Response Compaction with any Number of Unknowns using a new LFSR Architecture*

Erik H. Volkerink [1,2] and Subhasish Mitra [2,3]

Agilent Laboratories [1]
Palo Alto, CA
Erik_Volkerink@Agilent.com

Intel Corporation [2]
Folsom, CA
Subhasish.Mitra@Intel.com

Center for Reliable Computing [3]
Stanford University
Stanford, CA

ABSTRACT

This paper presents a new test response compaction technique with any number of unknown logic values (X's) in the test response bits. The technique leverages an X-tolerant response compactor (X-compact), and forces X's that are not tolerated by X-Compact to known values. The data required to designate the X's not tolerated by the X-compactor, also called mask data, is stored in a compressed format on the tester and decompressed on-chip. We applied this technique to four industrial designs and obtained 26-fold to 60-fold reduction in test response data volume with no impact on test quality.

Categories and Subject Descriptors

B.8.1 [Integrated Circuits]: Reliability, Testing, and Fault Tolerance.

General Terms

Algorithms, Design, Economics, Reliability, Theory.

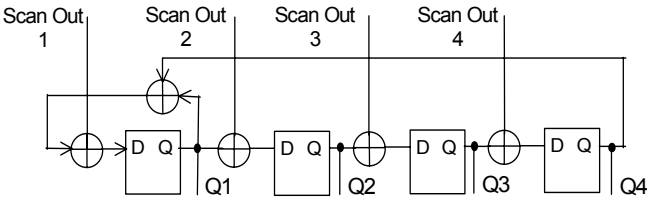
Keywords

VLSI Test, Compression, X-compact, LFSR, BIST.

1. INTRODUCTION

Boolean testing is performed by applying test patterns to an integrated circuit chip from the tester and observing the corresponding responses. A logic simulator is used to simulate the fault free design to obtain the responses expected from a fault-free chip for the applied test patterns. The tested integrated circuit chip passes the test if and only if all observed test response bits match the simulated fault-free test response bits. Unfortunately, for complex designs, logic simulators cannot accurately predict the logic values of all test response bits. This is due to the presence of un-initialized and uncontrollable bistables, bus contention, floating busses, multiple clock domains or simply because the simulation model is inaccurate. The test response bits whose logic values are not accurately predicted by the simulators are also called *unknown test response bits* or *Xs*.

A major problem arises when test responses are compacted using on-chip hardware. For example, classical signature analyzers such as Multiple Input Signature Registers (MISRs) [3, 5, 15] are probably the best response compactors. The major problem with classical signature analysis is that the signature can be corrupted in the presence of Xs. Figure 1 illustrates the problem. The outputs of four scan chains are connected to the inputs of the MISR. The initial MISR state is 0000. The states of the MISR during the first four clock cycles are shown in Fig. 1.



Clock Cycle	Scan Outputs				MISR State			
	Output 1	Output 2	Output 3	Output 4	Bit 1	Bit 2	Bit 3	Bit 4
1	1	0	0	0	1	0	0	0
2	1	0	0	X	0	1	0	X
3	0	0	X	1	X	0	X	1
4	0	1	X	0	X	X	X	X

Figure 1. Multiple Input Signature Register

Xs appearing at scan chain outputs corrupt the MISR contents. After four clock cycles, the expected MISR signature obtained from fault-free simulation consists of all X's. Signature bits whose expected values are X's are ignored during comparison of the expected signature with the actual signature. In this example, no comparison can be made.

This paper presents a technique to force the Xs to known values before compacting the test response data on-chip. The data required to identify the Xs, also called the *mask data*, is compressed and stored in the tester memory, and decompressed using on-chip hardware.

Application of this technique to four industrial designs results in 26-fold to 60-fold reduction in test response data volume without impact on test quality.

Section 2 presents previous work. Section 3 presents the basic technique. Section 4 presents the results. The paper ends with the conclusions in Section 5.

