Center for Reliable Computing

TECHNICAL REPORT

# Synthesis Techniques for Pseudo-Random Built-In Self-Test

Nur A. Touba

**96-4**

(CSL TN # 96-x)

August 1996

**Center for Reliable Computing**
ERL 460
Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-4055

**Abstract:**

This technical report contains the text of Nur Touba's thesis "Synthesis Techniques for Pseudo-Random Built-In Self-Test." The thesis appendices have appeared as CRC Technical Reports, and are not included here.

**Funding:**

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.

# SYNTHESIS TECHNIQUES
# FOR
# PSEUDO-RANDOM BUILT-IN SELF-TEST

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

**Nur A. Touba**

June 1996

# ABSTRACT

Built-in self-test (BIST) techniques enable an integrated circuit (IC) to test itself. BIST reduces test and maintenance costs for an IC by eliminating the need for expensive test equipment and by allowing fast location of failed ICs in a system. BIST also allows an IC to be tested at its normal operating speed which is very important for detecting timing faults. Despite all of these advantages, BIST has seen limited use in industry because of area and performance overhead and increased design time. This dissertation presents automated techniques for implementing BIST in a way that minimizes area and performance overhead.

A low-overhead approach for BIST is to use a linear feedback shift register (LFSR) to apply pseudo-random test patterns to the circuit-under-test. Unfortunately, many circuits contain random-pattern-resistant faults which limit the fault coverage that can be obtained for pseudo-random BIST. Several different approaches for solving this problem are presented.

A logic synthesis procedure that performs testability-driven factoring to generate a random pattern testable design is presented. By considering random pattern testability during the factoring process, the overhead can be minimized.

For hand-designed circuits or circuits that are not synthesizable, an innovative test point insertion procedure is described for inserting test points to make the circuit random pattern testable. A path tracing procedure is used for test point placement. A few of the existing primary inputs are ANDed together to form signals that drive the control points. These innovations result in fewer test points than previous methods.

If it is not possible or not desirable to modify the circuit-under-test, then a procedure is described for synthesizing mapping logic that can placed at the output of the LFSR to transform the pseudo-random patterns so that they provide the required fault coverage. Much less overhead is required compared with weighted pattern testing methods.

Lastly, a technique is described for placing bit-fixing logic at the serial output of an LFSR to embed deterministic test patterns for the random pattern resistant faults in the pseudo-random bit sequence. This method does not require any performance overhead beyond what is needed for scan.

# ACKNOWLEDGMENTS

I express my deep gratefulness to my adviser, Prof. Edward J. McCluskey, for his guidance and support during my time at Stanford. He modeled the high quality teaching and research that I aspire to emulate in my career. He taught me much about finding good research problems and clearly presenting results. Many things that I learned from him will be of great help to me during my career.

I would like to thank Prof. Giovanni De Micheli, my associate advisor, Prof. Robert Gray, my committee chairman, and Prof. Oyekunle Olukotun for being the final member of my committee. Special thanks to Prof. Joseph Goodman for being my third reader.

I have greatly appreciated my colleagues at the Center for Reliable Computing: Khader "KD" Abdel-Hafez, Dave Brokaw, Yi-Chin Chu, Dr. Hong Hao, Erin Kan, Sunil Koslage, Wern-Yan Koe, Vincent Lo, Samy Makar, Shridhar Mukund, Rong Pan, Dr. Alice Tokarnia, and Sanjay Wattal. I want to especially thank Dr. LaNae Avra for helping me get my start at CRC, Dr. Piero Franco for answering my many questions, Dr. Siyad Ma for sharing many trials and joys, Dr. Nirmal Saxena for his encouragement and advice, and Rob Norwood, Jonathan Chang, and Philip Shrivani for being so fun to work with (and to beat in basketball).

I want to especially thank Siegrid Munda for her administrative support. I very greatly appreciated her kindness and helpfulness. Special thanks also to Sherry Turner for her assistance.

I would like to thank the CRC visitors who have helped me: Francoise Martinolle for reading my early papers, Prof. Irith Pomeranz for her advice and suggestions, and Prof. Hans-Joachim Wunderlich and Prof. Sybille Hellebrand for our many technical discussions.

I am grateful to Prof. Larry Kinney and Prof. William Plice at the University of Minnesota for getting me interested in IC testing in the first place.

I want to thank my many friends in the IVCF Grad group for their prayers and support. I would like to mention just a few by name: Jennifer Amyx, Beth Bryson, Dan Clendenin, Loren Eyres, Scott Hunicke-Smith, Mike Kaliski, Alfred Kwok, Vince Mooney, Elaine Naugle, Jeff Rembold, Robin Seydel, Jim Strzelec, Mary K. Wilson, and Conrad Yoder. I want to especially thank Kim Norman for all of her encouragement and prayers. Her coffee maker helped me make it through many all-nighters needed to meet conference submission deadlines.

Finally, I would like to thank my parents for their tremendous love, endless support, and many prayers. They have always believed in me and always been there for me. I dedicate this dissertation to them.

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.

# TABLE OF CONTENTS

# LIST OF TABLES

**Table** **Title**

# LIST OF ILLUSTRATIONS

# Chapter 1
# Introduction

## 1.1 Background

In the production of integrated circuits, testing is done to identify defective chips. This is very important for shipping high quality products. Testing is also done to diagnose the reason for a chip failure in order to improve the manufacturing process. In system maintenance, testing is done to identify parts that need to be replaced in order to repair a system.

Testing a digital circuit involves applying an appropriate set of input patterns to the circuit and checking for the correct outputs. The conventional approach is to use an external tester to perform the test. However, built-in self-test (BIST) techniques have been developed in which some of the tester functions are incorporated on the chip enabling the chip to test itself. BIST provides a number of well-known advantages. It eliminates the need for expensive testers. It provides fast location of failed units in a system because the chips can test themselves concurrently. And, it allows *at-speed testing* in which the chip is tested at its normal operating clock rate which is very important for detecting timing faults. Despite all of these advantages, BIST has seen limited use in industry because of its area and performance overhead, increased design time, and lack of BIST design tools. These are problems that this dissertation addresses.

The research described in this dissertation is timely because the interest in BIST is growing rapidly. The increasing pin count, operating speed, and complexity of IC's is outstripping the capabilities of external testers. BIST provides solutions to these problems.

## 1.2 Pseudo-Random BIST

Figure 1.1 is a block diagram showing the architecture for BIST. The circuit that is being tested is called the *circuit-under-test (CUT).* There is a *test pattern generator* which applies test patterns to the CUT and an *output response analyzer* which checks the outputs. The test pattern generator must generate a set of test patterns that provides a high fault coverage in order to thoroughly test the CUT.

Pseudo-random testing is an attractive approach for BIST. A linear feedback shift register (LFSR) can be used to apply pseudo-random patterns to the CUT. An LFSR has a simple structure requiring small area overhead. Moreover, an LFSR can also be used as an output response analyzer thereby serving a dual purpose. BIST techniques such as circular BIST [Stroud 88], [Krasniewski 89], and BILBO registers [Koenemann 79] make use of this advantage to reduce overhead.

