Sequence Length, Area Cost and Non-Target Defect Coverage Tradeoffs in Deterministic Logic BIST

Piet Engelke* Valentin Gherman** Ilia Polian* Yuyi Tang**
Hans-Joachim Wunderlich** Bernd Becker*

*Department of Computer Architecture
Institute for Computer Science
Albert-Ludwigs-University
Georges-Köhler-Allee 51
D-79110 Freiburg i. Br., Germany
{engelke|polian|becker}@informatik.uni-freiburg.de

**Institute of Computer Architecture
and Computer Engineering
University of Stuttgart
Pfaffenwaldring 47
D-70569 Stuttgart, Germany
{ghermanv|Yuyi.Tang|wu}@informatik.uni-stuttgart.de

Abstract

For the first time, we study the coverage of non-target defects for Deterministic Logic BIST (DLBIST) architecture. We consider several DLBIST implementation options that result in test sequences of different lengths. Resistive bridging faults are used as a surrogate of non-target defects. Experimental data obtained for largest ISCAS benchmarks suggests that, although DLBIST always guarantees complete stuck-at coverage, test sequence length does influence the non-target defect detection capabilities. For circuits with a large fraction of random-pattern resistant faults, the embedded deterministic patterns as well as a sufficient amount of random patterns are both demonstrated to be essential for non-target defect detection. It turns out, moreover, that area cost is lower for DLBIST solutions with longer test sequences, due to additional degrees of freedom for the embedding procedure and a lower number of faults undetected by pseudo-random patterns. This implies that DLBIST is particularly effective in covering non-target defects.

Keywords: Test Tradeoffs, Logic BIST, Defect Coverage, Resistive Bridging Faults

1 Introduction

Built-In Self Test (BIST) [1, 2] is an attractive technique in nanoscale technologies. After having become predominant for memories in the 1990s, it is now increasingly used for logic parts of a circuit. There are several advantages of BIST over conventional, tester-based test application process. First, large test sets do not have to be stored on the tester. Second, they do not have to be transferred from the tester to the device under test through a narrow interface, which is particularly difficult when testing deeply embedded cores. Third, test application can be performed at speed, which is a precondition for detecting dynamic defects and otherwise requires costly high-speed tester channels and IC pins that function correctly at high speed. Fourth, BIST mechanisms are useful for in-field inspection and are important for on-line error detection [3]. Fifth, external test access to systems such as smartcards might compromise the security standards of these systems.

Historically, BIST solutions are based on applying pseudo-random pattern sequences produced by a *test pater generator* block, often a linear feedback shift register (LFSR) or an LFSR derivative. However, many circuits have faults that are not detected by these sequences (pseudo-random resistant faults). There are several solutions to this problem. The straightforward technique is the application of deterministic patterns in addition to the pseudo-random sequence. Typically, the number of such patterns is still high, as the random-pattern resistant faults often require specific deterministic patterns. Inserting *test points* [4], the fault coverage can be improved, at the expense of increased area and delay, and the need for a new synthesis and timing verification run. In *weighted random testing* (which has been proposed for single-pattern [5] and two-pattern testing [6]), the signal probabilities of the pseudo-random sequence are modified such as to detected yet-undetected faults. This requires a complex synthesis process tightly coupled with ATPG, and many probability sets for some circuits, resulting in a large and complex control architecture. The *Circular Self-Test Path* technique uses the circuit functionality to generate new patterns; it has also been applied to single-pattern [7] and two-pattern [8], but in general fails to achieve a desired fault coverage.

In contrast to the above-mentioned schemes, the *deterministic logic BIST* (DLBIST) architectures guarantee the application of a predefined deterministic test set among the pseudo-random patterns. *Reseeding* omits those parts of the pseudo-random sequence that are not needed for testing. Pseudo-random pattern generators used in reseeding architectures so far include LFSRs [9, 10], multiple-polynomial LFSRs [11, 12], twisted-ring counters [13] and folding counters [14]. A similar solution for two-pattern testing is based on a multiple input shift register [15, 16]. *Test set embedding* modifies some of the bits of the pseudo-random sequence such that every pattern from a given deterministic test pattern set occurs in the sequence. Instances of test set embedding are *bit-flipping* [17, 18] (which inverts selected bits using XOR gates controlled by the *bit-flipping logic* BFL) and bit-fixing [19] (which employs AND and OR gates controlled by the bit-fixing logic BFX). A related technique is *Embedded Deterministic Test* (EDT) [20], which involves interaction with the tester and hence is considered a test compression rather than a BIST method.