DAC 2005, June 13-17, 2005, Anaheim, California, USA.

* This work was done at Stanford CRC

2. PREVIOUS WORK

Test response data can be compressed by generating signatures using signature analyzers such as MISRs (Multiple Input Signature Registers). However, as discussed in Sec. 1, the major drawback of signature analysis is signature corruption in the presence of one or more Xs. Removal of all possible X sources through Design for Testability structures such as test points is not practical – often, the X sources cannot be identified until after the chips have already been manufactured. The other drawback of signature analysis is that the failing flipflops cannot be directly identified from the signature for diagnostics.

The X-Compact technique [8, 9] is a response compaction technique that guarantees detection of any erroneous test response in the presence of a limited number of Xs, also called the *tolerated Xs*. The X-Compactor is a combinational circuit designed with XOR gates, and the number of tolerated Xs determines the compactor design and the number of compactor outputs. Of course, the number of tolerated X's needs to be known during design time. Several publications discuss design of X-Compactors and X-tolerant signature analyzers that can tolerate several Xs [9]. The techniques presented in [13, 14, 21, 22, 23] also use a compactor tree that tolerates Xs. The techniques in [14, 21] use the X tolerant compactor circuitry and reduce the number of bits to be observed at the expense of the number of X's tolerated over multiple scan cycles. The technique in [23] introduces a graph-theoretic formulation, in contrast with the matrix-theoretic formulation. The response compaction technique in [16] uses a parity bit for every scan chain in addition to XOR-trees – however, a single X in a scan chain will corrupt the parity bit. The I-Compact technique [11] performs post-processing of compacted responses on the tester to identify defective parts. Although this is a promising approach, its application is limited by the number of Xs that can be handled, and the tester support required for post-processing of test responses.

Our test response compression technique is used together with an X-Compactor or a MISR. The technique guarantees detection of any erroneous test response in the presence of any number of Xs, even if the number of Xs exceed the number of Xs that can be tolerated by the X-Compactor. The fundamental concept behind our technique is to mask Xs in the test response using on-chip hardware. The *mask bits*, i.e., the bits used to identify the Xs, at a certain clock cycle form a *mask bit slice.* The mask bit slice masks an *output bit slice*, i.e., the scan chain outputs at a certain clock cycle. Figure 2 presents the basic architecture.

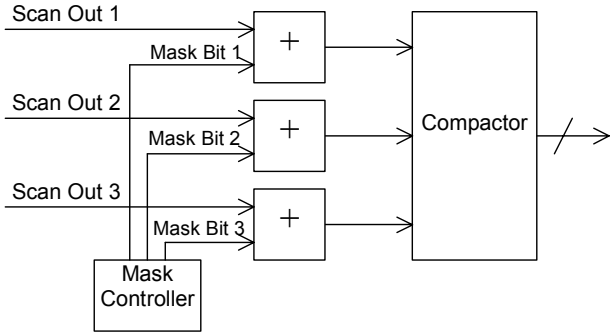


Figure 2. Basic Compression Architecture

Figure 3a presents example scan chain outputs. Figure 3b presents the corresponding mask bits.

	Clock Cycle			
	1	2	3	4
Scan Out 1	X	0	0	0
Scan Out 2	X	X	1	X
Scan Out 3	1	1	X	X

a) Example Scan Chain Outputs

	Clock Cycle			
	1	2	3	4
Mask Bit 1	1	0	0	0
Mask Bit 2	1	1	0	1
Mask Bit 3	0	0	1	1

b) Mask Bits Corresponding to Example (a)

Figure 3. Example Scan Chain Outputs & Mask Bits

We are not the first ones proposing bit masking (e.g., [2, 4, 10, 12, 17, 20]). However, the previous techniques also mask non-X responses.

In [10, 20] a technique was proposed, that masks unknown logic values on-chip and encodes the mask bits using the LFSR reseeding technique[5], see Fig. 4.