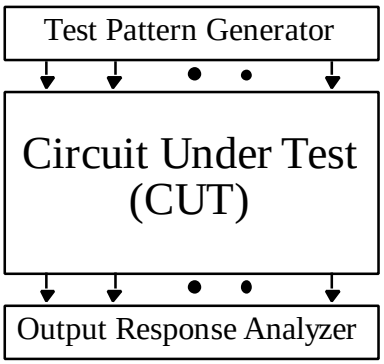



Figure 1.1. Block Diagram for BIST

There are limits on the *test length*, which is the number of pseudo-random patterns that can be applied during BIST. One limit is simply the amount of time that is required to apply the patterns. Another limit is the fault simulation time required to determine the fault coverage. A third limit is heat dissipation for an unpackaged die. Thus, in order for pseudo-random pattern testing to be effective, a high fault coverage must be obtained for an "acceptable" test length. What is considered acceptable depends on the particular test environment.

The probability of detecting a fault with a single random pattern is defined as the *detection probability* for the fault and is given by the number of patterns that detect the fault divided by the total number of inputs patterns, $2^n$, where $n$ is the number of inputs in the circuit. Unfortunately, many circuits contain faults with very low detection probabilities. Such faults are said to be *random-pattern-resistant (r.p.r.)* [Eichelberger 83] because they are hard to detect with random patterns and therefore limit the fault coverage for pseudo-random testing. A circuit is said to be *random pattern testable* if it does not contain any r.p.r. faults.

If the fault coverage for pseudo-random BIST is insufficient, then there are two solutions. One is to modify the circuit-under-test to make it random pattern testable, and the other is to modify the test pattern generator so that it generates patterns that detect the r.p.r. faults. Innovative techniques for both of these approaches are described in this dissertation. These techniques enable automated design of pseudo-random BIST implementations that satisfy fault coverage requirements while minimizing area and performance overhead. These techniques have been incorporated in the TOPS (Totally Optimized Synthesis-for-test) tool being developed at the Center for Reliable Computing.

## 1.3 Outline

This dissertation summarizes my work in pseudo-random BIST. Detailed descriptions of results are found in the appendices which are reprints of published or submitted papers.

Chapter 2 describes techniques for modifying a circuit to make it random pattern testable. A survey of previous work is presented followed by a summary of the new techniques.

An innovative test point insertion technique is described which uses a path tracing procedure to place both control and observation points. Rather than using extra scan elements to drive the control points, a few of the existing primary inputs are ANDed together to form signals that drive the control points. This test point insertion procedure can be used to target both stuck-at and bridging faults.

Given a logic function, a logic synthesis procedure is described for generating a random pattern testable implementation. By considering testability during the factor section process, the procedure performs testability-driven factoring to generate a random pattern testable implementation.

Chapter 3 describes techniques for modifying the test pattern generator so that it generates patterns that detect the r.p.r. faults. A survey of the previous work for both weighted pattern testing and mixed-mode testing is presented followed by a summary of the new techniques.

A procedure is described for synthesizing mapping logic that can be placed at the output of the LFSR to transform the pseudo-random patterns that are generated so that they provide the required fault coverage. By considering a broader class of mapping functions, not just those that implement weight sets, the overhead is significantly minimized compared with weighted pattern testing methods.

A new approach for mixed-mode scan BIST is described. Logic at the serial output of the LFSR to "fix" certain bits in the sequence in order to embed deterministic test patterns that detect the r.p.r. faults.

Chapter 4 concludes the dissertation.

# Chapter 2
# Random Pattern Testable Design

If pseudo-random BIST does not provide sufficiently high fault coverage for a circuit, then one solution is to modify the circuit to make it random pattern testable. This chapter begins with a survey of the previous work that has been done in this area and then summarizes the new techniques presented in Appendices I, IV, and V.

## 2.1 Previous Work in Random Pattern Testable Design

Previous work in random pattern testable design focused on inserting test points into a circuit to make it random pattern testable. Test point insertion involves adding control and observation points to the circuit in a way that the system function remains the same, but the testability is improved [Hayes 74]. An *observation point* is an additional primary output that is inserted in the circuit to increase the observability of faults in the circuit. In the example in Fig. 2.1, an observation point is inserted at the output of gate *G1* such that faults are observable regardless of the logic value at node *y*. A *control point* is inserted in the circuit such that when it is activated, it fixes the logic value at a particular node to increase the controllability of some faults in the circuit. A control point can also affect the observability of some faults in the circuit because it can change the propagation paths in the circuit. In the example in Fig. 2.2, a control point is inserted to fix the logic value at the output of gate *G1* to a '1' when the control point is activated (this is called a *control-1 point*). This is accomplished by placing an OR gate at the output of gate *G1*. In the example in Fig. 2.3, a control point is inserted to fix the logic value at the output of gate *G1* to a '0' when the control point is activated (this is called a *control-0 point*). This is accomplished by placing an AND gate at the output of gate *G1*. During system operation, the control points are not activated and thus don't affect the system function. However, control points do add an extra level of logic to some paths in the circuit. If a control point is placed on a critical timing path, it can increase the cycle time of the circuit.

Since test points add both area and performance overhead, it is important to try to minimize the number of test points that are inserted to achieve the desired fault coverage. Optimal test point placement for circuits with reconvergent fan-out has been shown to be NP-complete [Krishnamurthy 87]. An ad-hoc approach for placing test points was presented in [Eichelberger 83]. Briers and Totton [Briers 86] were the first to propose a systematic method for test point placement to increase pseudo-random pattern testability. They use simulation statistics to identify correlations between signals, and then insert test points to break the correlation. The number of test points inserted by this method is large. Iyengar and Brand [Iyengar 89] proposed an improved method that uses fault simulation to identify gates that block fault propagation, and then inserts test points to enable propagation. Savaria *et al.*, in [Savaria 91] and

[Youssef 93], use the COP testability measures [Brglez 84] to guide the placement of test points. They identify sectors of hard-to-detect faults and insert test points at the origins of the sectors. Seiss *et al.*, in [Seiss 91], form a cost function based on the COP testability measures and then compute, in linear time, the gradient of the function with respect to each possible test point. The gradients are used to approximate the global testability impact for inserting a particular test point. Based on these approximations, a test point is inserted and the COP testability measures are recomputed. This process iterates until the testability is satisfactory. Cheng and Lin, in [Cheng 95], enhance the procedure in [Seiss 91] to consider the performance impact of inserting a particular test point. They showed that by avoiding control point insertion on critical timing paths, high fault coverage can be achieved with zero performance degradation.

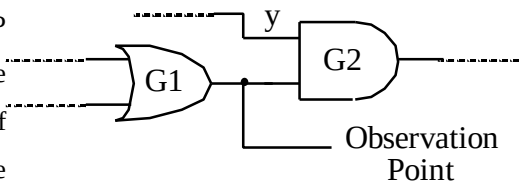



Figure 2.1. Example of Observation Point

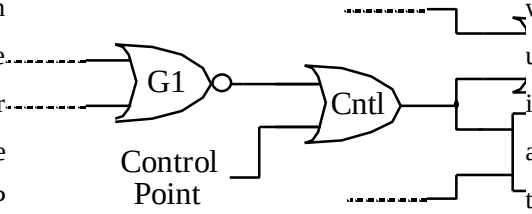



Figure 2.2. Example of Control-1 Point

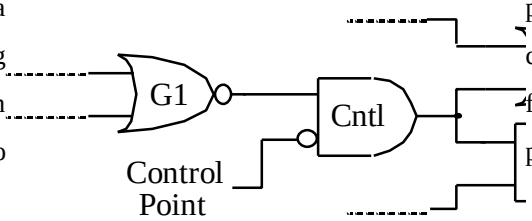



Figure 2.3. Example of Control-0 Point

## 2.2 Test Point Insertion Based on Path Tracing

A new test point insertion method is presented in [Touba 96a]. It provides two innovations compared with previous methods. Instead of using probabilistic techniques for test point placement, fault simulation and a path tracing procedure are used to place both control and observation points. Instead of adding extra scan elements to drive the control points, a few of the existing primary inputs to the circuit are ANDed together to form signals that drive the control points.

