Hybrid BIST Using an Incrementally Guided LFSR

C.V. Krishna and Nur A. Touba
Department of Electrical and Computer Engineering
Computer Engineering Research Center
University of Texas, Austin, TX 78712-1084

E-mail: {krishna, touba}@ece.utexas.edu

Abstract

A new hybrid BIST scheme is proposed which is based on using an "incrementally guided LFSR." It very efficiently combines external deterministic data from the tester with on-chip pseudo-random BIST. The hardware overhead is very small as a conventional STUMPS architecture [1] is used with only a small modification to the feedback of the LFSR which allows the tester to incrementally guide the LFSR so that it can embed patterns that detect the random-pattern-resistant faults in the pseudo-random sequence. Compared with external testing, the proposed approach achieves dramatic reductions in tester storage requirements while using very simple on-chip hardware. Results indicate that the proposed approach provides very attractive tradeoffs between test length and tester storage requirements.

1. Introduction

The test data volume required for testing increasingly complex system-on-chip (SOC) designs continues to grow rapidly and is outstripping the capabilities of ATE (automated test equipment). One well-known solution to this problem is to use built-in self-test (BIST). BIST involves performing test pattern generation and output response compaction on the chip. The most economical BIST schemes are based on pseudo-random pattern testing. The problem with pseudo-random pattern testing, however, is that it generally does not provide high enough fault coverage due to the presence of random-pattern-resistant (r.p.r.) faults [5]. There are two solutions to this problem. One is to modify the circuit to eliminate the random pattern resistance by inserting test points [5], and the other is to modify the test pattern generator by adding additional hardware to generate patterns that detect the hard faults [6, 10, 11, 15]. Both approaches have significant drawbacks. Test point insertion requires modifying the function logic which can degrade system performance, and modifying the test pattern generator can require large amounts of additional silicon area.

In order to reduce the hardware overhead for BIST, a hybrid approach between external testing and BIST can be taken. A "hybrid BIST" scheme is one in which some external data from the tester is combined with the BIST hardware to achieve the desired fault coverage. A hybrid BIST scheme reduces the test data stored on the tester compared with full external testing, but it does not require as much hardware overhead as full stand-alone BIST.

A simple hybrid BIST scheme is to use a STUMPS architecture [1] to apply pseudo-random patterns to detect the random pattern testable faults, and then use deterministic vectors from the tester to detect the random pattern resistant (r.p.r.) faults. Results for using this scheme on large industrial designs have shown that it only achieves around a 30-50% reduction in tester storage requirements compared with conventional external testing [8, 14]. More sophisticated hybrid BIST schemes have been developed including using hybrid patterns [2], folding counters [7], two-dimensional compression [13], weighted pattern testing [9], and RESPIN [3, 4].

Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03)
1063-6722/03 $17.00 © 2003 **IEEE**

In this paper, a new hybrid BIST scheme is proposed which is based on using an "incrementally guided LFSR." The hardware overhead is very small. A conventional STUMPS architecture is used with only a small modification to the feedback of the LFSR which allows the tester to incrementally guide the LFSR so that it can embed patterns that detect the r.p.r. faults in the pseudo-random sequence. Compared with external testing, the proposed approach achieves dramatic reductions in tester storage requirements while using very simple on-chip hardware. Results indicate that the proposed approach provides very attractive tradeoffs between test length and tester storage requirements.

