Applying Two-Pattern Tests Using Scan-Mapping

Nur A. Touba and Edward J. McCluskey

Center for Reliable Computing
Stanford University, Stanford, CA 94305

Abstract

This paper proposes a new technique, called scan-mapping, for applying two-pattern tests in a standard scan design environment. Scan-mapping is performed by shifting the first pattern (V_1) into the scan path and then using combinational mapping logic to generate the second pattern (V_2) in the next clock cycle. The mapping logic is placed in the scan path and avoids the performance degradation of using more complex scan elements to apply two-pattern tests. A procedure is described for synthesizing the mapping logic required to apply a set of two-pattern tests. Scan-mapping can be used in deterministic testing to apply two-pattern tests that can't be applied using scan-shifting or functional justification, and it can be used in built-in self-testing (BIST) to improve the fault coverage for delay faults. Experimental results indicate that, for deterministic testing, scan-mapping can reduce area overhead and test time compared with using complex scan elements; and for pseudo-random testing, scan-mapping can significantly improve the fault coverage using only a small amount of mapping logic.

1. Introduction

In high performance systems, many chip defects result in delay faults. Testing for delay faults requires two-pattern tests. The first pattern (V_1) initializes the circuit to a certain state, and then the second pattern (V_2) causes a signal transition that provokes the fault and propagates its effect to the outputs which are sampled at the normal operating clock speed. In a standard scan design, it is not possible to apply an arbitrary two-pattern test by *scan-shifting* (one bit shifting of the scan chain). Only a fraction, $2^{-(n-1)}$, of the possible pairs of patterns (V_1,V_2) can be applied (those in which V_2 can be generated by shifting V_1). This is a major obstacle in obtaining high coverage for delay faults. The scan chain can be re-ordered to improve the fault coverage [Mao 90], [Savir 91, 93], [Patil 92], [Cheng 93], however, this can add substantial routing overhead and doesn't always provide sufficient fault coverage.

If shifting patterns in a standard scan path doesn't provide high enough fault coverage, one solution is to use enhanced scan elements, such as those described in [Malaiya 84], [Glover 88], and [Dervisoglu 91], that have an extra holding latch so they can store 2 bits of state (i.e., both V_1 and V_2). Any possible two-pattern test can be applied using an enhanced scan path, however, this approach is rarely used because of the area and performance overhead [Varma 94].

To avoid the overhead required for using enhanced scan elements, *functional justification* techniques have been proposed in [Cheng 93], [Underwood 94], and [Varma 94]. The first pattern (V_1) is shifted into the scan path, and then the second pattern (V_2) is generated through the functional logic and loaded into the scan path with the system clock. The limitations of functional justification are that two-cycle based test generation is more time consuming and in many cases it is not possible to generate a test for a fault that could be tested using enhanced scan elements. If functional justification cannot provide sufficient fault coverage, then a partial enhanced scan approach can be used in which some of the scan elements are enhanced. Cheng *et al.* [Cheng 93] describe an algorithm that attempts to minimize the number of scan elements that have to be enhanced.

This paper proposes a new technique for generating two-pattern tests in a standard scan path called *scan-mapping*. As illustrated in Fig. 1, it involves adding combinational mapping logic in the scan path. When the mapping logic is enabled, it maps the next pattern generated in the scan chain into a new pattern. In the example in Fig. 2, if the pattern *1000* is in the scan chain, and a *1* is shifted in, then the next pattern normally would be *1100*, however, if the mapping logic is enabled then it maps the next pattern into *1101*. The mapping logic can be designed so that it can be used to apply the required two-pattern tests that can't be applied by scan-shifting or functional justification. Scan-mapping is performed by shifting the first pattern (V_1) into the scan path and then using the mapping logic to