Traditional optimization goals in design of a BIST architecture are the area cost of the additional circuitry and the test application time, which is given by the length of the sequence. However, there is a further parameter that did not receive proper attention in earlier works: the detection of actual defects. Stuck-at fault coverage is traditionally the sole test quality target of a BIST solution. While the poor correspondence between stuck-at faults and actual defects was well known long ago [21], new defect mechanisms in deep submicron technologies [22] imply the need to evaluate the coverage of unmodeled defects by a BIST scheme.

In this paper, we analyze the relation between the test length, the area cost, and the unmodeled defect detection capability of the DLBIST architecture based on bit flipping. Since there is no such thing as an accurate 'model of unmodeled defects', we are forced to use a surrogate instead. In this paper, *resistive bridging faults* (RBF) serve as such a surrogate. There are several reasons for the choice of RBF. First, this model corresponds to resistive short defects, which are increasingly more important in nanoscale technologies. Second, it accurately reflects electrical phenomena including *pattern-dependency* (different circuit behavior for different patterns seemingly leading to same logical values on internal lines) and *Byzantine behavior* (voltage on a fanout that is interpreted as logic-1 by some of the succeeding gates and as logic-0 by others). These phenomena are ignored by the stuck-at fault model. Finally, the BIST logic synthesis algorithm does not employ any RBF information and does not target any RBFs. Consequently, all RBF detections must be purely accidental. Since RBF detection conditions are sufficiently different from stuck-at detection conditions, RBFs are likely to be representative for coverage of a class of unmodeled defects.

Unmodeled defect detection by DLBIST was also analyzed in [23]. However, that paper concentrated on handling designs that produce unknown values which potentially could corrupt the signature and invalidate the results of a BIST run. Consequently, an additional logic block called XML (X Masking Logic) was used in [23] to block the unknown values without compromising the stuck-at fault coverage of the test. The implications of adding XML to the BIST architecture for detection of unmodeled defects were studied, and additional requirements for XML design were formulated such that unmodeled defect coverage is kept high. In contrast to [23] that considered potential coverage loss between the block under test and the test response evaluator, this paper focuses on unmodeled defect coverage of the sequence applied by the test pattern generator to the block under test.

We synthesize several DLBIST schemes and investigate the three-dimensional tradeoff of associated area cost, test application time and RBF coverage. It turns out that longer test sequences result in better detection of non-target defects (RBFs) and also smaller bit-flipping logic. The first result indicates that the embedded deterministic stuck-at patterns alone are not sufficient for good coverage of unmodeled defects and that the additional pseudo-random patterns are useful for this purpose. Since DLBIST applies both deterministic and pseudo-random patterns, it appears to be superior to conventional tester-based test application in covering unmodeled defects.

The remainder of the paper is organized as follows. The DLBIST architecture and resistive bridging faults are reviewed in the Sections 2 and 3, respectively. Experimental results are reported in Section 4. Section 5 concludes the paper.

2 Deterministic Logic BIST

This section reviews the bit-flipping DLBIST architecture [17, 18] and its synthesis [24].

2.1 Architecture

Figure 1 shows the bit-flipping DLBIST architecture. An LFSR with a phase shifter is used as the source of random patterns. Some of the patterns are *useless*, i.e. they do not detect any (stuck-at) faults not detected by other patterns. In order to achieve the desired fault coverage, some of the bits produced by the LFSR are inverted (flipped), which is controlled by *bit-flipping logic* (BFL). BFL is a combinational block that takes the LFSR state, the pattern number (from the pattern counter) and the bit number (from the bit counter) and selects the LFSR outputs to be inverted by driving a logic-1 at the inputs of the corresponding XOR gates. Note that the architecture can be used for combinational and