### 2.2.1 Using Path Tracing for Test Point Placement

Previous methods insert test points one at a time. The test point that is inserted is selected by a greedy algorithm that estimates which test point would maximize the probability of detecting the undetected faults. The procedure described in [Touba 96a] is not based on probability. Rather, fault-free simulation is performed for each pseudo-random pattern that is applied during BIST. For each pattern, a set of test points that would enable each undetected fault to be detected is computed by tracing sensitized paths in the circuit. After all the information about which test points enable detection of which undetected faults is gathered, a set covering procedure is used to select a set of test points that provides the required fault coverage. Experimental results shown in Appendix IV for benchmark circuits indicate that the path tracing method inserts fewer test points to provide the same or better fault coverage than previous methods. Fewer test points means less area and performance overhead for BIST.

The computation time for this procedure depends on the size of the circuit, the test length, and the number of r.p.r. faults. For each pattern, fault simulation is performed followed by path tracing from each r.p.r. fault site. The fast approximate procedure for tracing sensitized paths that is given in [Abramovici 84] can be used.

In [Touba 96a], a heuristic set covering procedure was used to select the test points. Some experiments were performed to validate the heuristics. Results are shown in Table 2.1 comparing the exact solution to the set covering problem versus the heuristic solution. As can be seen, there was only one case, *s1238*, where using the exact procedure made a difference for these circuits.

Table 2.1. Comparison Between Heuristic and Exact Set Covering Procedures

| Circuit Name | Heuristic Set Covering | | Exact Set Covering | |
|---|---|---|---|---|
| | Con | Obs | Con | Obs |
| s420 | 2 | 0 | 2 | 0 |
| s641 | 1 | 1 | 1 | 1 |
| s713 | 1 | 1 | 1 | 1 |
| s838 | 2 | 0 | 2 | 0 |
| s1238 | 6 | 5 | 5 | 5 |

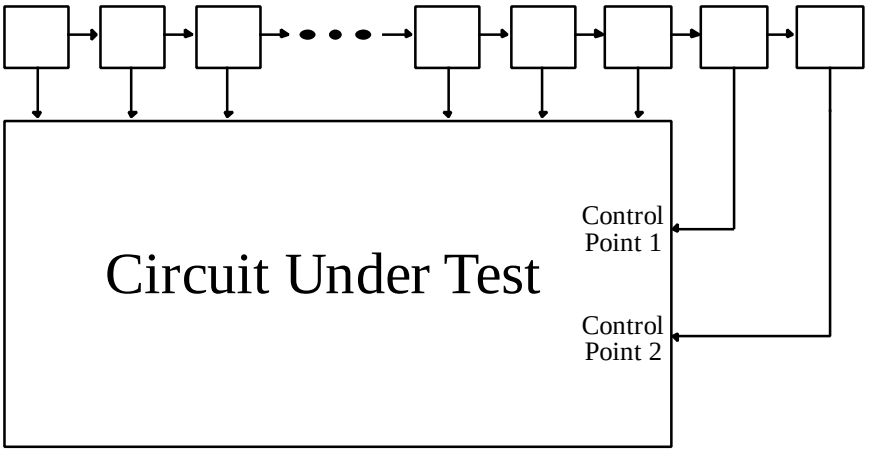



Figure 2.4. Control Points Driven by Extra Scan Elements

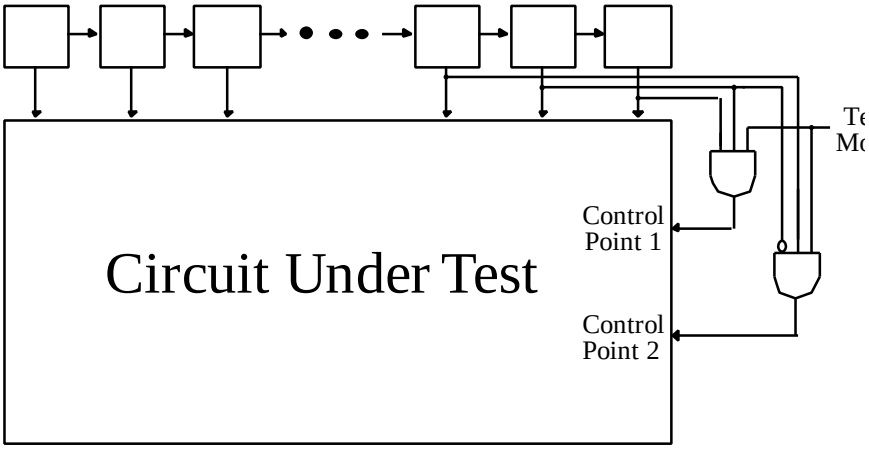



Figure 2.5. Control Points Driven by Pattern Decoding Logic

**2.2.2 Control Point Activation**

Once the test points have been inserted, the remaining task is to design the logic that drives the control points. Previous test point insertion methods add extra scan elements to drive the control points. This is illustrated in Fig. 2.4 where two extra scan elements are added to drive the two control points. The pseudo-random generator is used to shift values into the extra scan elements. Thus, a control point is randomly activated for roughly half of the patterns. This approach limits the potential of each control point. There may be some patterns for which a control point is not activated, but if the control point had been activated, some faults would have been detected. Conversely, there may be some patterns for which the control point is activated, but if it hadn't been activated, some faults would have been detected.

A new approach for driving the control points is presented in [Touba 96a]. As illustrated in Fig. 2.5, pattern decoding logic is used to select those patterns for which the control point is activated. A procedure

for synthesizing this logic in a way that maximizes the effectiveness of each control point for detecting undetected faults is described in [Touba 96a]. In the experimental results in [Touba 96a], on average, fewer than 2 gates were required per control point using this method. This approach eliminates the need for extra scan elements to drive the control points while maximizing the effectiveness of each control point.

As indicated in Fig. 2.5, a test mode line is used to disable the control point during system operation. The test logic is activated during BIST by setting the test mode line to a '1'. When synthesizing the pattern decode logic, all of the patterns that are not applied during BIST are placed in the don't care set. This ensures that the resulting logic does not contain any redundant faults with respect to the patterns applied during BIST, thus the logic is fully tested during BIST.

## 2.3 Test Point Insertion for Non-Feedback Bridging Faults

A common physical defect in MOS technologies is a short between two signal lines which results in a bridging fault [Shen 85], [Ferguson 88]. Although bridging faults are generally more random pattern testable than stuck-at faults [Millman 89], examples are shown in [Touba 96c] to illustrate that some bridging faults are much less random pattern testable than stuck-at faults. Data is presented which indicates that even after inserting test points that result in 100% single stuck-at faults coverage, many bridging faults are still not detected. A test point insertion procedure that targets both single stuck-at faults and bridging faults is presented in [Touba 96c].