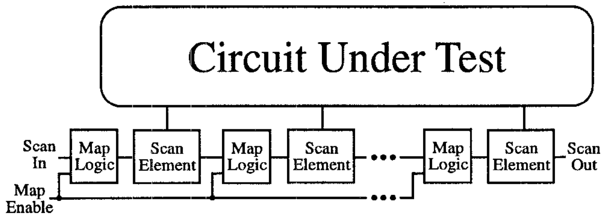

Figure 1 . Block Diagram for Scan-Mapping

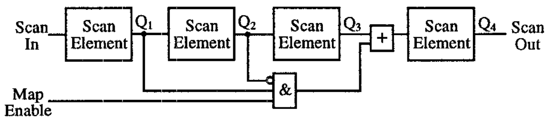

Figure 2 . Example of Scan-Mapping

generate the second pattern (V_2). Given a set of two-pattern tests that are to be applied using scan-mapping, a procedure is described in Sec. 2 for synthesizing the required mapping logic in a way that minimizes area overhead.

Scan-mapping has application in both deterministic testing and built-in self-testing (BIST). In deterministic testing, scan-mapping can be used to apply two-pattern tests that can't be applied using scan-shifting or functional justification. In BIST, scan-mapping can be used to improve the fault coverage for delay faults. A delay fault may not be detected during BIST because the test pattern generator is either unable to generate a two-pattern test for the fault [Furuya 91], or requires a very long test sequence to generate a two-pattern test for the fault [Savir 86]. Scan-mapping can be used to map selected patterns in a given pseudo-random sequence into new patterns to produce two-pattern tests that detect faults that would have otherwise been missed, thereby improving the fault coverage. A procedure is described in Sec. 4 for designing mapping logic to achieve a desired fault coverage for a given pseudo-random sequence.

The paper is organized as follows: In Sec. 2, a procedure is described for designing mapping logic to apply a given set of two-pattern tests using scan-mapping. In Sec. 3, the application of scan-mapping to deterministic testing is discussed and experimental results are given comparing scan-mapping with using enhanced scan elements. In Sec. 4, the application of scan-mapping to BIST is discussed and experimental results are shown. Sec. 5 is a conclusion.

2. Synthesizing Mapping Logic for Scan-Mapping

Scan-mapping is performed by placing mapping logic in the scan path. Given a set of two-pattern tests that are to be applied using scan-mapping, this section describes a procedure for synthesizing mapping logic to perform the desired scan-mapping. The goal of the synthesis procedure is to minimize area overhead.

2.1 Specifying a Mapping Function

Using scan-mapping to apply a two-pattern test involves shifting the first pattern, V_1, into the scan chain and then mapping the next state of the scan chain, denoted as *NextState(V_1)*, into the second pattern, V_2. Given a set of two-pattern tests that are to be applied by scan-mapping, a mapping function is specified by having the next state for each V_1 pattern, *NextState(V_1)*, be mapped into the corresponding V_2 pattern. The V_1 patterns must be distinct, however the V_2 patterns may contain unspecified inputs that are left as don't cares (X's). Fig. 3 shows an example of a mapping function for applying the following set of two-pattern tests: *{ (1110, 0011), (0100, X000), (1010, 1XX0), (1111, 110X) }*. The next states for the V_1 patterns are *1111*, *0010*, *0101*, and *0111* respectively, so these patterns are mapped into the V_2 patterns *0011*, *X000*, *1XX0*, and *110X* respectively.

NS(V_1)		V_2	B-Matrix							
$x_1x_2x_3x_4$		$x_1x_2x_3x_4$	x_1'	x_2'	x_3'	x_4'	x_1	x_2	x_3	x_4
1 1 1 1	→	0 0 1 1	1*	1*	0	0	0	0	1	1
0 0 1 0	→	X 0 0 0	1	1	1*	1	1	0	0	0
0 1 0 1	→	1 X X 0	0	1	1	1*	1*	1	1	0
0 1 1 1	→	1 1 0 X	0	0	1*	1	1*	1	0	1