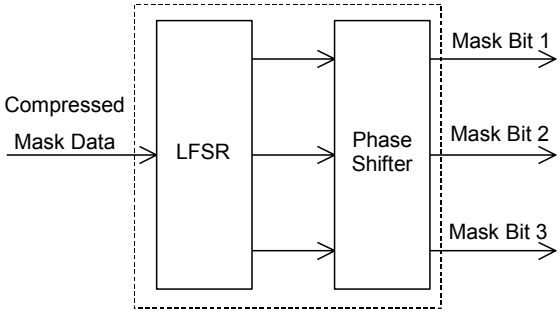


Figure 4. Generating Mask Bits using LFSR Reseeding

The phase-shifter is designed to minimize the probability of linear dependencies between the mask bits (see [1]).

The mask data compression algorithm computes the initial LFSR state (also called the *seed*) such that the LFSR generates a '1' for each X. The seed calculation algorithm encodes the Xs as '1' care bits and the non-X's as don't care bits (*d*) (see [5]).

Fig. 5a presents how the mask bits will be encoded in a LFSR seed. Fig. 5b presents the mask bits generated by the LFSR using the seed.

	Clock Cycle			
	1	2	3	4
Mask Bit 1	1	*d*	*d*	*d*
Mask Bit 2	1	1	*d*	1
Mask Bit 3	*d*	*d*	1	1

a) Encoding in Seeds (*d=don't care bit)*

	Clock Cycle			
	1	2	3	4
Mask Bit 1	1	1	0	0
Mask Bit 2	1	1	1	1
Mask Bit 3	0	1	1	1

b) Generated Mask Bits by the LFSR

Figure 5. Example Mask Bit Encoding using a LFSR (Three Non-X Responses are Masked)

A major problem is the fact that the LFSR assigns the don't care bits (*d*) to '1' with a probability of 50%. In other words, some non-X test responses may be masked, causing test escapes. For example, in 4b three non-X responses are masked out.

[10] reduces the probability of test escapes by generating additional patterns to improve fault coverage. [17] selects the non-X responses to be masked out intelligently using a stuck-at n-detection metric, such that the impact of masking on the defect coverage is reduced.

However, sole dependence on the SSF fault coverage may not be justified because a very small percentage of all defects behave as single stuck faults [7]. It was demonstrated in [18] that, in response to a test pattern, a defect might create errors in only a subset of flip-flops that are predicted to be erroneous by single stuck-at or transition fault simulation.

Compared to previous work, our technique:

1. Enables a very high compression ratio without masking *any* non-X responses.
2. Allows an even higher compression ratio by allowing masking some non-X test responses.
3. Does not increase the number of test patterns.
4. Does not depend on the used fault model.

The next section presents the basic technique.

3. BASIC TECHNIQUE

Figure 6 presents the basic technique. A mask bit is generated by the logical AND of *multiple* phase-shifter outputs, instead of only by *one* phase shifter output (see Fig. 4). The compression algorithm computes the seed such that all AND gate inputs are '1' during the clock cycles where a '1' mask bit is required.

The LFSR assigns the don't care bits in the AND gate input to '1' pseudo-randomly with a probability of 50%. However, since the AND gate output is '1' if and only if all AND inputs (*n*) are '1', the don't care bits are assigned to '1' pseudo-randomly with a probability of only 2^{-n}.

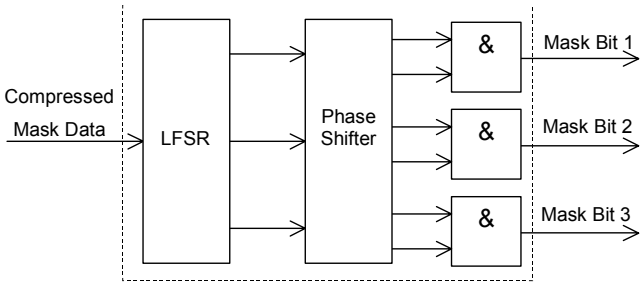


Figure 6. Generating Mask Bits using our Technique

For example, suppose we encode the mask bits presented in Fig. 3b using the new LFSR technique. The seed is computed by encoding the '1' mask bits by two '1' care bits instead of by only one care bit. Fig. 7 presents the mask bits generated by the LFSR using the calculated seed: Instead of masking three non-X responses (Fig. 5), the new technique masks no non-X responses!