Bridging faults can be divided into two classes. Feedback bridging faults are those in which there is a path in the fault-free circuit from one of the shorted lines to the other thereby creating feedback in the fault circuit. Non-feedback bridging faults are those for which no feedback is introduced when the two lines are shorted together. Feedback bridging faults may add state causing the circuit to no longer be combinational, and thus are more complicated to simulate. Since feedback bridging faults have been found to be easier to detect than non-feedback bridging faults [Millman 88], only non-feedback bridging faults were considered in [Touba 96c]. However, the techniques described in [Touba 96c] can be applied to feedback bridging faults in a straightforward manner. The only difference is the added complexity for simulation.

In [Touba 96c], a fast fault simulation procedure for identifying random-pattern-resistant non-feedback bridging faults is described. Using this procedure, the path tracing method described in [Touba 96c] can be enhanced to target both single stuck-at faults and non-feedback bridging faults. The experimental results shown in [Touba 96c] indicate that by considering both types of faults when selecting the location of the test points, higher fault coverage can be obtained with little or no increase in overhead. Thus, the test point insertion procedure described in [Touba 96c] is a low-cost way to improve the quality of built-in self-test.

## 2.4 Logic Synthesis of Random Pattern Testable Circuits

Instead of designing a circuit and then inserting test points to make it random pattern testable, why not consider random pattern testability during logic synthesis? That is the idea presented in [Touba 94]. Given a two-level representation of a circuit and a constraint on the minimum fault detection probability

(threshold below which faults are considered r.p.r.), a testability-driven factoring procedure that satisfies the constraints while minimizing the literal count is described in [Touba 94]. The strategy is to identify r.p.r. faults in the two-level starting point, and then find factors that "eliminate" these faults. Once the r.p.r. faults have been eliminated, normal logic optimization using random pattern testability preserving logic transformations can then proceed since such transformations will not introduce new r.p.r. faults. It is proven in [Touba 94] that algebraic factoring is random pattern testability preserving and that random pattern testability preserving transformations are a superset of test-set preserving transformations.

As the minimum probability threshold is increased, a point is reached where some r.p.r. faults cannot be eliminated by algebraic factoring alone. When this is the case, test points are inserted during the synthesis process in order to generate a random pattern testable implementation. Factors are chosen which maximize the effectiveness of each test point thereby minimizing the total number of test points that are required.

Experimental results are shown in [Touba 94] comparing the implementations generated by the proposed procedure with the implementations generated using the algebraic and rugged scripts in SIS 1.1 (an updated version of MIS [Brayton 87]). The proposed procedure significantly reduces the pseudo-random pattern test length required for 100% fault coverage with only a modest increase in area. For many circuits, the test length was reduced by an order of magnitude or more with less than 10% increase in area. The reason for the area overhead is the fact that in order to satisfy the random pattern testability constraints, the proposed procedure must select some factors based on improving the testability instead of reducing the literal count. Note that the proposed procedure need only be used for logic blocks containing r.p.r. faults, so the overhead penalty is only incurred for a small portion of an overall design.

A limitation of the method proposed in [Touba 94] is that it requires a two-level representation as a starting point thereby limiting its application to control circuits and other circuits that can be flattened (i.e., two-level representation is not exponential). However, control circuits are an important application because they can contain large fan-in cubes that cause r.p.r. faults.

Some other work in logic synthesis of random pattern testable circuits has been published after [Touba 94]. The work in [Chiang 94] was done independently. The synthesis procedure in [Chiang 94] is based on single and double cube divisors [Rajski 92] and does not consider test points. It uses an approximate method for computing the effect of each factor on fault detection probabilities whereas the method used in [Touba 94] is exact. New exclusive-or based transformations were introduced in [Chatterjee 95] which can be used to improve random pattern testability. These transformations can be used in conjunction with those in [Touba 94] to provide even better results.

# Chapter 3
# Test Pattern Generator Design

If pseudo-random BIST provides insufficient fault coverage, instead of modifying the circuit-under-test, another option is to modify the test pattern generator. This involves augmenting the pseudo-random pattern generator with additional logic to generate patterns that detect the r.p.r. faults. In some cases this is the only option because it is either not possible or not desirable to modify the circuit-under-test (e.g., if it is a macrocell, core, or proprietary design).

There are two types of test pattern generators: serial ("test-per-scan") and parallel ("test-per-clock"). Figure 3.1 shows a diagram for a serial BIST scheme. A serial sequence of bits is shifted into a scan chain. When a full pattern has been shifted into the scan chain, it is applied to the circuit-under-test and the response is loaded back into the scan chain and shifted out to a serial signature register for compaction as the next pattern is shifted in. Figure 3.2 shows a diagram for a parallel BIST scheme. A test pattern is applied to the circuit-under-test each clock cycle and the response is loaded into a parallel signature register (MISR) for compaction.

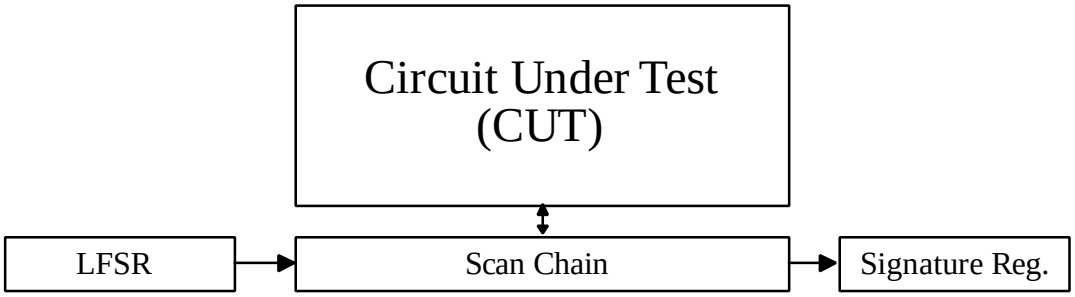



Figure 3.1. Block Diagram for Serial BIST Scheme ("Test-Per-Scan")

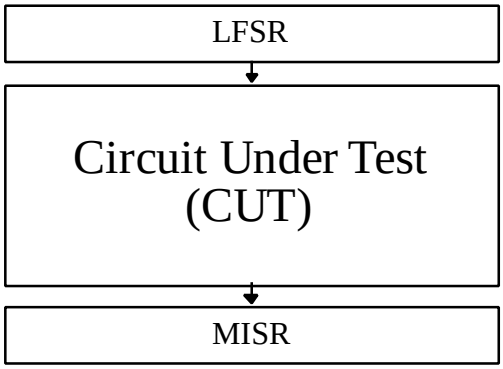



Figure 3.2. Block Diagram for Parallel BIST Scheme ("Test-Per-Clock")

This chapter begins with a survey of the previous work that has been done in designing test pattern generators and then summarizes the new techniques presented in Appendices II, III, and VI.

## 3.1 Previous Work in Test Pattern Generator Design

Two approaches for improving the fault coverage for a pseudo-random pattern generator are weighted pattern testing and mixed-mode testing. *Weighted pattern testing* involves adding logic to bias the pseudo-random patterns towards those that detect the r.p.r. faults. *Mixed-mode* testing involves adding logic to generate deterministic patterns that detect the faults that the pseudo-random patterns miss.

