CTC) problem.

In Figure 1, let $TS = \{t_1, t_2, t_3, \ldots, t_{12}\}$ denote the sequence of random test patterns generated by an LFSR, and let $F = \{f_1, f_2, f_3, \ldots, f_9\}$ denote the set of target faults in the CUT. The relations between the tests and detected faults can be described by a bipartite graph [8], where *TS* and *F* represent the two disjoint sets of nodes, and an edge *(t, f)* is given if and only if a test *t* detects a fault *f*. A fault *f* is said to be covered by a test set *T* if it contains at least one test pattern *t* that detects *f*. Let the bipartite graph of Figure 1a represent an instance of test and fault detection profile of an example.

In order to identify the useful patterns, the usual practice is to run incremental fault simulation. Thus, fault simulation in forward order renders the test pattern t_4 useless. This happens because the faults $\{f_1, f_4\}$ that can be detected by t_4 have already been detected by the tests $\{t_1, t_2\}$ chosen earlier and hence it does not contribute to incremental fault dropping. Similarly, the tests $\{t_7, t_{11}, t_{12}\}$ will be marked as useless. The remaining *8* useful patterns appear as *3* segments: $\{t_1, t_2, t_3\}$, $\{t_5, t_6\}$, and $\{t_8, t_9, t_{10}\}$ (Figure 1b). Once an initial set of useful patterns is identified, reverse fault simulation (from bottom to top) on this chosen set may be performed to reduce the size of the useful pattern set further. In Figure 1c, the result of reverse simulation is shown, where the test t_2 is found to be useless as the fault f_3 is detected by t_3 earlier in reverse order. Thus, we end up with *7* useful patterns appearing as *4* segments: $\{t_1\}$, $\{t_3\}$, $\{t_5, t_6\}$, and $\{t_8, t_9, t_{10}\}$ (Figure 2c). Therefore, *3* jumps in the state space of the LFSR would be needed to skip the useless patterns.

A close look into the graph however, leads to a better solution. In Figure 1d, we choose only *2* segments - $\{t_2, t_3, t_4, t_5\}$ and $\{t_{10}, t_{11}, t_{12}\}$, which cover all the faults. Hence, *7* useful patterns can be generated by employing only one jump, without any loss of fault coverage. Such optimization problem can be abstracted as the following *consecutive test cover* (CTC) problem in a bipartite graph. The set *TS* is *linearly-ordered* from top to bottom as it denotes a sequence of tests. We assume that each node in *F* is covered by at least one node in *TS*, i.e. the faults, which are either redundant, or not detectable by *TS* are not represented in the set *F*.

Given a bipartite graph *G(TS, F)*, where the elements of the set *TS* is linearly-ordered, the CTC problem is to find a subset *T* of *TS* such that:

(i) *T* covers all nodes in *F,* and
(ii) the size of *T* is minimum, and
(iii) the number of segments (*q*) in *T* is minimum.

It may be noted that the criteria (ii) and (iii) may not be satisfiable simultaneously. The optimization problem satisfying (i) and (ii) is already known to be NP-hard [18], and in all its likelihood would remain intractable for (i) and (iii) when the size of *T* is bounded. In the next section, we present a BIST scheme based on a heuristic solution to the CTC problem that identifies pareto points for designers to explore this additional dimension of the design space.

4. A Heuristic Algorithm for Solving the CTC Problem

It has been observed that the bipartite graph *G(TS, F)* defined in Section 3 depicts certain behavioral patterns for benchmark circuits. The degree distribution of nodes on the test side (*TS*) is found to be nearly uniform, whereas degrees of nodes on the fault side (*F*) are chaotic in nature [8]. This indicates that the number of faults detectable per vector is nearly the same, but the number of test patterns detecting a fault may largely vary. In other words, given a CUT, its fault set can be categorized into *hard-to-detect* (HTD) faults and *easy-to-detect* (ETD) faults.

Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design (VLSID'05)
1063-9667/05 $20.00 © 2005 **IEEE**

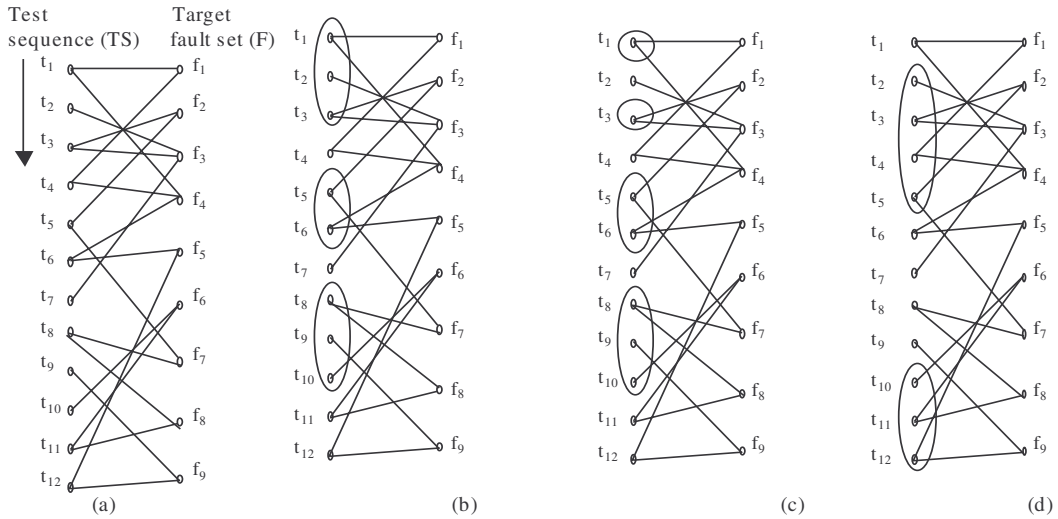

Figure 1: (a) Test sequence and fault coverage as a linearly-ordered bipartite graph; (b) useful vectors after forward fault simulation; (c) useful vectors after forward and reverse fault simulation; (d) optimal solution of the CTC problem with *2* segments and *7* test vectors

Further, patterns that cover all HTD faults will tend to cover many other ETD faults as they have high testability. Based on this property, we propose a heuristic solution to the CTC problem.

Let N_{up} be the number of useful patterns obtained after performing forward and reverse fault simulation on the LFSR test sequence *TS* with fault dropping. A compact segment set $S = \{S_1, S_2, ..., S_q\}$, where $|S| = q$ denotes the number of segments, should satisfy the following conditions:

1. *S* should cover all modeled faults detected by *TS*
2. $|S_1| + |S_2| + ... + |S_q| \sim N_{up}$
3. $q << N_{up}$

The first condition ensures that there should not be any degradation of fault coverage; the second condition ensures that the number of applied patterns is nearly equal to that obtained by a naïve choice, and hence does not worsen energy-saving property. The third condition ensures that the complexity of mapping logic is smaller. There is a trade-off between the number of segments and the total number of the applied patterns. On one extreme, we can put the full pseudorandom sequence *TS* in a single segment of a very large number of patterns; on the other extreme only the useful patterns are included thereby increasing the number of segments. In the following, we present a heuristic algorithm to find a compact segment set while keeping the number of useful patterns low. The algorithm does not construct the bipartite graph explicitly.

We limit our discussion to random pattern testable single stuck-at faults. Let $TS = \{t_1, t_2, t_3, \ldots, t_n\}$ be the sequence of pseudorandom test patterns generated by the LFSR and $F = \{f_1, f_2, f_3, \ldots, f_k\}$ be the set of HTD faults. The HTD faults can be determined by using a testability analysis program [14, 15], or by running a fault simulator with fault dropping [16].

We propose a greedy algorithm for the CTC problem based on segment selection, which is stated below:

1. A small subset of the faults is identified as HTD faults: F_{htd}. The algorithm first selects segments to cover this subset; most of the remaining faults will be detected by the already selected segments. The procedure to select these segments can then be repeated to cover the remaining undetected faults.
2. The original LFSR sequence of test vectors is simulated without fault dropping (full fault simulation) over F_{htd}. The result naturally divides the original LFSR sequence into (maximal) useful segments of consecutive test vectors each of which detects at least one fault in F_{htd}, separated by complementary (useless) segments of consecutive test vectors that do not detect any fault in F_{htd}. The size of a segment is defined as the number of test vectors in it.
3. A gap parameter *g*, with a positive integer value, is used to collapse consecutive useful segments that are separated by useless segments of size *g* or less. Thus a gap value *g* allows *g* patterns not detecting new HTD faults to be added to a segment if the (*g* + 1)th pattern detects a new HTD fault. However, these *g* patterns may contribute to detecting faults not in F_{htd}.
4. Given *g*, the useful segments define the candidate set from which the algorithm selects a subset that covers all the faults in F_{htd} while trying to minimize the size of the

selected subset. The selection process uses repeated steps and in each step greedily selects a useful segment according to segment-testability measure defined below.