	Clock Cycle			
	1	2	3	4
Mask Bit 1	1	0	0	0
Mask Bit 2	1	1	0	1
Mask Bit 3	0	0	1	1

Figure 7. Mask Bits using new LFSR technique

The probability of masking a non-X response can be made arbitrarily small by increasing the number of AND gate inputs (*n*). In the rare occasion that the LFSR masks a non-X response, the LFSR can be bypassed: The correct mask bit slice can be be stored uncompressed in the ATE memory and downloaded to the chip. Figure 8 presents an overview of the compression algorithm.

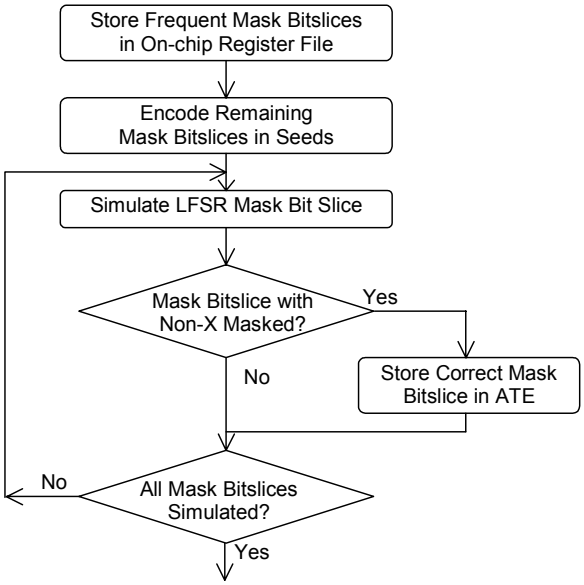


Figure 8. Compression Algorithm

We have observed that the majority of Xs occur in only a few scan chains (also see for example [4, 13, 14]). Hence, the same mask bit slice can be used very often Therefore our technique stores a few frequently used mask bit slices in a register file on-chip. The majority of Xs are compressed by merely storing a pointer to the register file instead of encoding each individual X (see Fig. 8).

The mask LFSR decompression circuitry can be combined with the stimulus LFSR decompression circuitry, see Fig. 9.

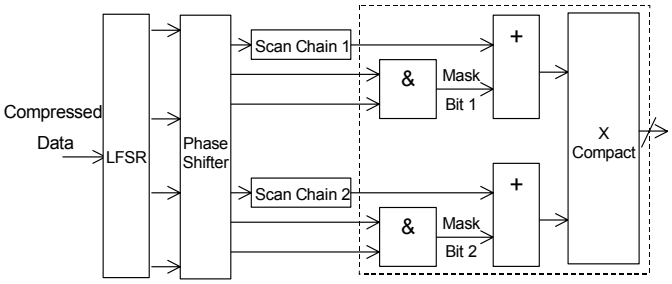


Figure 9. Combining Stimulus and Mask Decompression

The phase shifter can be designed with enough outputs to (1) shift the decompressed stimulus patterns into the scan chains and (2) supply the AND gate inputs with the mask data.

4. SIMULATION EXPERIMENTS

The response compaction techniques are applied to four industrial designs, see Table 1.