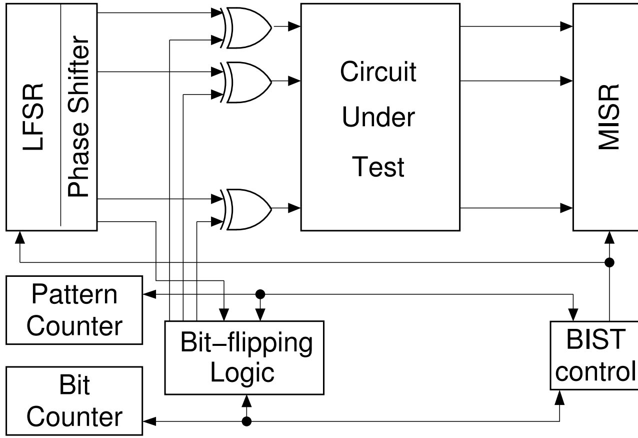

Figure 1: Deterministic Logic BIST

scan cores employing one or several chains [18]. The responses of the CUT are fed into a MISR.

2.2 Synthesis

The synthesis procedure for the DLBIST hardware is broken into the following five steps (refer to [24] for details):

1. Fault simulate the LFSR sequence; drop detected faults.
2. Perform ATPG (without random fill of don't care positions) for the remaining faults.
3. For every deterministic test pattern obtained by ATPG, select a useless pseudo-random pattern from the pseudo-random sequence to be transformed into that pattern (details are given below).
4. Synthesize compact BFL that performs the transformation of useless patterns into deterministic patterns (details are given below).
5. Fault simulate the final sequence produced by LFSR combined with BFL.

A pseudo-random pattern r that is selected for assignment to a deterministic ATPG pattern t in Step 3 has three kinds of bits: *matching bits* (specified in t, same value in r and t); *conflicting bits* (specified in t, opposite values in r and t); and *don't care* bits (unspecified in t). More formally, if the ith bit in r (t) is denoted as r_i (t_i), i is a matching bit if $t_i = r_i = 1$ or $t_i = r_i = 0$, a conflicting bit if either $t_i = 1$ and r_i = 0 or $t_i = 0$ and $r_i = 1$, and a don't care bit if $t_i = X$ irrespective of the value of r_i (where X stands for a don't care value). Conflicting bits must be flipped by the BFL, while matching bits must not be flipped. It is irrelevant whether don't care bits are flipped or not.

In Step 3, a useless pseudo-random pattern is assigned to each deterministic pattern. In order to obtain an efficient implementation of the BFL, the pseudo-random pattern with the minimum number of conflicting bits is selected. If several such patterns exist, the pattern is chosen which minimize the number of:

a) scan chains containing both matching and conflicting bits

b) clock cycles during which matching and conflicting bits are shifted into the scan chains

The cost function a) attempts to minimize the BFL size per scan chain. The cost function b) has a lower priority and attempts to maximize the logic sharing among the BFL's corresponding to different scan chains.

After mapping all the deterministic patterns to the pseudo-random sequence, the BFL is generated in Step 4 by formulating an instance of logic synthesis with don't cares [25] and solving it using a BDD-based procedure. BFL takes the state of the DLBIST hardware (defined as $LFSR \times PC \times BC$, where $LFSR$ is the state of the LFSR, PC is the value of the pattern counter and BC is the value of the bit counter) and maps it to 1 (for a conflicting bit), 0 (for a matching bit that must not be flipped) or don't care (for don't care bits). This instance is represented as a pair of BDDs, transformed into an RTL description and synthesized by a commercial logic synthesis tool.

3 Resistive Bridging Fault Model

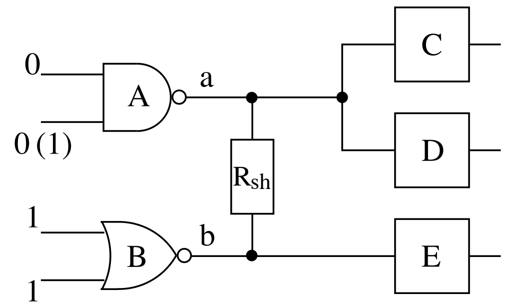

Figure 2: Example circuit

In this section, we provide a brief overview of the resistive bridging fault (RBF) model, which is used as a surrogate of unmodeled defects in this paper. The material here is restricted to concepts necessary for understanding the analysis in this paper; [26] gives an in-depth consideration.