2. Overview of Proposed Scheme

The proposed scheme embeds deterministic test cubes (test vectors where the unspecified bits are left as don't cares) that detect the r.p.r. faults in the pseudo-random sequence generated by an LFSR. The architecture of the proposed scheme is shown in Fig. 1. A conventional STUMPS [1] architecture is used except that an extra exclusive-or (XOR) gate is added in the feedback of the LFSR. The extra exclusive-or gate is controlled by the tester and allows the tester to incrementally guide the LFSR. Some control logic is required to obtain data from the tester only during the first few clock cycles of every test vector generation. In order to be able to generate the test cubes with a high degree of probability, the LFSR length, r, is at least s_{max}+ 20 where s_{max} is the maximum number of specified bits in any test cube. The r-bit LFSR can be initialized with a starting r-bit state, or it can start from the reset state. Since the length of the scan chains is m-bits, the LFSR needs m clock cycles to generate a test vector within the scan chains. During the first clock cycle while a test vector is being shifted into the scan chains from the LFSR, one bit of deterministic data is obtained from the tester and XORed with the feedback of the LFSR. During the remaining m-1 clock cycles while the LFSR is filling the scan chains with the test vector, the tester does not shift any data into the LFSR. This process is repeated for all the test vectors that are generated by the LFSR. So the test data storage requirement is one bit per test vector applied to the CUT. While the LFSR shifts in the test vector, the output response contained in the scan chains is shifted out into a multiple-input signature register (MISR).

The one deterministic bit coming in from the tester for each test vector can be chosen to be either a zero or a one. It is essentially a "free-variable" whose value can be chosen in a way that incrementally guides the LFSR towards a state in which it will produce a required test cube. The impact of each free-variable on the subsequent test vectors that are applied to the CUT can be determined by symbolic simulation of the LFSR.

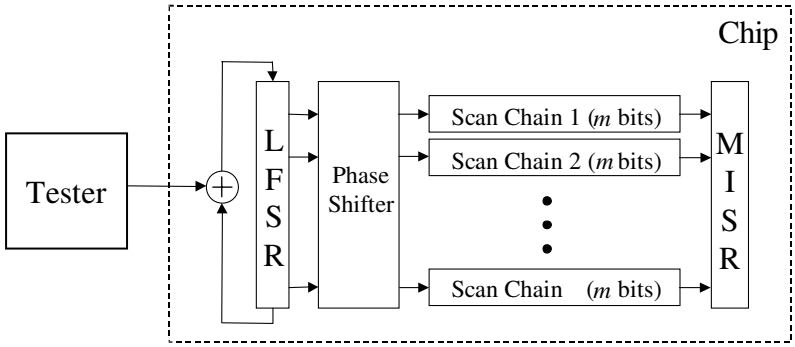

Figure 1. Architecture for Pseudo-Random BIST with Incrementally Guided LFSR

Consider the example in Figs. 2-4. Assume that the LFSR starts in state *1011*, and one bit comes in from the tester which is represented by the free-variable *a*. The LFSR then runs in autonomous mode to fill the scan chains with the first test vector. The first test vector is computed by symbolic simulation and is shown in Fig. 3. Then the next bit comes in from the tester which can be represented by the free-variable *b*. The LFSR is then symbolically simulated to determine the second test vector (which now is in terms of both free-variables *a* and *b*). This process continues as more and more free-variables are introduced. At some point, it may be possible to embed a test cube for an r.p.r. fault by assigning values to some of the free-variables. For example, the test cube X1XX, X1X0, 1X01 could be embedded in the third test vector by assigning the values *a=1* and *b=0*. Note that the equations for each bit in the test vector are linear (sum modulo 2), so finding an assignment to the free-variables to embed the test cube can be solved very efficiently using a linear equation solver [12]. By using the computed values for the free-variables, the data from the tester incrementally guides the LFSR into a state in which the test cube for the hard fault gets applied to the CUT. This process of adding more free-variables and embedding additional test cubes for hard faults can be repeated until sufficiently high fault coverage is achieved. This is the basic approach for BIST based on an incrementally guided LFSR.