5. *Segment-testability* of a fault $f \in F_{htd}$ is defined as the number of segments σ_f that detects f. For a segment that detects f, its *effectiveness* in detecting the fault is defined by $1/\sigma_f$. When the segment uniquely detects the fault, its effectiveness has the highest possible value of 1. The overall effectiveness of a segment s in detecting faults in F_{htd} is then defined as follows:

$$E_s = \frac{\sum_{\substack{f \in F_{htd} \\ \text{and } s \text{ detects } f}} 1/\sigma_f}{|S|}$$

At each step, the algorithm selects the segment with the highest average effectiveness value. The set F_{htd} and the fault table are updated by removing the faults that were detected by this segment. Thereafter, we reinitialize the HTD faults with the remaining undetected RTFs and run the algorithm again to cover these remaining faults. On termination, the selected segments will detect all originally identified HTD faults and most of the other faults that are random-pattern testable, but will miss covering the HTD faults that failed to be identified by the chosen method of HTD identification (fault simulation with fault dropping in our case). The final result is obtained by merging the results of these two passes. We can further reduce the number of test patterns by running reverse fault simulation to remove the useless patterns located at the boundaries of the segments. The (modeled) fault coverage after segmentation thus remains unchanged. Choice of suitable gap values leads to inclusion of more useless vectors in the segments, which in turn may increase the coverage of non-modeled faults as well. A pseudo-code for one pass of the proposed algorithm is described below.

Segment_Select(g)
F_{htd} = Initial hard-to-detect fault set;
enter gap value g;
While $F_{htd} \neq \varnothing$
  determine the set S of useful segments;
  compute segment-testability for each fault;
  For each segment $s \in S$ compute its effectiveness E_s;
  Let $\lambda = arg\ max\{E_s \mid s \in S\}$;
  Select λ;
  $F_{htd} \leftarrow F_{htd} - faults$ detected by λ;
end while.

5. Results and Discussions

The algorithm was tested on the ISCAS-89 (scan) benchmark circuits. A 25-bit LFSR was used to generate an initial test sequence of 20,000 pseudorandom patterns. Fault simulation was performed (with fault dropping) for single stuck-at faults using the HOPE fault simulator [16] to determine the original numbers of segments and useful patterns (Table 1, columns 4 & 5). The HTD faults were chosen as a fraction of the detectable faults that were detected last by the pseudorandom patterns.

In the first experiment, we compare the numbers of segments and patterns obtained by of algorithm for zero gap value against the original values obtained after fault simulation. With zero gap value, the algorithm will only combine useful patterns (that add to the fault coverage) in a segment. The results are shown in Table 1. Substantial savings, particularly in the number of segments, are evident from these results.

Table 1: Numbers of segments and patterns before and after applying the algorithm for gap value = 0

Circuit	# of faults	FC (%)	By fault simulation		After segmentation		Seg. Ratio	Pat. Ratio
			# of segments	# of patterns	# of segments	# of patterns	col-6/col-4	col-7/col-5
s1196	1242	99.3	142	158	75	139	0.53	0.88
s1488	1486	100.0	121	145	64	128	0.53	0.88
s5378	4551	98.6	244	268	134	251	0.55	0.94
s9234.1	6927	85.3	285	296	185	275	0.65	0.93
s13207.1	9815	94.7	407	423	302	420	0.74	0.99
s15850.1	11725	91.7	314	339	200	304	0.64	0.90
s38417	31180	93.7	574	630	417	599	0.73	0.95
s38584.1	36303	95.2	632	688	397	636	0.63	0.92

A second experiment was conducted to determine the effectiveness of the gap parameter in further reducing the number of segments. The results for the four largest benchmark circuits are shown graphically in Figures 2a and 2b. The charts show that dramatic savings are possible with larger values with only a modest increase in the number of test patterns. The number of segments appears to decay exponentially and the number of patterns rises almost linearly with the gap value. As example, for the s38417 circuit, there were 417 segments comprising 599 patterns for gap value = 0. The corresponding values for gap = 19 are 191 segments and 2781 patterns.