Table 1. Industrial Devices used for Experiments

	Design 1	Design 2	Design 3	Design 4
Number of Scan Chains	400	400	400	100
Total Number of Uncompressed Data Bits	1.5 Gbits	660 Mbits	150 Mbit	40 Mbits

Important metrics to evaluate response compaction techniques are:

$$\text{Compression Ratio} = \frac{\text{Number of Uncompressed Bits}}{\text{Number of Compressed Bits}}$$

$$\text{Area Overhead} = \frac{\text{Number of Gates for Mask Decompression}}{\text{Total Number of Gates in Chip}}$$

$$\text{Detection Loss} = \frac{\text{Number of Masked Non-X Test Response Bits}}{\text{Total Number of Test Response Bits}}$$

Table 2 presents the compression ratio without using an LFSR, but simply downloading the mask bit-slices from the ATE (if and only if the output bit slice requires masking).

Table 2. Without LFSR

	Compression Ratio [X]	
	MISR	X-Compact
Design 1	9.3	46.4
Design 2	56.1	58.9
Design 3	6.9	28.9
Design 4	4.7	5.3

The table shows the compression ratio of using a MISR and a X-compactor. The detection loss is 0% and the area overhead is negligible for all designs (<0.04%).

Instead of storing a mask bit slice on the ATE, one could decide to simply mask all responses (if and only if the bit slice requires masking of at least one X). However, this approach would cause a very high detection loss (up to 38% for Device 4).

Table 2 shows that Design 4 has the lowest compression ratio: Therefore, we focus on Design 4. Figure 10 presents the compression ratio as function of the number of AND gate inputs when our LFSR technique is applied to Design 4.

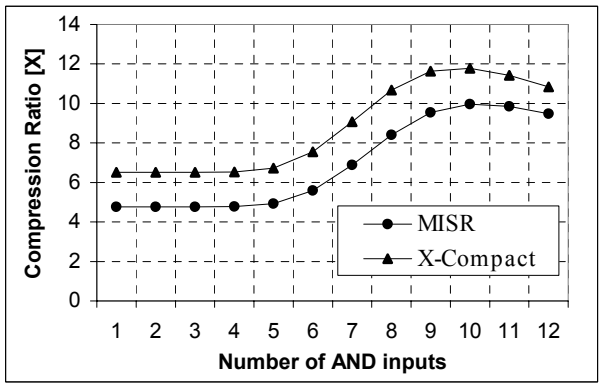


Figure 10. Our LFSR technique (Design 4)

Our LFSR technique with one AND gate input is equivalent to using the conventional LFSR technique. Fig. 10 shows that the conventional LFSR technique can only achieve a compression ratio of about 6x, whereas our new LFSR technique achieves a compression of 12x. Note that we did use the LFSR technique presented in [19] instead of [5].

The results above have a detection loss of 0% (i.e., if and only if the LFSR would mask a non-X response, then we simply download the mask bit-slice from the ATE).

If a small detection loss is acceptable, the compression ratio can be improved further. Figure 11 presents the compression ratio as function of the allowable detection loss.

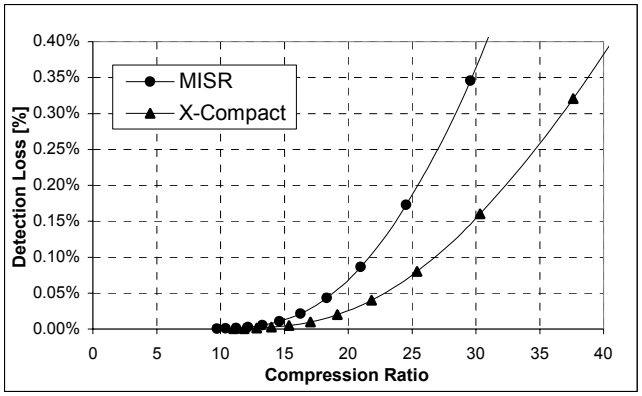


Figure 11. Detection loss vs. Compression Ratio (Design 4)

Note that a very high compression ratio can be achieved (up to 40x), with only a minor impact on detection loss (<0.38%).