Figure 3. Example of Rectangles in a B-Matrix

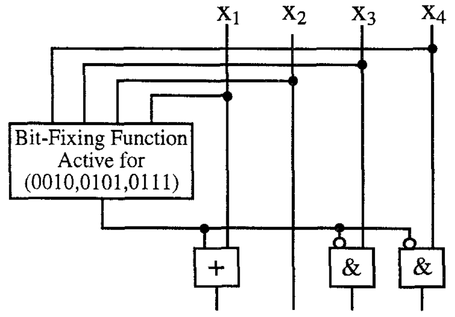

Figure 4. Bit-Fixing Logic Corresponding to the Larger of the Two Rectangles in the B-Matrix in Fig. 3

For most faults, there are a number of possible initialization (V_1) patterns, so there is some degree of freedom in selecting the V_1 patterns that are matched with the V_2 patterns for each two-pattern test. Thus, there are many possible mapping functions depending on which V_1 patterns are selected. The amount of logic required to implement each possible mapping function varies greatly. So the problem of minimizing the mapping logic involves careful selection of the mapping function.

2.2 Minimizing the Mapping Logic

In order to map pattern X into pattern Y, each bit in pattern X that differs from the corresponding bit in pattern Y must be "fixed" so that it matches. For example, if the pattern *0010* is being mapped into the pattern *1000*, then the first bit must be fixed to a *'1'*, and the third bit must be fixed to a *'0'*. An example of "bit-fixing" logic is shown in Fig. 4. There is some "bit-fixing function" which is active for some set of patterns (in this case, *0010*, *0101*, and *0111*). When the bit-fixing function is active, it fixes some of the outputs to a specific logic value (in this case, x_1 = *'1'*, x_3 = *'0'*, x_4 = *'0'*). So in this example, the bit-fixing logic maps the original patterns *0010*, *0101*, and *0111* into the patterns *1000*, *1100*, and *1100*, respectively. The strategy for minimizing the mapping logic is to select the mapping function in a way that minimizes the amount of bit-fixing that is required.

The problem of finding the minimum number of bit-fixing functions required to implement a mapping function can be formulated as one of finding the minimum rectangle cover in a binate matrix as described in [Touba 95]. A binate matrix B, where $B_{ij} \in \{0,1\}$, is formed in which each V_2 pattern is represented by a row.

There is a complemented and uncomplemented column corresponding to each input in the V_2 pattern. If an input in a V_2 pattern is a *'0'*, then its corresponding complemented and uncomplemented column entries are set equal to 1 and 0, respectively. If an input in a V_2 pattern is a *'1'*, then its corresponding complemented and uncomplemented column entries are set equal to 0 and 1, respectively. If an input in a V_2 pattern is a don't care (*'X'*), then its corresponding complemented and uncomplemented column entries are both set equal to 1. A 1-entry in *B* is starred if it corresponds to a bit difference between the *NextState*(V_1) pattern and the V_2 pattern. A *rectangle* in *B* is a subset of rows *R* and a subset of columns *C* such that $B_{ij} = 1$ for all $i \in R$ and $j \in C$. A rectangle in *B* corresponds to a common bit-fixing function between the outputs of the mapping logic. In the example in Fig. 3, there are two rectangles that are shown; the larger rectangle corresponds to a bit-fixing function that is active if any of the *NextState*(V_2) patterns that correspond to the rows in the rectangle, i.e., *0010*, *0101*, and *0111*, are applied to the mapping logic. When the bit-fixing function is active, it fixes the outputs that correspond to the columns in the rectangle to a specific logic value, i.e., x_1 = *'1'*, x_3 = *'0'*, x_4 = *'0'*. Thus, the larger of the two rectangles shown in Fig. 3 corresponds to a transformation in which the *NextState*(V_1) patterns *0010*, *1101*, and *0111* are mapped into the patterns *1000*, *1100*, and *1100*, respectively, thereby generating the required two-pattern tests. Each rectangle in *B* corresponds to a bit-fixing function that forces the logic value at a set of outputs. In order for the mapping logic to transform all of the *NextState*(V_1) patterns so that they match their respective V_2 patterns, each bit difference (which corresponds to a starred entry in *B*) must be contained in a rectangle that is implemented by the mapping logic. The mapping logic is designed by finding a set of rectangles that covers all of the starred entries in *B* and then constructing logic to implement the transformations corresponding to those rectangles. Minimizing the mapping logic corresponds to finding a minimum set of rectangles that covers all of the starred entries in *B*.