The above behavior of dramatic decline in the number of segments with only a modest increase in the number of test patterns provides design optimization opportunities based on energy *vs*. logic costs. In Figure 3, we show the complexity of control logic depending on the number of segments for s38417.scan with gap value varying from 0 to 19. The control logic is synthesized with *sis* and mapped to the *stdcell2_2.genlib library*. A linear relationship between the area of the control logic and the number of segments is evident.

In the BIST environment, the parameters for optimization might be the costs of energy (test length) and the control logic (number of segments). We explored a linear cost function for this optimization:

$$\text{cost} = \{\alpha \times \text{number-of-segments}\} + \{(1-\alpha) \times \text{number-of-test-patterns}\}$$

where α is a parameter in the [0, 1] range to account for the relative weights assigned to the energy *vs.* logic costs. Figure 4 shows the cost function for various values of the gap value for the s38417 circuit. The curves obtained for α ranging from 0.5 to 1.0 are shown. For α = 0.5 (and below) the zero gap value provides the optimum solution. However, between the [0.5,1.0) range, the curves dip down and then rise again, with the minimum value occurring for small gap values greater than 0. Thus, depending on the value of α, a designer can choose an optimum gap value to minimize the cost. When α = 1.0, the cost however, is entirely dependent on the number of segments and decreases continuously with the gap value.

6. Comparison with Prior Work

The idea of skipping parts of a pseudorandom test sequence for low-energy BIST design was reported earlier in [5, 11]. Authors in [11] suggest analyzing the incremental fault coverage of the test sequence in its *original* order and inhibiting the largest useless subsequences (those that detect no new faults) from being applied to the circuit under test. Further, in order to minimize the cost of logic, they suggest inhibiting only a small number of such subsequences; their reported overhead for *single inhibition* is in the 2% range. Further, when we ran our algorithm on s1488, a gap value of 230 produced 6 segments comprising of 1198 test patterns. Following the approach in [11], we again obtained a 6-segment solution that consisted of 1664 patterns by deleting the largest 5 useless subsequences. Thus, for the same number of segments, the proposed CTC algorithm retains a fewer number of test patterns compared to that in [11]. We also synthesized s34817.scan (along with the multiplexers that feed the scan flip-flops) and the mapping logic for various gap values. The area overhead is found to lie between 9% (for g = 19) and 18% (for g = 0).

7. Conclusions

We have introduced a new problem called consecutive test cover of a random sequence and presented a heuristic method to solve it. It offers a new approach to low-cost LFSR modification by selecting segments of consecutive test vectors and without any loss of modeled fault coverage. The total energy requirement during testing decreases drastically. For example, in s38584.1, only 636 test patterns in 397 segments are needed out of 20000 random patterns to achieve the same fault coverage after segmentation (Table 1). Thus, it will provide significant amount of savings in terms of energy consumption as well as hardware overhead. This approach to compaction is essentially different from that of Kiefer et al. [17], which uses a mapping logic to embed deterministic test patterns to achieve high fault coverage instead of using a state control unit to suppress useless patterns, as in our design. A future research problem would be to investigate how to incorporate the classical reseeding or test embedding techniques in the proposed method in order to increase fault coverage of random-pattern resistant faults.

References

[1] N. Nicolici and B. M. Al-Hashimi, Power-Constrained Testing of VLSI Circuits. *Kluwer Academic Publishers*, Boston, MA, 2003.

[2] P. Girard, "Survey of low-power testing of VLSI circuits," *IEEE Design & Test of Computers*, pp. 82-92, May-June 2002.

[3] A. Chandra and K. Chakrabarty, "Low-power scan testing and test data compression for system-on-a-chip," *IEEE TCAD*, pp. 597-604, May 2002.

[4] S. Wang and S.K. Gupta, DS-LFSR: "A BIST TPG for low switching activity," *IEEE TCAD*, pp. 842-851, July 2002.

[5] B. B. Bhattacharya, S. C. Seth, and S. Zhang, "Low- energy BIST design for scan-based logic circuits," *Proc. Int. Conf. on VLSI Design*, pp. 546-551, 2003.

[6] S. Gerstendoerfer and H. -J. Wunderlich, "Minimized power consumption for scan-based BIST," *Proc. ITC,* pp. 77-84, 1999; (also in *JETTA,* January 2000).