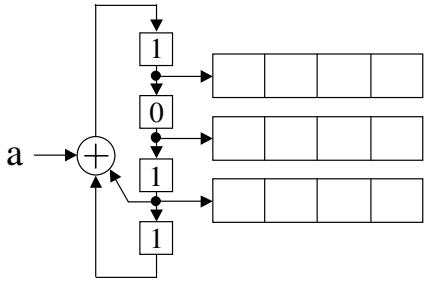

Figure 2. LFSR Configuration for Example

Vector 1			
a'	1	1	a
1	1	a	1
1	a	1	0

Vector 2			
b	a	0	a'+b
a	0	a'+b	a'
0	a'+b	a'	1

Vector 3			
a'+c	a+b	a	a'+b+c
a+b	a	a'+b+c	b
a	a'+b+c	b	a

Figure 3. Symbolic Simulation of LFSR in Fig. 2

Vector 1			
0	1	1	1
1	1	1	1
1	1	1	0

Vector 2			
0	1	0	0
1	0	0	0
0	0	0	1

Vector 3			
c	1	1	c
1	1	c	0
1	c	0	1

Figure 4. Vectors after Assignment of a=1 and b=0.

The approach is very efficient because there are a lot of degrees of freedom for embedding the test cubes for the r.p.r. faults. The degrees of freedom are the following: selecting the values of the free-variables, selecting which test cube to embed, and selecting which test vector in which to embed the test cube.

So far we have been assuming that only one bit of data will come in from the tester for each test vector. However, this may result in too long of a test length for this BIST scheme. The test length for this BIST scheme can be reduced by increasing the number of channels by which data

is brought in from the tester. For example, if we used 3 channels, we could bring in 3 bits of data from the tester during the first clock cycle each time a test vector is shifted into the scan chains. This would increase the rate at which free-variables are introduced and hence reduce the overall BIST test length because the test cubes for the hard faults could be embedded sooner. The degree of freedom in choosing which test vector in which to embed each test cube would be reduced since the test length is reduced, and hence the encoding efficiency would be slightly degraded. However, this provides a very easy way for trading off test data bandwidth requirements, test data storage requirements, and test length.

Note that the proposed scheme can also be used to implement traditional "stand-alone" BIST where no data comes from the tester. In this case, a ROM would be used. The ROM would supply the deterministic data to guide the LFSR instead of having that data coming from an external tester.

3. Determining Data on Tester

Given the set of test cubes for the r.p.r. faults and the maximum allowable test length (L), this section describes how to obtain the data that is stored on the tester. Since the test cubes will be embedded within a pseudo-random sequence of test vectors, only the test cubes for the r.p.r. faults are required.

An important parameter that needs to be determined for the proposed scheme is the number of bits of data that will come in from the tester for each test vector. If n bits of data need to be brought from the tester for each test vector, then n channels are required between the tester and the LFSR in order to XOR these n bits of data with the contents of the LFSR during the first clock cycle. These n bits can be XORed with the contents of the LFSR at regularly spaced tap points within the LFSR.

The amount of storage required on the tester for storing the test data can be obtained from the test length and the number of channels from the tester. For a test length of L vectors with n bits coming from the tester per vector, the number of bits that need to be stored on the tester is $L*n$. This provides an upper bound on the amount of tester storage. If the test cubes can be embedded in less than L vectors, then the amount of tester storage is reduced.

Given the set of test cubes T and the test length L, an initial estimate can be found for the number of bits n to be used per test vector. Given the total number of specified bits within the set of test cubes, the initial estimate of n is given by:

$$n = Max[\ \lfloor Total\ number\ of\ specified\ bits\ /\ Test\ Length \rfloor,\ 1\ ]$$

Based on this initial estimate of n, the proposed scheme tries to embed the test cubes for the r.p.r. faults within the maximum allowable test length L. If the test cubes cannot be embedded within the test length L, then more data needs to be introduced per vector in order to embed the given test cubes. Thus the value of n has to be increased, and this process is repeated with the higher value of n. If the test cubes can be embedded within L test vectors, then fault simulation is done in order to ensure that this sequence of vectors with embedded test cubes provides sufficient fault coverage. If the fault coverage is not satisfactory, then automatic test pattern generation (ATPG) can be performed on the undetected faults to obtain a set of *top-up* cubes. The T test cubes and the *top-up* cubes are then together embedded within the given test length L. Once this is done, the amount of test data that needs to be stored on the tester is given by $L*n$.