The main difficulty when dealing with resistive faults is that, unlike for the non-resistive case, there is an unknown

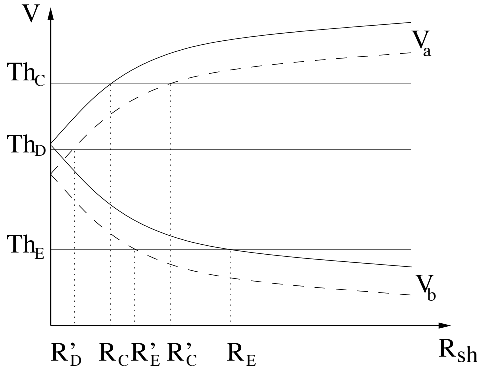

Figure 3: R_{sh}-V-diagram

value to be taken into account, the resistance. This is because it cannot be known in advance which particle will cause the short defect corresponding to the bridge, as parameters like its shape, size, conductivity, exact location on the die, evaporation behavior, electromigration and the temperature of its environment can influence the resistance of the short defect. A short defect may be detected by a test pattern for one resistance value, and the short between the same nodes may not be detected by the same pattern for another resistance. This fundamentally changes the meaning of standard testing concepts, like redundancy, fault coverage, and so forth.

In order to handle this ambiguity, Renovell et al. [27, 28] introduced the concept of *Analogue Detectability Interval* (ADI) and probabilistic fault coverage. In the following, we will illustrate the model by means of an example.

Consider the circuit in Figure 2. The lines *a* and *b* are bridged, with *a* (*b*) being the output of a NAND2 (NOR2) gate. Let us first assume that the applied pattern is 0011. In CMOS, two p transistors from the pull-up network of gate A (connected in parallel) drive node a, and two n transistors (also in parallel) from the pull-down network of gate B drive node b. Thus, in absence of the bridge there will be a 1 on *a* and a 0 on *b*. In presence of the bridge, the voltage V_a on *a* and the voltage V_b on *b* both depend on the bridge resistance R_{sh}. For $R_{sh} = 0\Omega$, there will be some intermediate voltage identical for both lines. For $R_{sh} = \infty$, V_a will equal V_{DD} and V_b will equal 0V, as if the bridge was not present. A possible voltage distribution for intermediate values of R_{sh} (those between 0Ω and ∞) is depicted by solid curves in Figure 3. The X-axis corresponds to different values of R_{sh}, the Y-axis shows which voltages are assumed on the lines *a* and *b* if the bridge has such a resistance. With increasing R_{sh}, V_a and V_b diverge, with V_a approaching V_{DD} and V_b approaching 0.

The gates succeeding the bridge (gates C and D are successors of *a* and gate E is successor of *b*) will interpret these voltages as a logic value 1 or a logic value 0, depending on their *input threshold* (switching threshold of input). In accordance to previous works, we assume an exact-defined threshold voltage Th, which however may be different for different gate types.

In Figure 3, the thresholds for gates C, D, E are shown as horizontal lines labeled by Th_C, Th_D and Th_E, respectively. Consider gate C. Given a resistance R_{sh}, this gate will either interpret the value on *a* as logic-0 (for $R_{sh} < R_C$) or as logic-1 (for $R_{sh} > R_C$): for a bridge with low resistance, the value 0 on the line *b* has larger impact on the voltage on *a* than for a highly-resistive bridge. Hence, the RBF is detected on the output of gate C iff $R_{sh} \in [0, R_C]$. For gate D, the threshold Th_D is below the curve. This means that for any R_{sh} gate D will recognize the voltage on *a* as logic value 1. The fault is not detectable for any value of R_{sh}. For gate E, the solid curve V_b is relevant. E interprets the voltage on *b* as faulty logic value (1) only for $R_{sh} \in [0, R_E]$.

Overall, the fault can be detected at the output of C iff $R_{sh} \in [0, R_C]$, at the output of D iff $R_{sh} \in \emptyset$ (i.e. for no value of R_{sh}) and at the output of E iff $R_{sh} \in [0, R_E]$. The fault effect is visible at one of the outputs iff $R_{sh} \in [0, R_C] \cup \emptyset \cup [0, R_E] = [0, R_E]$. The interval $[0, R_E]$ (in which the fault is detected at (at least) one output) is called *Analogue Detectability Interval (ADI)* of the pattern 0011. In contrast to fault simulation for 'classical' fault models (which determine for a fault whether it has been detected or not), RBF simulation determines for a fault and a test pattern the ADI, i.e. *for which values of bridge resistance* the fault has been detected. If the ADI is empty, then the fault is not detected for any R_{sh}.