### 3.1.1 Weighted Pattern Testing

Weighted pattern testing is performed by weighting the *signal probability* (probability that the signal is a '1') for each input to the circuit-under-test. Two issues in weighted pattern testing are what set of weights to use and how to generate the weighted signals. Many techniques have been proposed for computing weight sets [Bardell 87]. It has been shown that for most circuits, multiple weight sets are required to achieve sufficient fault coverage [Wunderlich 88]. For BIST, the weight sets must be stored on-chip and control logic is needed to switch between them which can result in a lot of overhead.

In order to reduce the BIST overhead for weighted pattern testing, researchers have looked for efficient methods for on-chip generation of weighted patterns. Wunderlich proposed a Generator of Unequiprobable Random Tests (GURT) in [Wunderlich 87] that requires very little hardware overhead but is limited to only one weight set. Hartmann and Kemnitz proposed a method in [Hartmann 93] that uses a modified GURT structure and described test pattern generators for the *C2670* and *C7552* benchmark circuits [Brglez 85] that require very little overhead. However, both of these methods are not general methods because they use only a single weight set and therefore will not provide sufficient fault coverage for many circuits. Methods that use multiple weight sets with 3 different weight values (0, .5, and 1) were described in [Pomeranz 93] and [AlShaibi 94]. These methods essentially “fix” the value of certain inputs while random patterns are being applied. The method in [Pomeranz 93] uses 3-gate modules to fix the values while the method in [AlShaibi 94] uses specially designed flip-flops. Techniques for generating weighted random patterns using inhomogeneous cellular automata were described in [Neebel 93, 94].

Less weight logic is required for serial test pattern generation (“test-per-scan”) than for parallel test pattern generation (“test-per-clock”). The weight logic can be placed at either the input of the scan chain as described in [Brglez 89] or in the individual scan elements themselves as described in [Muradali 90].

### 3.1.2 Mixed-Mode Testing

In the simplest case, mixed-mode testing can be performed by using an LFSR to generate pseudo-random patterns to detect the random pattern testable faults and then loading deterministic test patterns for the random pattern resistant faults from a ROM. The problem with this approach is that the size of the required ROM is often prohibitive. Several compression techniques have been proposed for

reducing the size of the ROM [Agarwal 81], [Aboulhamid 83], [Dandapani 84], [Edirisooriya 92], [Dufaza 93].

Instead of storing the test patterns themselves in a ROM, techniques have been developed for storing LFSR seeds that can be used to generate the test patterns [Koenemann 91]. The LFSR that is used for generating the pseudo-random patterns is also used for generating the deterministic patterns by reseeding it with computed seeds. Since the seeds are smaller than the test patterns themselves, they require less ROM storage. One problem is that for a normal LFSR with a fixed feedback polynomial, it may not always be possible to find a seed that will generate a required deterministic test pattern. A solution to that problem was proposed in [Hellebrand 92] in which a multiple-polynomial LFSR (MP-LFSR) is used. An MP-LFSR is an LFSR with a reconfigurable feedback network. In [Hellebrand 92], a polynomial identifier is stored with each seed to select the feedback polynomial that will be used for that seed as illustrated in Figure 3.3. Techniques for "merging" and "concatenating" test patterns to reduce the number of LFSR seeds that need to be stored were proposed in [Venkataraman 93] and [Hellebrand 95a]. Even further reduction can be achieved by using variable-length seeds [Zacharia 95] and a special ATPG algorithm [Hellebrand 95b].

Another approach for mixed-mode testing is to design a special counter that generates a deterministic set of test patterns. Daehn and Mucha, in [Daehn 81], proposed using a non-linear LFSR. Akers and Jansz, in [Akers 89], proposed using an LFSR followed by a linear network of XOR gates. Dufaza and Cambon, in [Dufaza 91], proposed using an LFSR with a reconfigurable feedback network. None of these techniques scales well for larger circuits.

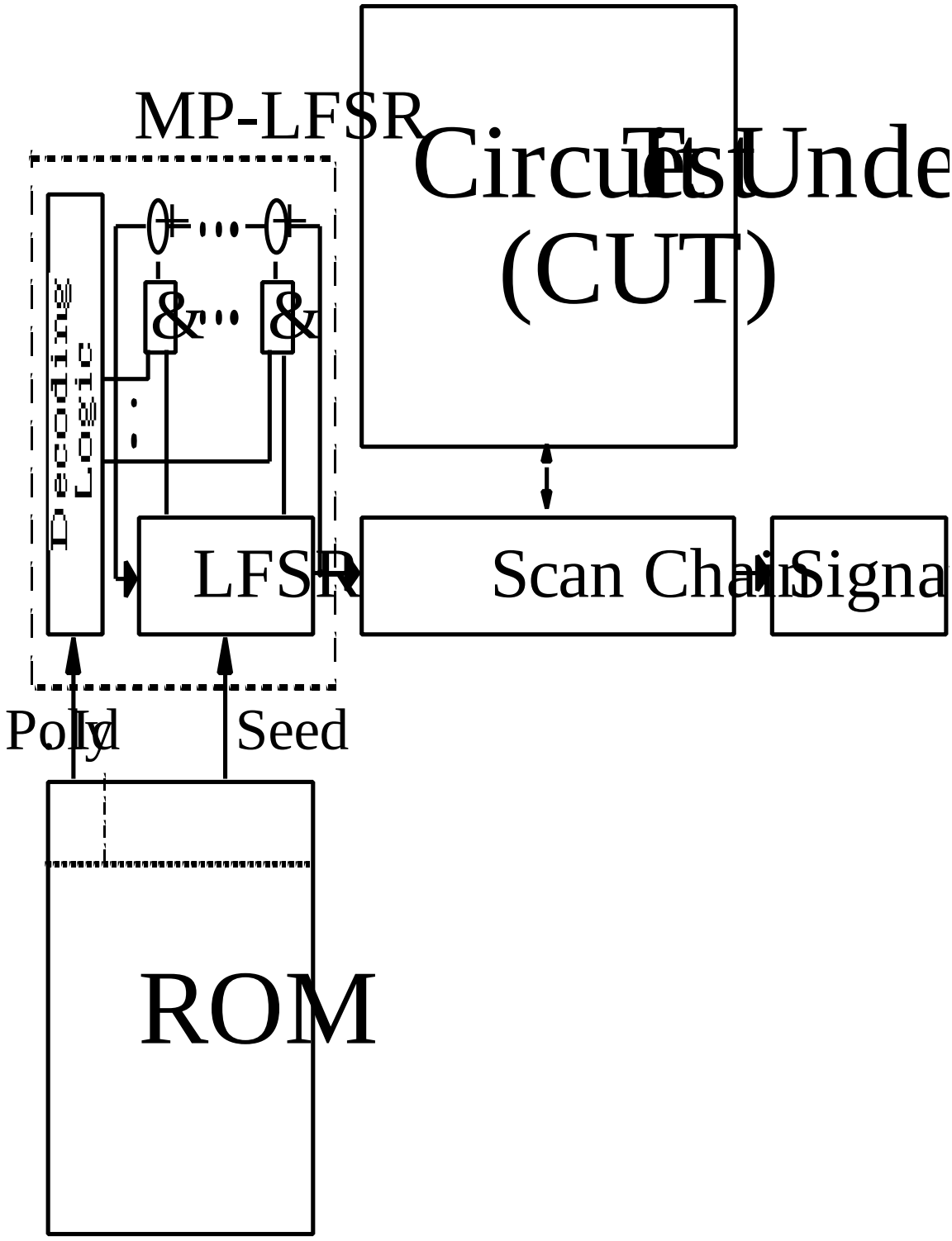
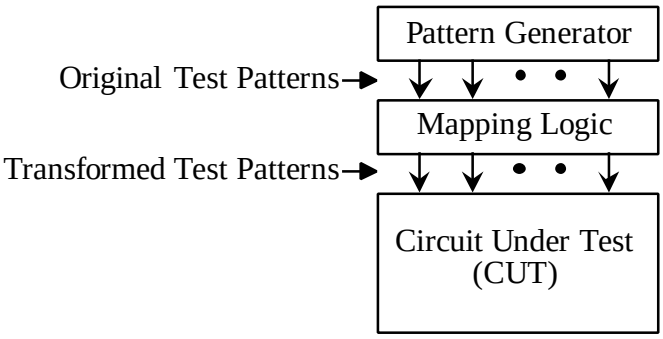