The set of stared entries in *B* depends on which V_1 pattern is used for each V_2 pattern. Therefore, the problem of selecting a mapping function to minimize the amount of mapping logic corresponds to selecting the V_1 patterns for the V_2 patterns in each two-pattern test in a way that minimizes the number of rectangles required to cover all the resulting stared entries. A heuristic procedure for solving this problem is described in detail in [Touba 95]. In [Touba 95], the procedure is described for matching original patterns with test cubes, however, the same procedure can be used to match *NextState*(V_1) patterns with V_2 patterns to select the mapping function that requires the least amount of logic to implement.

2.3 Implementing the Mapping Logic

After the mapping function has been selected and the set of rectangles that covers all the stared entries in the *B*-matrix has been minimized, the mapping logic can be synthesized.

NextState(V_1) Patterns: 1111, 0010, 0101, 0111

Bit-Fixing Function 1:
- On-Set = 1111
- Off-Set = 0010, 0101, 0111
- Synthesized Logic $<Q_1,Q_2,Q_3,Q_4>$ = $Q_1\ Q_3$

Bit-Fixing Function 2:
- On-Set = 0010, 0101, 0111
- Off-Set = 1111
- Synthesized Logic $<Q_1,Q_2,Q_3,Q_4>$ = $Q_1' + Q_3'$

Figure 5 . Bit-Fixing Functions Corresponding to Set of Rectangles in Fig. 3

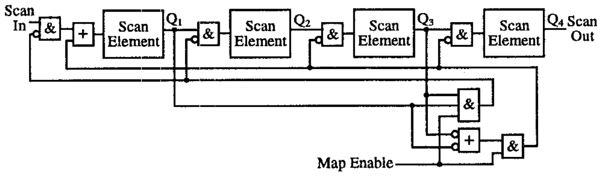

Figure 6 . Hardware Implementation of Mapping Logic for Set of Rectangles in Fig. 3.

This is best explained with an example. In Fig. 5, the bit-fixing functions corresponding to the two rectangles in Fig. 3 are specified. For each rectangle, the *NextState*(V_1) patterns corresponding to the rows in the rectangle are placed in the on-set since the bix-fixing function must be active for those patterns. The other *NextState*(V_1) patterns should not be transformed, so the bit-fixing function shouldn't be active for those patterns, therefore they are added to the off-set. The on-set and off-set specify the bit-fixing function and can be passed to a logic synthesis tool to generate a logic implementation.

Fig. 6 shows a hardware implementation for the mapping function specified in Fig. 3. The bit-fixing functions were derived as shown in Fig. 5. Each bit-fixing function forces the logic value at the outputs corresponding to the columns in its rectangle. When bit-fixing function 1, $Q_1\ Q_3$, is active, it forces D_1 = *'0'* and D_2 = *'0'*. This is implemented by adding AND gates at the inputs of D_1 and D_2. When bit-fixing function 2, $Q_1' + Q_3'$, is active, it forces D_1 = *'1'*, D_3 = *'0'* and D_4 = *'0'*. This is implemented by adding an OR gate at the input of D_1, and AND gates at the inputs of D_3 and D_4. A "map enable" signal is ANDed in at the output of each bit-fixing function to control when scan-mapping is performed. When the "map enable" signal is not active, then normal scan-shifting is performed.