Now imagine that there is a logic value 1 on the second input of the NAND gate (pattern applied is 0111). Then, only one p transistor will pull up the voltage on the line a to the power supply. This results in logic-1 being driven with less strength on a. With logic-0 driven on b with the same strength as before (two parallel n transistors), the voltage characteristic for V_a and V_b in the R_{sh}-V-diagram will be described by curves situated underneath the original ones (one possibility is shown by the dashed curves). This results in new detection conditions: $R_{sh} \in [0, R'_C]$ at the output of C, $R_{sh} \in [0, R'_D]$ at the output of D (note that this interval has been empty for the pattern 0011), and $R_{sh} \in [0, R'_E]$ at the output of E. The ADI for 0111 is $[0, R'_C] \cup [0, R'_d] \cup [0, R'_E] = [0, R'_C]$. So, a RBF with $R_{sh} \in [R'_C, R_E]$ is detected by the pattern 0011 but not by 0111, although the logic values on the lines a and b in the fault-free circuit are identical for these two patterns (pattern-dependency).

C-ADI of a test set (C stands for 'covered') is defined as the union of the ADIs of individual test patterns. G-ADI (G means 'global') is the C-ADI of the exhaustive test set. Hence, C-ADI includes all the bridge resistances for which the fault has been detected by at least one test pattern, while G-ADI consists of all values of R_{sh} for which the fault is *de-*

Circuit	1K	5K	10K
c7552	92.38	93.51	94.68
cs09234	72.31	80.79	83.60
cs13207	76.56	86.76	91.47
cs15850	84.58	89.98	91.14
cs38417	86.23	90.57	92.61
cs38584	90.47	93.44	94.31

Table 1: Stuck-at coverage of pseudo-random sequences before deterministic pattern embedding

tectable. If C-ADI of a test set equals G-ADI, then this test set is as effective in detecting RBF as the exhaustive test set. A bridging fault with resistance not in G-ADI is redundant.

The *global fault coverage* FC [28, 26] is defined as

$$FC(f) = 100\% \cdot \left(\int_{C\text{-ADI}} \rho(r)dr \right) / \left(\int_{G\text{-ADI}} \rho(r)dr \right), \quad (1)$$

where $\rho(r)$ is the probability density function of the short resistance r obtained from manufacturing data. Thus, FC relates C-ADI to G-ADI, weighted by the likelihood of different values of R_{sh}.

4 Experimental Results

The embedding procedure has been applied to pseudo-random test sequences of 1,000, 5,000, and 10,000 test patterns. Deterministic patterns with don't cares were generated by a stuck-at ATPG procedure. The fault coverage achieved by the pseudo-random sequence (before deterministic pattern embedding) is quoted in Table 1. Note that after the embedding, the sequence detects all irredundant stuck-at faults not aborted by the ATPG tool. The results are quoted for the ISCAS 85 and the combinational cores of the ISCAS 89 circuits (indicated as "cs") for which the pseudo-random sequence of length 10K did not detect all the target faults. (For other ISCAS circuits, no pattern embedding is required for 10K patterns).

We performed resistive bridging fault (RBF) simulation for pseudo-random test sequences before and after deterministic test embedding. The fault set consisted of 10,000 randomly selected non-feedback resistive bridging faults. We employed the density function ρ derived from one used in [29] for all experiments. All measurements were performed using the simulator from [26]. The SAT-based ATPG procedure from [30] was used for computing the exact value of G-ADI. Recall that the embedding procedure considers only stuck-at fault detection; hence resistive bridging faults are a valid surrogate for non-target defects.

The results are summarized in Table 2. Resistive bridging fault coverage of the pseudo-random test sequence before deterministic pattern embedding (Random RBFC), RBF coverage of the sequence after the embedding (Embedded RBFC) and the size of the synthesized bit-flipping logic (BFL) in gate equivalents (Embedded LSIZE) are given for the circuits mentioned above and the three test sequence lengths 1K, 5K and 10K. Note that Random RBFC corresponds to a scenario in which no BFL is employed at all (LSIZE = 0).