Figure 3.3. Block Diagram for Reseeding using a Multi-Polynomial LFSR (MP-LFSR)

### 3.2 Synthesis of Mapping Logic

In weighted pattern testing, weight logic is placed at the output of the LFSR. One way to view this weight logic is that it transforms each original pattern generated by the LFSR into a new pattern that is applied to the circuit-under-test. Thus, the original set of patterns generated by the LFSR is mapped into a new set of patterns that provides the required fault coverage. This is illustrated in Fig. 3.4.




| Original Patterns | | Transformed Patterns |
|---|---|---|
| 100110 | ? ? | 001010 |
| 010011 | ? ? | 010011 |
| 011000 | ? ? | 011000 |
| 101101 | ? | 101101 |
| 010111 | ? ? | 111010 |
| 001101 | ? ? | 010101 |
| Cov = 89% | | Cov = 100% |

Figure 3.4. Transforming Pseudo-Random Patterns

In [Touba 95a], the idea of generalizing the “weight” logic to perform any mapping function, not just those that weight signal probabilities, is proposed. A procedure is described for synthesizing combinational mapping logic that can be placed between the LFSR and the circuit-under-test to map the original set of test patterns generated by the LFSR into a new set of patterns that provides the required fault coverage. The strategy for designing the mapping logic is to decode sets of patterns that don’t detect any new faults and map them into patterns that detect the hard-to-detect faults. Results are shown for benchmark circuits which indicate that an LFSR plus a small amount of mapping logic reduces the test length required for a particular fault coverage by orders of magnitude compared with using an LFSR alone. These results were compared with the best weighted pattern testing schemes, and in all cases it was shown that the mapping logic required much less overhead to achieve the same fault coverage for the same test length.

In [Touba 95b], an improved synthesis procedure for designing the mapping logic is described. Given an LFSR and a circuit-under-test, there are many possible mapping functions that will provide the required fault coverage. The problem of finding a mapping function that can be implemented with the smallest number of gates is formulated as one of finding a minimum rectangle in a binate matrix. A heuristic procedure involving EXPAND, IRREDUNDANT, and REDUCE operations (analagous to what is used in ESPRESSO [Brayton 84]), is used to minimize the rectangle cover that corresponds to a mapping function. By iteratively performing global operations, the procedure is able to find better mapping functions thereby synthesizing mapping logic that requires less hardware overhead than other methods. Results indicate that a significant hardware reduction is achieved.

As described in Appendices II and III, the mapping logic is enabled during BIST by using a test mode line. During system operation, the test mode line is set to a '0' to disable the mapping logic. When synthesizing the mapping logic, all of the patterns that are not applied during BIST are placed in the don't care set. This ensures that the resulting mapping logic does not contain any redundant faults with respect to the patterns applied during BIST, thus the mapping logic is fully tested during BIST.

## 3.3 Synthesis of Bit-Fixing Sequence Generator

A new mixed-mode BIST scheme is described in [Touba 96b] for circuits with scan. Deterministic test patterns that detect the random-pattern-resistant faults are embedded in a pseudo-random sequence of bits generated by an LFSR. This is accomplished by altering the pseudo-random sequence of bits by adding logic at the LFSR’s serial output to “fix” certain bits. As illustrated in Fig. 3.5, logic is added to generate a bit-fixing sequence that alters the pseudo-random sequence by causing certain bits to be fixed to either a ‘1’ or a ‘0’. A procedure is described for designing the bit-fixing sequence generator in a way that minimizes area overhead.

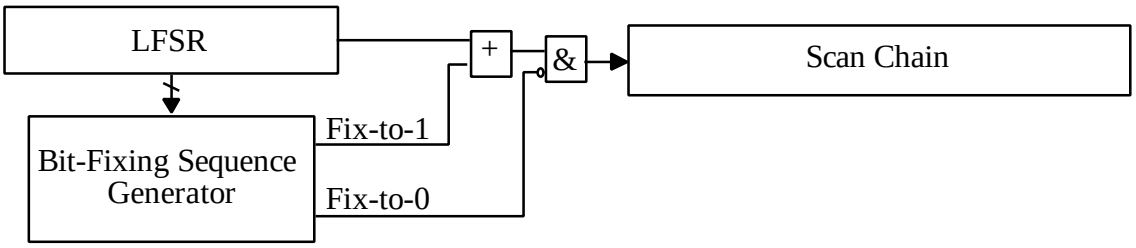



Figure 3.5. Logic for Altering the Pseudo-Random Bit Sequence

Previous mixed-mode schemes for serial pattern generation ("test-per-scan") are based on storing compressed data in a ROM. In the proposed procedure, no data is stored in a ROM, rather a multilevel circuit is used to dynamically fix bits in a way that exploits bit correlation among the test patterns for the random-pattern-resistant faults. Small numbers of correlated bits are fixed in selected pseudo-random patterns to make the pseudo-random patterns match the test patterns. So rather than trying to compress the test patterns themselves, the proposed scheme essentially compresses the bit differences between the test patterns and a selected set of pseudo-random test patterns. Since there are so many pseudo-random test patterns to choose from, a significant amount of compression can be achieved, resulting in reduced overhead.

Schemes based on reseeding an LFSR require that the LFSR have at least as many stages as the maximum number of specified bits in any test pattern. This is necessary to ensure that a seed can be found to generate each of the test patterns. A hardware tradeoff that is made possible by the scheme presented in [Touba 96b] is that a smaller LFSR can be used for generating the pseudo-random bit sequence. This may cause some faults to not be detected because of linear dependencies in the patterns that are generated, but deterministic test patterns for those faults can be embedded at the expense of additional logic in the bit-fixing sequence generator. Data is presented in [Touba 96b] showing how much logic is required for different sized LFSR's.

The scheme described in [Touba 96b] uses a *one phase test*, the BIST logic runs in the same mode for the entire test length. Thus, the BIST control logic is very simple. Figure 3.6 shows the control logic that is required. If there are *m* stages in the scan chain, then a *mod(m+1)* counter is used to keep track of how many bits have been shifted into the scan chain (it is incremented each clock cycle). While the value of the counter is less than *m*, the scan chain operates in shift mode. When the counter contains the value *m*, then the scan chain operates in system mode to load the response of the circuit into the scan chain. There is also a pattern counter to keep track of how many patterns have been applied to the circuit-under-test. The pattern counter is incremented when the *mod(m+1)* counter contains the value *m*. When the value of the pattern counter equals the test length, then the test is complete. Reset logic is needed to initialize the counters, the signature register, and the LFSR at the start of the test.