The steps of this procedure are as follows:

1. Fault simulate L pseudo-random vectors
2. Do ATPG for faults not detected by the pseudo-random vectors to obtain set of test cubes, T
3. Embed test cubes T with n channels to obtain embedded vectors, L'

4. If $|L'| > |L|$, then increment n and go to step 3
5. Fault simulate L' embedded vectors
6. If the fault coverage is satisfactory, then the procedure is done.
7. Do ATPG for faults not detected by L' and add the resulting test cubes to the set T
8. Go to step 3

The procedure iterates until the fault coverage is satisfactory. Generally this requires only one iteration.

4. Improving Compression Using Lookahead

The compression achieved using the basic hybrid BIST scheme described earlier can be improved further by performing a *lookahead* while generating the test data. The idea is that the data from the tester is used to guide the LFSR towards a test cube in a manner that also facilitates the embedding of subsequent test cubes. Thus once a test cube has been embedded, the subsequent test cube can be embedded within fewer cycles than the previous approach. The previous approach is equivalent to using a *lookahead* of 0, since none of the subsequent test cubes are considered while generating the current test cube. A *lookahead* of k means that the equations for the subsequent k test cubes are considered while solving the equations for the current test cube.

With this feature, the degrees of freedom in the solution space of the linear equations for a test cube t_i are used to help solve for a subsequent test cube t_{i+1} since the free-variables occurring in the equations of t_i are used to ease the problem of finding a solution for t_{i+1}. Since the subsequent test cube is being embedded using fewer cycles, fewer bits have to be stored on the tester to guide the LFSR towards t_{i+1}. This helps to improve the compression achieved using this BIST scheme.

This procedure for a *lookahead* of k is described as follows. As explained in Fig. 4, each test vector is determined by a symbolic simulation of the LFSR using the free-variables from the tester. After it is determined that a test cube t_i can be embedded by solving the linear equations, an assignment of values to the free-variables is not done immediately. Instead, more free-variables are introduced until another k test cubes can be embedded within the vectors generated by the LFSR. This is done by concatenating the equations for the k test cubes with the equations for t_i to form a system of linear equations. A solution to this system of equations indicates that it is possible to embed the subsequent k test cubes using all the free-variables introduced until then. Based on this solution, an assignment of values is done to *only the free-variables that occur in the equations for* t_i. The values assigned to these free-variables are then used to update the equations for the subsequent k test cubes as well as the state of the LFSR. Thus the assignment of values to the free-variables occurring in t_i is done in a manner that helps to solve for the next k test cubes.

The concept behind this can be shown with a simple example. Let a and b be two free-variables occurring in the equations for test cube t_i, and {01,10} be two possible assignments to these variables. For a *lookahead* of 0, either of these two assignments can be used. But if a *lookahead* greater than 0 is used, it is possible that assigning 01 to the variables might help to embed test cube t_{i+1} earlier than would have been possible by using the assignment 10. This degree of freedom is utilized by concatenating and solving together the equations of t_i and t_{i+1}. This ensures that the variables in t_i are assigned values which also help to solve the equations of t_{i+1} and thus guide the LFSR towards generating t_{i+1} earlier than would have been possible with a *lookahead* of 0.

5. Experimental Results

Experiments were performed on the largest ISCAS 89 benchmark circuits. Given different values of *n* (the number of bits that are used per vector), the proposed hybrid BIST scheme was used to embed test cubes in a pseudo-random sequence to detect all non-redundant faults. The resulting test length and tester storage requirements are graphed in Fig. 5 in a log-log plot. A *lookahead* of 0 was used in generating these results. As can be seen, the proposed scheme provides a tradeoff between test length and tester storage requirements. As the test length increases, the tester storage requirements are reduced. As *n* is increased, the test length reduces towards that of conventional external deterministic testing. As *n* is decreased, the tester storage requirements drop and if the test length were made long enough to detect all faults, the tester storage requirements would go to zero as in conventional stand-alone BIST. A designer can determine what test length is desired and use the proposed scheme to minimize the tester storage requirements for that test length. This is achieved with very little hardware overhead beyond what is needed for a STUMPS architecture.