3. Application of Scan-Mapping to Deterministic Testing

In deterministic testing, scan-shifting and functional justification can be used to generate two-pattern tests for as many faults as possible. If the fault coverage is not sufficient, then there are two options. One is to use enhanced scan elements, and the other is to use the proposed scan-mapping technique. To use scan-mapping, a set of two-pattern tests that will detect the remaining undetected faults is determined, and then the mapping logic required to apply those tests is synthesized using the procedure described in Sec. 2. Scan-mapping is performed by shifting in the first pattern (V_1), and then enabling the mapping logic on the next cycle to produce the second pattern (V_2). Scan-mapping doesn't add any extra stages to the scan chain, and therefore requires less test time and test storage compared with using enhanced scan elements.

Table 1 shows some experimental results for using scan-mapping to apply two-pattern tests for CMOS stuck-open faults in some of the ISCAS 89 [Brglez 89] benchmark circuits. Results are shown for this particular type of delay fault because of the availability of the SOPRANO program [Lee 90] for fault simulation and test pattern generation. Note, however, that scan-mapping can be used for applying any two-pattern test set regardless of what fault model or software is used to generate it. The flip-flops and primary inputs in each circuit were configured in a scan path using the default ordering. The first two columns give the circuit name and the number of elements in the scan chain. Then results are shown for applying two-pattern tests using a standard scan path. Fault simulation was performed using all of the 2^{n+1} possible two-pattern tests that can be applied using a standard scan path. The fault coverage and number of undetected faults are shown. Next, results are shown for using an enhanced scan path. Since all possible two-pattern tests can be applied, the fault coverage is 100%. The overhead is an extra latch for each scan element. In order to compare the overhead with scan-mapping, the number of gate equivalents (GE's) are shown. The gate equivalents were computed by counting each n-input NAND or NOR as $(0.5)(n)$ GE's and each latch as 2.5 GE's to reflect a static CMOS technology. In the last columns, results are shown for using scan-mapping. The mapping logic was designed to provide 100% fault coverage, and the number of gate equivalents required to implement the mapping logic is shown. The results indicate that a small amount of mapping logic can be used in a standard scan design to significantly increase the fault coverage for faults requiring two-pattern tests.

Table 1. Experimental Results for Using Scan-Mapping for Deterministic Testing of CMOS Stuck-Open Faults

Circuit	Scan	Std Scan		Enhanced Scan		Scan-Mapping	
Name	Elem	Cov	Und	Cov	Ovrhd	Cov	Ovrhd
s208	18	97.4%	10	100%	45 GE	100%	22 GE
s298	17	99.3%	3	100%	42 GE	100%	3 GE
s349	24	99.3%	3	100%	60 GE	100%	3 GE
s386	13	96.4%	22	100%	32 GE	100%	9 GE
s444	24	99.0%	4	100%	60 GE	100%	4 GE
s820	23	97.7%	35	100%	57 GE	100%	6 GE
s832	23	98.1%	29	100%	57 GE	100%	4 GE
s1488	17	98.5%	36	100%	42 GE	100%	32 GE

4. Application of Scan-Mapping to BIST

One approach for BIST is to use a pseudo-random pattern generator, e.g., an LFSR, to shift a pseudo-random sequence into the scan chain. Faults requiring two-pattern tests may go undetected for two reasons: (1) it impossible to generate a two-pattern test for the fault due to the order of the scan path, or (2) the probability of generating a two-pattern test for the fault is so low that an unreasonably long test length would be required to detect the fault. Using enhanced scan elements can solve the first problem by making it possible to generate all possible two-pattern tests, however, it doesn't help with the second problem of low fault detection probabilities. The proposed scan-mapping technique, however, can solve both problems. If an initialization pattern (V_1) for some fault occurs in the pseudo-random sequence, then the fault can be detected by using scan-mapping to generate the second pattern (V_2) of the two-pattern test.