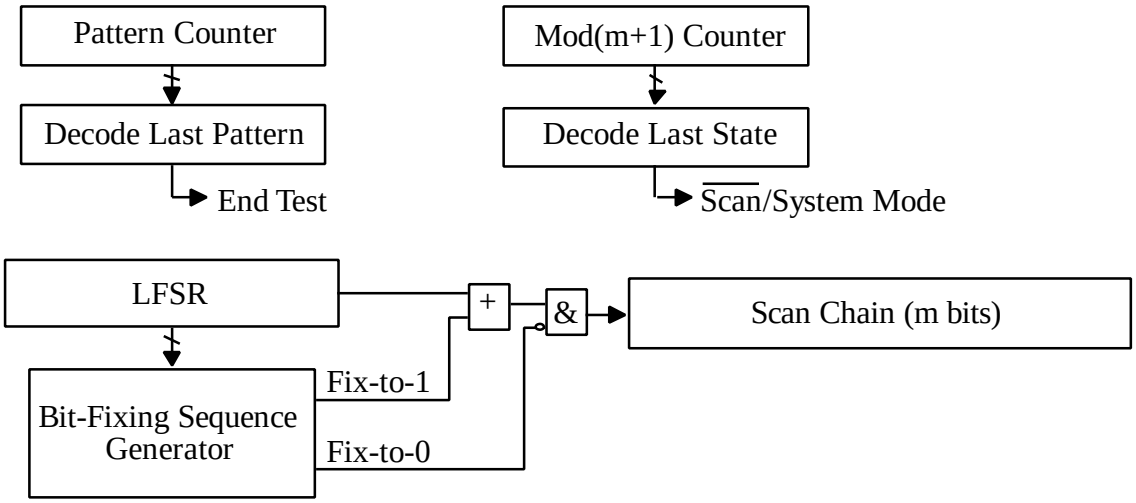



Figure 3.6. Control Logic for Scheme

The advantages of the scheme in [Touba 96b] are that no function logic modification is required, no performance overhead is added beyond what is needed for scan, and the control logic is simple. All of these features combine to make the scheme a very attractive option.

# Chapter 4
# Concluding Remarks

This dissertation summarizes my contributions to automated design of circuits with pseudo-random BIST. BIST is a technique that reduces test and maintenance costs, but it has seen limited use in industry due to area and performance overhead, increased design time, and lack of BIST design tools. Pseudo-random testing is a low-cost approach for BIST, but is only effective for random pattern testable circuits.

If a circuit is not random pattern testable, then the logic synthesis procedure described in [Touba 94] can be used to synthesize a random pattern testable implementation. Testability-driven factoring is used to minimize overhead.

If it is a hand-designed circuit or if it is not synthesizable, then the test point insertion procedure described in [Touba 96a] can be used. This procedure uses path tracing to place both control and observation points and uses pattern decoding logic to drive the control points thereby maximizing the effectiveness of each control point. This results in fewer test points than previous methods. A higher quality test can be obtained by using the procedure in [Touba 96c] to target bridging faults. This procedure significantly improves the bridging fault coverage by inserting just a few additional test points.

If it is not possible to modify the circuit-under-test, then the procedures in Appendices II and III can be used to synthesize mapping logic that can be placed between the LFSR and the circuit-under-test to satisfy the fault coverage requirement. This results in much less overhead compared with weighted pattern testing.

If performance is a major concern, then the procedure in [Touba 96b] can be used to synthesize a bit-fixing sequence generator that embeds deterministic test patterns for the r.p.r. faults in the pseudo-random sequence. This method does not require any performance overhead beyond what is needed for scan.

The end result of the work described in this dissertation is a set of automated synthesis tools that can be used to generate pseudo-random BIST implementations with less overhead and reduced design time. These synthesis tools have been integrated in the TOPS synthesis system.

There are several areas for further investigation. The logic synthesis procedure described in [Touba 94] requires a two-level starting point thereby limiting the types of circuits for which it can be used. Integrating an efficient technique for computing detection probabilities in an arbitrary multilevel circuit would increase the applications for this logic synthesis procedure. The bit-fixing scheme in [Touba 96b] could be combined with a reseeding scheme to further reduce overhead. By reseeding the LFSR with just a few selected seeds to generate some of the least correlated test cubes that require a lot of bit-fixing to embed, it may be possible to significantly reduce the complexity of the bit-fixing sequence generator. In Appendices II and III, the mapping logic was placed at the output of the LFSR and thus adds extra levels of logic between the flip-flops and the function logic thereby affecting system performance. If the mapping

logic could be placed in the feedback portion of the LFSR, then the system performance would not be affected.

# References


[Aboulhamid 83] Aboulhamid, M.E., and E. Cerny, "A Class of Test Generators for Built-In Testing," *IEEE Transactions on Computers* , Vol. C-32, No. 10, pp. 957-959, Oct. 1983.

[Abramovici 84] Abramovici, M., P.R. Menon, and D.T. Miller, "Critical Path Tracing: An Alternative to Fault Simulation," *IEEE Design & Test of Computers*, Vol. 1, pp. 89-93, Feb. 1984.

[AlShaibi 94] AlShaibi, M.F., and C.R. Kime, "Fixed-Biased Pseudorandom Built-In Self-Test for Random Pattern Resistant Circuits," *Proc. of International Test Conference*, pp. 929-938, 1994.

[Agarwal 81] Agarwal, V.K., and E. Cerny, "Store and Generate Built-In Testing Approach," *Proc. of FTCS-11*, pp. 35-40, 1981.

[Akers 89] Akers, S.B., and W. Jansz, "Test Set Embedding in a Built-In Self-Test Environment," *Proc. of International Test Conference*, pp. 257-263, 1989.

[Bardell 87] Bardell, P.H., W.H. McAnney, and J. Savir, *Buit-In Test for VLSI: Pseudorandom Techniques*, New York: Wiley, 1987.

[Brayton 84] Brayton, R.K., G.D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Boston: Kluwer Academic Publishers. 1984.

[Brayton 87] Brayton, R.K., R. Rudell, A. Sangiovanni-Vincentelli, A.R. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Transactions on Computer-Aided Design*, Vol. 6, Nov. 1987, pp. 1062-1081.

[Briers 86] Briers, A.J., and K.A.E. Totton, "Random Pattern Testability by Fast Fault Simulation," *Proc. of International Test Conference*, pp. 274-281, 1986.

[Brglez 84] Brglez, F., "On Testability of Combinational Networks," *Proc. of International Symposium on Circuits and Systems*, pp. 221-225, 1984.

[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., G. Gloster, and G. Kedem, "Hardware-Based Weighted Random Pattern Generation for Boundary Scan," *Proc. of International Test Conference*, pp. 264-274, 1989.

[Chatterjee 95] Chatterjee, M., D.K. Pradhan, and W. Kunz, "LOT: Logic Optimization with Testability - New Transformations using Recursive Learning," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, 1995.

[Chiang 94] Chiang, C.-H., and S.K. Gupta, "Random Pattern Testable Logic Synthesis," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pp. 125-128, 1994.

[Cheng 95] Cheng, K.-T., and C.J. Lin, "Timing-Driven Test Point Insertion for Full-Scan and Paritial-Scan BIST," *Proc. of International Test Conference*, pp. 506-514, 1995.