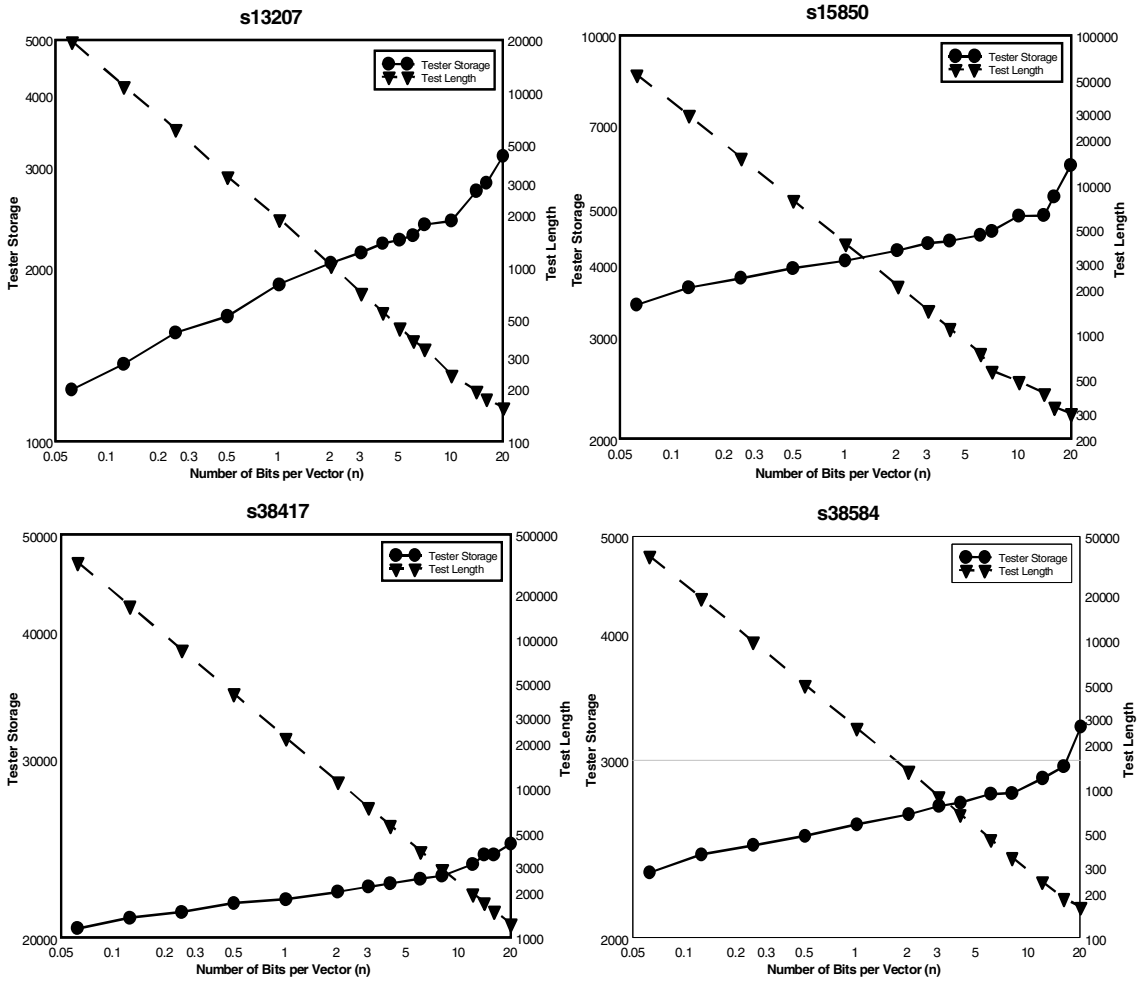

Figure 5. Graphs of Tester Storage and Test Length Versus Number of Bits from Tester Used per Vector for ISCAS Benchmark Circuits