Fault simulation can be done to determine which faults are detected by the pseudo-random sequence that is generated during BIST. For each fault that is not detected, an initialization pattern (V_1) for the fault can be identified (if one exists) in the pseudo-random sequence. The mapping logic can then be synthesized using the procedure described in Sec. 2 so that scan-mapping can be used to produce the corresponding V_2 patterns to generate a two-pattern test for each undetected fault. This method is capable of detecting all faults provided that an initialization pattern (V_1) for the fault is generated in the pseudo-random sequence.

Table 2 shows some experimental results for using scan-mapping to increase the fault coverage for pseudo-random pattern testing of CMOS stuck-open faults in some of the ISCAS 89 benchmark circuits. The flip-flops and primary inputs in each circuit were configured into a scan chain, and an LFSR was used to shift a pseudo-random sequence of 100,000 bits into the scan chain. The first two columns give the circuit name and the number of scan elements in the scan chain. Then the fault coverage obtained using a standard scan path is shown. Next the fault coverage using an enhanced scan path is shown. The overhead is an extra latch for each scan element which is shown in terms of gate equivalents. The last two columns show results for using scan-mapping. The mapping logic was designed to generate V_2 patterns for all undetected faults for which a V_1 pattern appears in the pseudo-random pattern sequence. The number of gate equivalents required to implement the mapping logic is shown.

The results indicate that a small amount of mapping logic is capable of significantly increasing the fault coverage during BIST. Using an enhanced scan path improves the fault coverage compared with using a standard scan path because it eliminates the shift correlation between V_1 and V_2, however, it doesn't help for faults with low detection probabilities. Scan-mapping improves the fault coverage more than using an enhanced scan path because it only requires that a V_1 pattern be generated for the fault thereby improving the detection probability. Scan-mapping was able to provide 100% coverage in all of the circuits except for *s420* and *s641*. In those two circuits, there were some faults for which no V_1 pattern was generated in the pseudo-random sequence. Techniques such as those in [Konemann 91], [Dufaza 91], [Hellebrand 95], and [Zacharia 95], can be used to reseed the LFSR to ensure that a V_1 pattern for each fault is generated.

Table 2. Experimental Results for Using Scan-Mapping for Pseudo-Random Testing of CMOS Stuck-Open Faults

Circuit Name	Scan Elem	Std Scan	Enhanced Scan		Scan-Mapping	
		Cov	Cov	Ovrhd	Cov	Ovrhd
s420	35	83.8%	88.9%	87 GE	90.0%	129 GE
s510	25	99.2%	100%	62 GE	100%	11 GE
s526	24	96.2%	99.7%	60 GE	100%	11 GE
s641	54	95.1%	96.5%	135 GE	98.0%	62 GE
s820	23	93.7%	99.6%	57 GE	100%	83 GE
s832	23	92.9%	98.7%	57 GE	100%	91 GE
s953	45	90.6%	96.2%	112 GE	100%	81 GE
s1423	91	97.9%	98.4%	227 GE	100%	74 GE
s1488	17	97.9%	99.2%	42 GE	100%	14 GE

5. Conclusions

Scan-mapping can be used to obtain high coverage of delay faults for either deterministic testing or BIST. For deterministic testing, scan-mapping enables 100% coverage of detectable delay faults while providing the following advantages compared to using enhanced scan elements: less performance degradation, less area overhead, faster test time, and less test storage requirements. For BIST, scan-mapping improves the fault coverage of not only the delay faults that can't be detected due to shift correlation, but also the "random-pattern-resistant" delay faults.

Acknowledgments

The authors would like to thank Piero Franco for his helpful comments. This work was supported in part by the Ballistic Missile Defense Organization, Innovative Science and Technology (BMDO/IST) Directorate and administered through the Department of the Navy, Office of Naval Research under Grant No. N00014-92-J-1782, by the National Science Foundation under Grant No. MIP-9107760, and by the Advanced Research Projects Agency under prime contract No. DABT63-94-C-0045.

References

[Brglez 85] Brglez, F., and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortan," *Proc. of International Symposium on Circuits and Systems*, pp. 663-698, 1985.