As discussed in Section 3, we have observed that the majority of Xs occur in only a few scan chains. Hence, the same mask bit slice can be used very often. Therefore, we can improve the compression ratio by storing frequently used mask bit slices in a register file on-chip.

Figure 12 represents how often each mask bit slice can be used: Each dot represents a mask bit slice (Design 4). The graph overlays 8 lines, i.e., for mask bit slices with respectively 1, 2, 3, 4, 5, 6, 7, and 8 Xs.

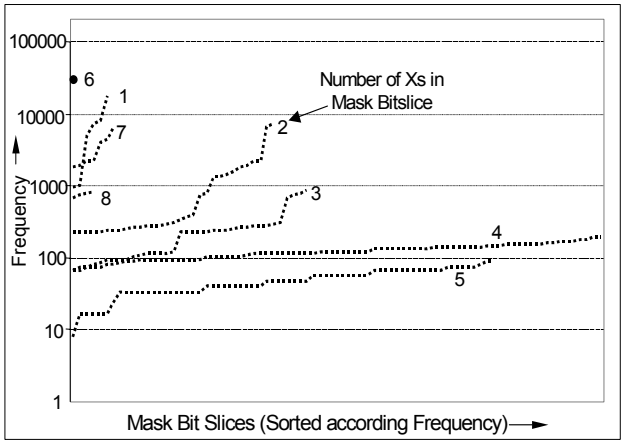


Figure 12. Mask Bit Slice Frequencies (Design 4)

Fig. 12 clearly shows that a few mask bit slices can be used for many clock cycles (e.g., the mask bit slice with 6 Xs is used 30,000 times). The many Xs can be compressed by merely storing a pointer to the register file instead of encoded each individual X.

Table 3 presents the compression ratio and area overhead using our LFSR technique as function of the number of on-chip stored mask bit slices.

Table 3. Using Register File and LFSR (Design 4)

Stored Bit slices	Compression Ratio [X]		Area Overhead
	MISR	X - Compact	
None	10.0	11.8	0.58%
3	13.3	16.4	0.62%
7	14.7	21.3	0.70%
15	20.9	25.9	0.86%

Table 3 demonstrates that storing mask bit-slices on-chip can improve the compression ratio to 25.9 (vs. 11.8 in case no bit-slices are stored on-chip).

5. CONCLUSIONS

This paper describes a new test response compaction technique with any number of unknown logic values (X's) in the test response bits by masking Xs that are not tolerated.

The concept of masking is not new. Compared to previous work, the technique (1) enables a very high compression ratio without masking non-X responses, (2) enables an even higher compression ratio by allowing some detection loss, (3) does not increase the number of test patterns, and (4) does not depending on a fault model.

We applied the technique to four industrial designs and obtained 26-fold to 60-fold reduction in test response data volume with no impact on test quality. We obtained a compression ratio of 80-fold to 120-fold if a detection loss <0.38% is allowed. Compared to using a conventional LFSR approach, our technique improves the compression ratio by >4x (26x vs. 6x). The area overhead of the technique is limited (<1%).

Our technique still depends on the ATE to supply data. Future work will be focused on reducing the ATE data requirements further.

6. ACKNOWLEDGEMENTS

This work is sponsored by Agilent Technologies.

7. REFERENCES

[1] Bardell, P.H., W. H. McAnney, J. Savir, "*Built-in Test For VLSI: Pseudo-random Techniques,*" Wiley Inter-Science, 1987.

[2] Barnhart K., B. Keller, B. Koenemann, R. Walther, "OPMISR: The Foundation for Compressed ATPG Vectors", *Proc. Int. Test. Conf.,* 2001.

[3] Benowitz, N., D. Calhoun, G. Alderson, J. Bauer, C. Joeckel, "An Advanced Fault Isolation System for Digital Logic" *IEEE Trans. Computers,* Vol.24, No.5, pp. 489-497, 1975.