[Daehn 81] Daehn, W., and J. Muncha, "Hardware Test Pattern Generation for Built-In Testing," *Proc. of Int. Test Conf.*, pp. 110-113, 1981.

[Dandapani 84] Dandapani, R., J. Patel, and J. Abraham, "Design of Test Pattern Generators for Built-In Test," *Proc. of International Test Conference*, pp. 315-319, 1984.

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

[Dufaza 93] Dufaza, C., C. Chevalier, and L.F.C. Lew Yan Voon, "LFSROM: A Hardware Test Pattern Generator for Deterministic ISCAS85 Test Sets," *Proc. of AsianTest Symposium*, pp. 160-165, 1993.

[Edirisooriya 92] Edirisooriya, G., and J.P. Robinson, "Design of Low Cost ROM Based Test Generators," *Proc. of VLSI Test Symposium*, pp. 61-66, 1992.

[Eichelberger 83] Eichelberger, E.B., and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test," *IBM Journal of Research and Development*, Vol. 27, No. 3, pp. 265-272, May 1983.

[Ferguson 88] Ferguson, F.J., and J.P. Shen, "A CMOS Fault-Extractor for Inductive Fault Analysis," *IEEE Transactions on Computer-Aided Design*, Vol. 7, No. 11, pp. 1181-1194, Nov. 1988.

[Hartmann 93] Hartmann, J., and G. Kemnitz, "How to Do Weighted Random Testing for BIST," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pp. 568-571, 1993.

[Hayes 74] Hayes, J.P., and A.D. Friedman, "Test Point Placement to Simplify Fault Detection," *IEEE Transactions on Computers* , Vol. C-23, No. 7, pp. 727-735, Jul. 1974.

[Hellebrand 92] Hellebrand, S., S. Tarnick, and J. Rajski, "Generation of Vector Patterns Through Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *Proc. of International Test Conference*, pp. 120-129, 1992.

[Hellebrand95a] Hellebrand, S., 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 Registers," *IEEE Transactions on Computers*, Vol. 44, No. 2, pp. 223-233, Feb. 1995.

[Hellebrand95b] Hellebrand, S., B. Reeb, S. Tarnick, and H.-J. Wunderlich, "Pattern Generation for a Deterministic BIST Scheme," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, 1995.

[Iyengar 89] Iyengar, V.S., and D. Brand, "Synthesis of Pseudo-Random Pattern Testable Designs," *Proc. International Test Conference*, pp. 501-508, 1989.

[Koenemann 79] Koenemann, B., J. Mucha, and G. Zwiehoff, "Built-in Logic Block Observation Technique," *Proc. of International Test Conference*, pp. 140-150, 1979.

[Koenemann 91] Koenemann, B., "LFSR-Coded Test Patterns for Scan Designs," *Proc. of European Test Conference*, pp. 237-242, 1991.

[Krasniewski 89] Krasniewski, A., and S. Pilarski, "Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 8, No. 1, pp. 46-55, Jan. 1989.

[Krishnamurthy 87] Krishnamurthy, B., "A Dynamic Programming Approach to the Test Point Insertion Problem," *Proc. of the 24th Design Automation Conference*, pp. 695-704, 1987.

[Millman 88] Millman, S.D., and E.J. McCluskey, "Detecting Bridging Faults with Stuck-At Test Sets," *Proc. of International Test Conference*, pp. 773-783, 1988.

[Millman 89] Millman, S.D., and E.J. McCluskey, "Pseudorandom Test for Bridging Faults," *CRC Technical Report 89-7*, Stanford University, Dec. 1989.

[Muradali 90] Muradali, F., V.K. Agarwal, and B. Nadeau-Dostie, "A New Procedure for Weighted Random Built-In Self-Test," *Proc. of International Test Conference*, pp. 660-668, 1990.

[Neebel 93] Neebel, D.J., C.R. Kime, "Inhomogeneous Cellular Automata for Weighted Random Pattern Generation," *Proc. of International Test Conference*, pp. 1013-1022, 1993.

[Neebel 94] Neebel, D.J., C.R. Kime, "Multiple Weighted Cellular Automata," *Proc. of VLSI Test Symposium*, pp. 81-86, 1994.

[Pomeranz 93] Pomeranz, I., and S.M. Reddy, "3-Weight Pseudo-Random Test Generation Based on a Deterministic Test Set for Combinational and Sequential Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 12, No. 7, pp. 1050-1058, Jul. 1993.

[Rajski 92] Rajski, J., and J. Vasudevamurthy, "The Testability-Preserving Concurrent Decomposition and Factorization of Boolean Expressions," *IEEE Transactions on Computer-Aided Design*, Vol. 11, No. 6, Jun. 1992, pp. 778-793.

[Savaria 91] Savaria, Y., M. Youssef, B. Kaminska, and M. Koudil, "Automatic Test Point Insertion for Pseudo-Random Testing," *Proc. of International Symposium on Circuits and Systems*, pp. 1960-1963, 1991.

[Seiss 91] Seiss, B.H., P.M. Trouborst, and M.H. Schulz, "Test Point Insertion for Scan-Based BIST," *Proc. of European Test Conference*, pp. 253-262, 1991.

[Shen 85] Shen, J.P., W. Maly, and F.J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits," *IEEE Design & Test of Computers*, pp. 13-26, Dec. 1985.

[Stroud 88] Stroud, C.E., "Automated BIST for Sequential Logic Synthesis," *IEEE Design & Test of Computers*, pp. 22-32, Dec. 1988.

[Touba 94] Touba, N.A., and E.J. McCluskey, "Automated Logic Synthesis of Random Pattern Testable Circuits," *Proc. of International Test Conference*, pp. 174-183, 1994.

[Touba 95a] Touba, N.A., and E.J. McCluskey, "Transformed Pseudo-Random Patterns for BIST," *Proc. of VLSI Test Symposium*, pp. 410-416, 1995.

[Touba 95b] 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.

[Touba 96a] Touba, N.A., and E.J. McCluskey, "Test Point Insertion Based on Path Tracing," *Proc. of VLSI Test Symposium*, pp. 2-8, 1996.

[Touba 96b] Touba, N.A., and E.J. McCluskey, "Altering a Pseudo-Random Bit Sequence for Scan-Based BIST," *Proc. of International Test Conference*, 1996.

[Touba 96c] Touba, N.A., and E.J. McCluskey, "Test Point Insertion for Non-Feedback Bridging Faults," *Technical Report No. 96-3,* Center for Reliable Computing, Stanford University, Stanford, CA, Aug. 1996.

[Venkataraman 93] Venkataramann, S., J. Rajski, S. Hellebrand, and S. Tarnick, "An Efficient BIST Scheme Based on Reseeding of Multiple Polynomial Linear Feedback Shift Registers," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pp. 572-577, 1993.

[Wunderlich 87] Wunderlich, H.-J., "Self-Test Using Unequiprobable Random Patterns," *Proc. of FTCS-17*, pp. 258-263, 1987.

[Wunderlich 88] Wunderlich, H.-J., "Multiple Distributions for Biased Random Test Patterns," *Proc. of International Test Conference*, pp. 236-244, 1988.

[Youssef 93] Youssef, M., Y. Savaria, and B. Kaminska, "Methodology for Efficiently Inserting and Condensing Test Points," *IEE Proceedings-E*, Vol. 140, No. 3, pp. 154-160, May 1993.

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