[Brglez 89] Brglez, F., D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. of Int. Symp. on Cir. and Sys.*, pp. 1929-1934, 1989.

[Cheng 93] Cheng, K.-T., S. Devadas, and K. Keutzer, "Delay-Fault Test Generation and Synthesis for Testability Under a Standard Scan Design Methodology," *IEEE Trrans. on CAD*, Vol. 12, No. 8, pp. 1217-1231, Aug. 1993.

[Dervisoglu 91] Dervisoglu, B.I., and G.E. Strong, "Design for Testability: Using ScanPath Techniques for Path-Delay Test and Measurement," *Proc. of International Test Conference*, pp. 365-374, 1991.

[Dufaza 91] Dufaza, C., and G. Cambon, "LFSR based Deterministic and Pseudo-Random Test Pattern Generator Structures," *Proc. of European Test Conf.*, pp. 27-34, 1991.

[Furuya 91] Furuya, K., and E.J. McCluskey, "Two-Pattern Test Capabilities of Autonomous TPG Circuits," *Proc. of International Test Conference*, pp. 704-711, 1991.

[Glover 88] Glover, C.T., and M.R. Mercer, "A Method of Delay Fault Test Generation," *Proc. of the 25th Design Automation Conference*, pp. 90-95, 1988.

[Hellebrand 95] Hellebrand, S., J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Transactions on Computers*, Vol. 44, No. 2, pp. 223-233, Feb. 1995.

[Konemann 91] Konemann, B., "LFSR-Coded Test Patterns for Scan Designs," *Proc. of Eurp. Test Conf.*, pp.237-242, 1991.

[Lee 90] Lee, H.K., and D.S. Ha, "SOPRANO: An Efficient Automatic Test Pattern Generator for Stuck-Open Faults in CMOS Combinational Circuits," *Proc. of the 27th Design Automation Conference*, pp. 90-95, 1990.

[Mao 90] Mao, W., and M. Ciletti, "Arrangement of Latches in Scan-Path Design to Improve Delay Fault Coverage," *Proc. of International Test Conference*, pp. 387-393, 1990.

[Malaiya 84] Malaiya, Y.K., and R. Narayanaswamy, "Modeling and Testing for Timing Faults in Synchronous Sequential Circuits," *IEEE Design and Test*, pp. 62-74, Nov. 1984.

[Patil 92] Patil, S., and J. Savir, "Skewed-Load Transition Test: Part II, Coverage," *Proc. of International Test Conference*, pp. 714-722, 1992.

[Savir 86] Savir, J., and W.H. McAnney, "Random Pattern Testability of Delay Faults," *Proc. of International Test Conference*, pp. 263-273, 1986.

[Savir 91] Savir, J., and R. Berry, "At-Speed Test is Not Necessarily an AC Test," *Proc. of International Test Conference*, pp. 722-728, 1991.

[Savir 93] Savir, J., and S. Patil, "Scan-Based Transition Test," *IEEE Transactions on Computer-Aided Design*, Vol. 12, No. 8, pp. 1232-1241, Aug. 1993.

[Touba 95] Touba, N.A., and E.J. McCluskey, "Synthesis of Mapping Logic for Generating Transformed Pseudo-Random Patterns for BIST," *Proc. of International Test Conference*, pp. 674-682, 1995.

[Underwood 94] Underwood, B., W.-O. Law, S. Kang, and H. Konuk, "Fastpath: A Path-Delay Test Generator for Standard Scan Designs," *Proc. of Int. Test Conf.*, pp. 154-163, 1994.

[Varma 94] Varma, P., "On Path Delay Testing in a Standard Scan Environment," *Proc. of International Test Conference*, pp. 164-173, 1994.

[Zacharia 95] Zacharia, N., J. Rajski, and J. Tyszer, "Decompression of Test Data Using Variable-Length Seed LFSRs," *Proc. of VLSI Test Symp.*, pp. 426-433, 1995.