Table 1 shows results for the proposed scheme for different values of *lookahead*. The number of scan elements is shown for each circuit. The next column shows the different number of bits per vector (*n*) that are used for each circuit. If *n* is less than 1, it indicates that a single bit

is received from the tester every (1/*n*) test vectors. Thus the value *n*=0.25 means that at the beginning of every fourth test vector, a single bit from the tester is XORed with the contents of the LFSR. For each circuit, results are shown for three different values of *n* using a *lookahead* of 0, 1, and 4. For each combination of *n* and the *lookahead* factor, the test length (L) required to embed the top-up cubes is shown, followed by the amount of data that needs to be stored on the tester. As can be seen, using a *lookahead* of 1 and 4 did reduce the storage requirements for some of the circuits, though not significantly.

Table 1. Results for Proposed Scheme using different values for *lookahead*

Circuit		Bits per Vector (*n*)	*lookahead* = 0		*lookahead* = 1		*lookahead* = 4	
Name	Scan Elements		Test Len (*L*)	Tester Storage	Test Len (*L*)	Tester Storage	Test Len (*L*)	Tester Storage
s13207	700	0.25	6,104	1,526	6,078	1,519	6,078	1,519
		1	1,856	1,856	1,818	1,818	1,818	1,818
		4	553	2,212	545	2,180	537	2,148
s15850	611	0.25	15,216	3,804	14,985	3,746	14,985	3,746
		1	4,124	4,124	4,115	4,115	4,115	4,115
		4	1,103	4,412	1,096	4,384	1,096	4,384
s38417	1664	0.25	85,093	21,273	85,093	21,273	85,093	21,273
		1	21,855	21,855	21,846	21,846	21,846	21,846
		4	5,623	22,492	5,598	22,392	5,582	22,328
s38584	1464	0.25	9,906	2,476	9,816	2,454	9,816	2,454
		1	2,592	2,592	2,562	2,562	2,562	2,562
		4	685	2,740	676	2,704	674	2,696

Table 2. Comparison of Results for Proposed Scheme with other Methods

Circuit		BIST followed by Top-up Vectors	RESPIN [3]	RESPIN + ATPG [4]	Proposed Hybrid BIST Scheme (*lookahead* = 0)			
Name	Scan Elements				Bits per Vector (*n*)	Test Len (*L*)	Tester Storage	% Compress
S13207	700	109,900	1,963	1,672	0.25	6,104	1,526	98.6
					1	1,856	1,856	98.3
					4	553	2,212	97.9
S15850	611	102,037	5,244	2,872	0.25	15,216	3,804	96.3
					1	4,124	4,124	95.9
					4	1,103	4,412	95.7
S38417	1664	565,760	31,656	8,412	0.25	85,093	21,273	96.2
					1	21,855	21,855	96.1
					4	5,623	22,492	96.0
S38584	1464	90,768	3,466	2,927	0.25	9,906	2,476	97.3
					1	2,592	2,592	97.1
					4	685	2,740	96.9

Table 2 shows a comparison of four different techniques. The first is simply using normal pseudo-random BIST to apply 10,000 pseudo-random patterns followed by top-up vectors (to detect the remaining undetected faults). The number shown is the storage requirements for the

deterministic top-up vectors. The second and third are the results for RESPIN which were taken directly from [3] and [4]. In [3], no special ATPG is used, however, in [4], a special ATPG tailored for RESPIN is used. Note that the test length and test storage for RESPIN are the same. Lastly, results are shown for the proposed scheme with a *lookahead* of 0. For each value of *n*, the test length (L) required to embed the top-up cubes is shown, followed by the amount of data that needs to be stored on the tester. Lastly, the percentage compression that is achieved using the proposed scheme compared with normal pseudo-random BIST followed by top-up vectors is shown. As can be seen, the storage required for the proposed scheme is less than the storage required for the basic RESPIN architecture in [3] for all the cases. When compared with the RESPIN plus tailored ATPG technique in [4], the proposed scheme shows better results for some of the circuits. One of the advantages of RESPIN is that it facilitates the use of a tailored ATPG, however, the runtime for that ATPG can be longer than that of conventional ATPG. The results for the proposed scheme were for using conventional ATPG.