First of all, one can see that the RBF coverage of the pseudo-random sequences is consistently higher than their stuck-at coverage. Interestingly, pseudo-random pattern resistant faults seem to be distributed differently for stuck-at and resistive bridging faults. It appears that there are two circuits with many pseudo-random pattern resistant RBFs (cs09234 and cs38584) compared to other circuits. While cs09234 has also the lowest stuck-at fault coverage, cs38584 has the second highest stuck-at coverage. Hence, the validity of stuck-at fault coverage in identifying circuits with many pseudo-random pattern resistant RBFs appears to be limited.

The results clearly demonstrate the importance of the embedded deterministic patterns, as the RBF coverage always increases significantly due to embedding. However, the pseudo-random patterns also seem to contribute to detecting non-target defects. This is implied by the fact that applying more pseudo-random patterns results in a higher RBF coverage. It is seen best for the two circuits with a large number of hard-to-detect RBFs (cs09234 and cs38584) for which the coverage gain from 1K to 5K is around 0.7%. Note that the circuits for which the sequence yielded a good RBF coverage before embedding also have the highest RBF coverage (compared to the other circuits) after embedding.

Finally, the longer sequences require less logic (up to a factor of 2.4 for cs13207). This is due to two facts: first, the pattern embedding procedure has more degrees of freedom that it can exploit. Second, more stuck-at faults are covered by the pseudo-random sequence before embedding. These faults do not have to be targeted during BFL synthesis. Overall, there is a three-dimensional tradeoff: longer DLBIST sequence means a larger test application time, but also less area cost. Moreover, it has an enhanced coverage of non-target defects.

5 Conclusions

Deterministic Logic BIST is a technique that combines a complete stuck-at coverage with relatively compact on-chip logic and is capable to apply a large number of pseudo-random patterns with hardware speed and without using the slow tester interface. We studied for the first time how effective it is in detecting non-target defects. We used resistive bridging faults as a model of non-target defects. They accurately represent pattern dependency, Byzantine behavior and other complex phenomena that are not regarded by the stuck-at fault model. Experimental results showed that both the deterministic patterns and the pseudo-random sequence

Circuit	1K			5K			10K		
	Random RBFC	Embedded RBFC	Embedded LSIZE	Random RBFC	Embedded RBFC	Embedded LSIZE	Random RBFC	Embedded RBFC	Embedded LSIZE
c7552	99.28	99.83	583	99.44	99.87	546	99.61	99.87	433
cs09234	90.68	98.55	1097	95.30	99.26	824	96.55	99.39	683
cs13207	95.58	99.31	889	97.62	99.66	541	98.53	99.70	367
cs15850	96.29	99.36	1107	98.34	99.67	783	98.81	99.70	686
cs38417	97.50	99.46	4135	98.57	99.54	3170	98.93	99.65	2697
cs38584	93.01	98.74	894	95.10	99.43	878	96.47	99.67	590

Table 2: Experimental results

are useful for detecting non-target defects and that increasing the length of the sequence enhances their coverage. This suggests that DLBIST is a better way to apply the patterns than just loading them from the tester. An other implication is that DLBIST should exploit the complete available time and apply the longest possible sequences for best defect detection. As a side effect, the size of the required bit-flipping logic is reduced.

The conclusions of this work should be validated on actual silicon. Of special interest would be a comparison with hand-generated functional test patterns which are often claimed to be more effective in detecting actual defects than ATPG patterns.

Acknowledgment

Parts of this research work were supported by the German Federal Ministry of Education and Research (BMBF) in the Project AZTEKE under contract number 01M3063C.