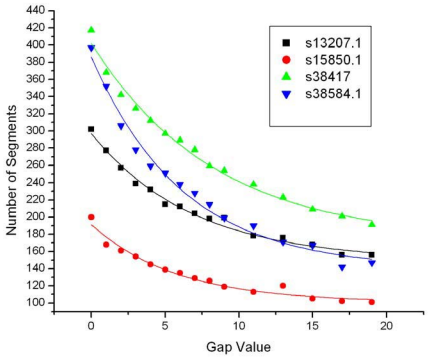

Figure 2a: # segments *vs.* gap

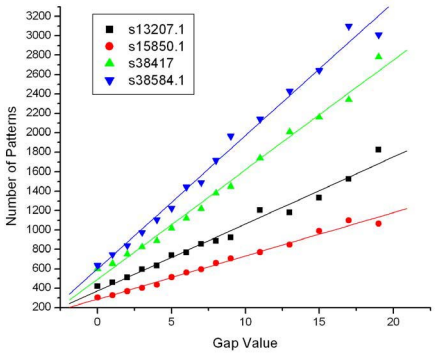

Figure 2b: # patterns *vs.* gap

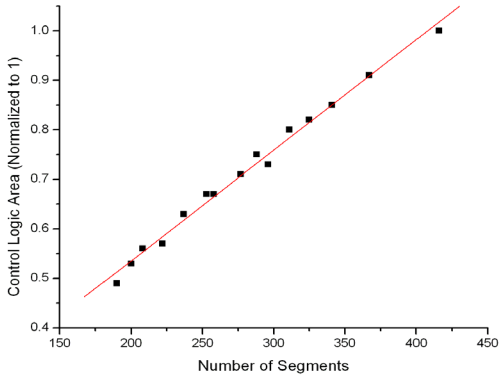

Figure 3: Overhead *vs.* # segments

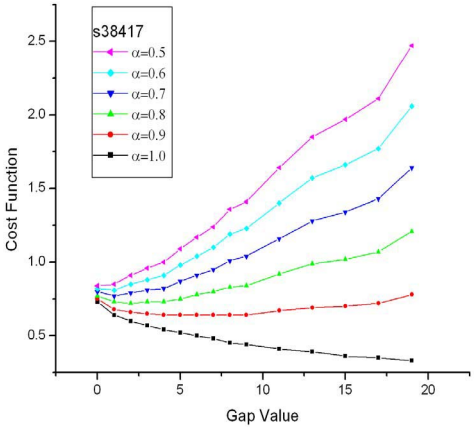

Figure 4: Cost *vs.* gap

[7] N. Z. Basturkmen, S. M. Reddy, and I. Pomeranz, "A low power pseudorandom BIST technique," *JETTA*, vol. 19, no. 6, pp. 637-644, Dec. 2003.

[8] H. Cui, S. C. Seth, and S. K. Mehta, "Modeling fault coverage of random test patterns," *JETTA,* vol. 19, no. 3, pp. 271-284, June 2003.

[9] S. J. Upadhyaya and L. C. Chen, "On-chip test generation for combinational circuits by LFSR modification," *Proc. ICCAD*, pp. 84-87, 1993.

[10] S. Swanminathan and K. Chakrabarty, "A deterministic scan-BIST architecture with application to field testing of high-availablility systems," *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 259-262, May 2001.

[11] P. Girard, P., L. Guiller, C. Landrault, and S. Pravossoudovitch, "A test vector inhibiting technique for low energy BIST design," *Proc. VTS*, pp. 407-412, 1999.

[12] O. Sinanoglu and A. Orailoglu, "A novel scan architecture for power efficient, rapid test," *Proc. ICCAD*, Nov. 2002.

[13] B. Bhattacharya, S. Seth, and S. Zhang, "Double-tree scan: A novel low-power scan-path architecture," *Proc. Int. Test Conf.*, pp. 470-479, 2003.

[14] L. H. Goldstein, "Controllability/Observability analysis of digital circuits," *IEEE Trans. Circuits and Systems*, vol. CAS-26, pp. 685-693, September 1979.

[15] S. K. Jain and V. D. Agrawal, "Statistical fault analysis," *IEEE Design & Test*, vol. 2, pp. 38-44, Feb. 1985.

[16] H. K. Lee and D. S. Ha, "HOPE: an efficient parallel fault simulator for synchronous sequential circuits," *Proc. 29th ACM/IEEE DAC*, pp. 336-340, 1992.

[17] G. Kiefer and H. Wunderlich, "Using BIST control for pattern generation," *Proc. ITC*, pp. 347-355, 1997.

[18] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness. *W. H. Freeman and Co.*, USA, 1979.