[4] Chickermane, V., B. Foutz, B. Keller, "Channel Masking Synthesis for Efficient On-Chip Test Compression," *Proc. Int. Test. Conf.*, 2004.

[5] Koenemann, B., "LFSR-Coded Test Patterns for scan Designs", *Proc. of European Test Conference*, pp. 237-242, 1991.

[6] McCluskey, E.J., "Logic Design Principles with Emphasis on Testable Semi-Custom Circuits," *Prentice Hall, Englewood Cliffs, NJ, USA*, 1986.

[7] McCluskey, E.J., C.W. Tseng, "Stuck-Fault Tests vs. Actual Defects," *Proc. Intl. Test Conf.*, pp. 336-343, 2000.

[8] Mitra, S., K. S. Kim, "X-compact: An Efficient Response Compaction for Test Cost Reduction," *Proc. of the International Test Conference*, 2002.

[9] Mitra, S., S. Lumetta, M. Mitzenmacher, "X-tolerant Signature Analysis," *Proc. of the International Test Conference,* 2004.

[10] Naruse, M., I. Pomeranz, S.M. Reddy, S. Kundu, "On-chip Compression of Output Responsoes with Unknown Values using LFSR Reseeding," *Proc. Intl. Test Conf.,* 2003.

[11] Patel, J.H., S.S. Lumetta, S.M. Reddy, "Application of Saluja-Karpovsky Compactors to Test Responses with Many Unknowns," *Proc. IEEE VLSI Test Symp.,* 2003.

[12] Pomeranz, I. S. Kundu, S.M. Reddy, "On Output Response Compression in the Presence of Unknown Output Values," *Proc. Design Automation Conf.* Pp. 255-258, 2002.

[13] Rajski, J, M. Kassab, N. Mukherjee, R. Thompson, H. Tasai, A. Hertwig, N. Tamarapalli, J. Tyszer, "Embedded Deterministic Test", *Proc. International Test Conference*, 2002.

[14] Rajski, J., J. Tyszer, C. Wang, S.M. Reddy, "Convolution Compaction of Test Responses," *Proc. Intl. Test Conf.*, 200.3

[15] Saxena, N.R., and E.J. McCluskey, "Parallel Signature Analysis Design with Bounds on Aliasing" *IEEE Trans. Computers,* Vol. 46, No.5, pp. 425-438, 1997.

[16] Sinanoglue, O., A. Orailoglu, "Compacting Test Responses for Deeply Embbedded SOC Cores," *IEEE Design & Test of Computers,* Vol. 20, Issue 4, 2003.

[17] Tang, Y., H-J Wunderlich, H. Vranken, F. Hapke, M. Wittke, P. Engelke, I. Polian, B. Becker, "X-Masking During Logic BIST and Its Impact on Defect Coverage," *Proc. International Test Conference,* 2004.

[18] Tseng, C.W., and E.J. McCluskey, "Multiple-Output Propagation Transition Fault Test," *Proc. Intl. Test Conf.*, pp. 358-366, 2001.

[19] Volkerink, E.H., S. Mitra, "Efficient LFSR reseeding," *Proc. VLSI Test Symposium*, 2003.

[20] Volkerink, E.H., S. Mitra, "Test Response Compression for Any Number of Unknowns," *IEEE Intl. Workshop Infrastructure IP,* 2003.

[21] Wang, C., S.M. Reddy, I. Pomeranz, J. Rajski, J. Tyszer, "On Compacting Test Response Data Containing Unknown Values," *Proc. Intl. Conf. Computer-Aided Design,* 2003.

[22] Wohl, P., J. Waicukauski, S. Patel, M. Amin, "X-Tolerant Compression and Application of Scan ATPG patterns in a BIST architecture*", Proc. Intl. Test Conf.,* 2003.

[23] Wohl, P., L. Huisman, "Analysis and Design of Optimal Combinational Compactors," *Proc. IEEE VLSI Test Symp.,* 2003.