6. Conclusions

The proposed hybrid BIST scheme provides a range of options in the continuum between external testing and stand-alone BIST for the designer to choose from. Based on tester storage requirements and the desired test length, the designer can select the one that best fits the particular test environment. The proposed scheme can also be used to implement conventional stand-alone BIST by storing the deterministic data in an on-chip ROM instead of on the tester.

Acknowledgements

This material is based on work supported in part by the Intel Corporation and in part by the National Science Foundation under Grant No. CCR-0306238.

References

[1] Bardell, P.H., and W.H. McAnney "Self-Testing of Multichip Logic Modules," *Proc. of International Test Conference*, pp. 200-204, 1982.

[2] Das, D., and N.A. Touba, "Reducing Test Data Volume Using External/LBIST Hybrid Test Patterns," *Proc. of International Test Conference*, pp. 115-122, 2000.

[3] Dorsch, R., and H.-J. Wunderlich, "Reusing Scan Chains for Test Pattern Decompression," *Proc. of European Test Workshop (ETW)*, May 2001.

[4] Dorsch, R., and H.-J. Wunderlich, "Tailoring ATPG for Embedded Testing," *Proc. of International Test Conference*, pp. 530-537, 2001.

[5] Eichelberger, E.B., and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test," *IBM Journal of Research & Development*, Vol. 27, No. 3, pp. 265-272, May 1983.

[6] Fagot, C., P. Girard, and C. Landrault, "On Using Machine Learning for Logic BIST," *Proc. of International Test Conference*, pp. 338-346, 1998.

[7] Hellebrand, S., H.-G. Liang, and H.-J. Wunderlich, "A Mixed Mode BIST Scheme Based on Reseeding of Folding Counters," *Proc. of International Test Conference*, pp. 778-784, 2000.

[8] Hetherington, G., T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan and J. Rajski, "Logic BIST for Large Industrial Designs: Real Issues and Case Studies," *Proc. of International Test Conference*, pp. 358-367, 1999.

[9] Jas, A., C.V. Krishna, and N.A. Touba, "Hybrid BIST Based on Weighted Pseudo-Random Testing: A New Test Resource Partitioning Scheme," *Proc. of VLSI Test Symposium,* pp. 2-8, 2001

[10] Kiefer, G., and H.-J. Wunderlich, "Deterministic BIST with Multiple Scan Chains," *Proc. of International Test Conference*, pp. 1057-1064, 1998.

[11] Kiefer, G., H. Vranken, E.J. Marinissen, and H.-J. Wunderlich, "Application of Deterministic Logic BIST on Industrial Circuits," *Proc. of International Test Conference*, pp. 105-114, 2000.

[12] Könemann, B., "LFSR-Coded Test Patterns for Scan Designs," *Proc of European Test Conf.*, pp. 237-242, 1991.

[13] Liang, H.-G., S. Hellebrand, and H.-J. Wunderlich, "Two-Dimensional Test Data Compression for Scan-Based Deterministic BIST," *Proc. of International Test Conference*, pp. 894-902, 2001.

[14] Pressly, M., D. Das, and C. Hunter, "LBIST for PowerPC™ Embedded Core Microprocessors: Feasible or Not?," *International Workshop on Microprocessor Test and Verification*, 1999.

[15] Touba, N.A., and E.J. McCluskey, "Altering a Pseudo-Random Sequence of Bits for Scan-Based BIST", *Proc. of International Test Conference*, pp. 167-175, 1996.