6 References

[1] H.-J. Wunderlich. BIST for systems-on-a-chip. *INTEGRATION, the VLSI Jour.*, 26(12):55–78, December 1998.
[2] P.H. Bardell, W.H. McAnney, and J. Savir. *Built-In Test for VLSI: Pseudorandom Techniques*. John Wiley & Sons, 1987.
[3] Michael Gössel and Steffen Graf. *Error Detection Circuits*. McGraw-Hill, London, 1993.
[4] J.P. Hayes and A.D. Friedman. Test point placement to simplify fault detection. *IEEE Trans. on Comp.*, C-33(7):727–735, 7 1974.
[5] H.-J. Wunderlich. On Computing Optimized Input Probabilities for Random Tests. In *Design Automation Conf.*, pages 392–398, Oct. 1987.
[6] W. Wang and S.K. Gupta. Weighted random robust path delay testing of synthesized multilevel circuits. In *VLSI Test Symp.*, pages 291–297, 1994.
[7] A. Krasniewski and S. Pilarski. Circular self-test path: A low-cost BIST technique for VLSI circuits. *IEEE Trans. on CAD*, 8(1):46–55, 1989.
[8] S. Pilarski and A. Perzyńska. BIST and delay fault detection. In *Int'l Test Conf.*, pages 236–242, 1993.
[9] C.V. Krishna, A. Jas, and N.A. Touba. Test vector encoding using partial LFSR reseeding. In *Int'l Test Conf.*, pages 885–893, 2001.
[10] N. Zacharia, J. Rajski, and J. Tyszer. Decompression of test data using using variable-length seed LFSRs. In *VLSI Test Symp.*, pages 426–433, 1995.
[11] 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 register. *IEEE Trans. on Comp.*, 44(2):223–233, February 1995.
[12] S. Hellebrand, S. Tarnick, B. Courtois, and J. Rajski. Generation of vector patterns through reseeding of multiple-polynomial linear feedback shift registers. In *Int'l Test Conf.*, pages 120–129, 1992.
[13] K. Chakrabarty, B.T. Murray, and V. Iyengar. Built-in test pattern generation for high performance circuits using twisted-ring counters. In *VLSI Test Symp.*, pages 22–27, 1999.
[14] S. Hellebrand, H.G. Liang, and H.J. Wunderlich. A mixed-mode BIST scheme based on reseeding of folding counters. *Jour. of Electronic Testing: Theory and Applications*, 17(3-4):159–170, February 2001.
[15] B. Wurth and K. Fuchs. A BIST Approach to Delay Fault Testing with Reduced Test Length. In *European Design & Test Conf.*, pages 418–423, 1995.
[16] I. Polian and B. Becker. Scalable delay fault BIST for use with low-cost ATE. *Jour. of Electronic Testing: Theory and Applications*, 20(2):181–197, 4 2004.
[17] H.-J. Wunderlich and G. Kiefer. Bit-flipping BIST. In *Int'l Conf. on CAD*, pages 337–343, 1996.
[18] G. Kiefer and H.-J. Wunderlich. Deterministic BIST with multiple scan chains. In *Int'l Test Conf.*, pages 1057–1064, 1998.
[19] N.A. Touba and E.J. McCluskey. Altering a pseudo-random bit sequence for scan based bist. In *Int'l Test Conf.*, pages 649–658, 1996.
[20] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee. Embedded deterministic test. *IEEE Trans. on CAD*, 23(5):776–792, 5 2004.
[21] R.C. Aitken. Finding defects with fault models. In *Int'l Test Conf.*, pages 498–505, 1995.
[22] R. Aitken. New defect behavior at 130nm and beyond. In *European Test Symposium (Emerging Ideas Contribution)*, pages 279–284, 2004.
[23] Y. Tang, H.-J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, P. Engelke, I. Polian, and B. Becker. X-masking during logic bist and its impact on defect coverage. In *Int'l Test Conf.*, pages 442–451, 2004.
[24] V. Gherman, H.J. Wunderlich, H. Vranken, F. Hapke, M. Wittke, and M. Garbers. Efficient pattern mapping for deterministic logic BIST. In *Int'l Test Conf.*, 2004.
[25] O. Coudert, C. Berthet, and J.C. Madre. Verification of synchronous sequential machines based on symbolic execution. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, pages 365–373. Springer Verlag, 1989.
[26] P. Engelke, I. Polian, M. Renovell, and B. Becker. Simulating resistive bridging and stuck-at faults. In *Int'l Test Conf.*, pages 1051–1059, 2003.
[27] M. Renovell, P. Huc, and Y. Bertrand. The concept of resistance interval: A new parametric model for resistive bridging fault. In *VLSI Test Symp.*, pages 184–189, 1995.
[28] M. Renovell, F. Azaïs, and Y. Bertrand. Detection of defects using fault model oriented test sequences. *Jour. of Electronic Testing: Theory and Applications*, 14:13–22, 1999.
[29] C. Lee and D. M. H. Walker. PROBE: A PPSFP simulator for resistive bridging faults. In *VLSI Test Symp.*, pages 105–110, 2000.
[30] P. Engelke, I. Polian, M. Renovell, and B. Becker. Automatic test pattern generation for resistive bridging faults. In *European Test Symp.*, pages 160–165, 2004.