Testability Analysis and Insertion for RTL Circuits Based on Pseudorandom BIST

Joan Carletta Christos Papachristou
Department of Computer Engineering and Science
Case Western Reserve University
Cleveland, OH 44106

Abstract

A testability analysis technique for Built-In Self-Test (BIST) at the system level is presented. While based on previous approaches, the model has several significant new features, including an iterative technique for modeling indirect feedback and an extension to the circular BIST methodology. Additionally, a new preprocessing transformation enables the correct modeling of word-level correlation. Examples validate the model, and demonstrate its applicability to test point insertion.

1 Introduction

The goal of this research is to develop a new method for BIST analysis at the register transfer level (RTL). RTL circuits consist of interconnections of registers, functional units (ALUs), multiplexers and buses. The analysis is done via testability metrics that measure the controllability and observability of individual registers. A new metric for observability, *transparency*, is introduced in this paper.

The motivation for this work lies in BIST insertion. Both conventional BIST [AKS93] and the newer circular BIST and circular self-test path technique [PiKK92, Stro88, POLB88] are well-suited for automatic circuit insertion at the RTL. Traditionally, each ALU in a circuit is made directly testable by placing test registers to generate test patterns at the ALU's inputs, and test registers to compact the responses at the ALU's output. However, it may not be necessary to add this many test registers [ChPa91]. For example, suppose that the input registers to the ALU are not directly controllable, but that they still generate patterns that are random enough to effectively test the ALU; in this case, there is no need to replace the normal system registers with more expensive, slower test registers. Thus, in selecting test registers, cost and speed may be traded off against test effectiveness. The testability metrics described in this paper can be used to evaluate various BIST configurations for a given RTL structure, and thus provide a mechanism for the cost / test effectiveness tradeoff.

The Markov model used here to compute the testability metrics is more general than previous models by Chuang and Gupta [ChGu89] and Kim, Ha, and Tront [KiHT88]. A preprocessing transformation is used to remove reconvergent fanout from RTL circuits, allowing the effects of *word-level correlation* on test quality to be accurately modeled. An iterative technique is developed so that the model can handle circuits with *indirect feedback*, i.e., circuits in which a register feeds back into itself via one or more intermediate registers. Also, the model is extended to include the *circular BIST* methodology as well as conventional BIST.

This work was supported by Semiconductor Research Corporation (SRC) Contract 93-DJ-527 and an SRC Education Alliance Graduate Fellowship.

2 Testability Metrics

This section defines metrics for the controllability and observability of registers in (RTL) circuits. We use two metrics for evaluating the controllability of registers. Both are based on the underlying *state probability distributions* for the registers in the circuit. The current *state* of the register is the value being stored in the register. Let X's state probability distribution be denoted by a row vector $\vec{p}_X$; the i^{th} element of the vector, $p_{X,i}$, is the probability that register X is in state i, for $i = 0, 1, 2, \ldots, 2^{|X|} - 1$, where $|X|$ is the bit width of the register. The first metric, *randomness*, is based on the entropy I_X of the register [ChPa91, ThAb89]:

$$MR(X) = \frac{I_X}{|X|} = \frac{1}{|X|} \sum_{i=0}^{2^{|X|}-1} p_{X,i} \log_2 \frac{1}{p_{X,i}}. \quad (1)$$

Randomness ranges from a value of 0 for registers whose state is constant, to a value of 1 for registers that generate uniformly distributed pseudorandom patterns. The second metric, *expected state coverage* for a register X, is the fraction of all $2^{|X|}$ possible states that are expected to be generated or *covered* during a testing session of a given length N [PiKK92]:

$$ESC_X(N) = \frac{1}{2^{|X|}} \sum_{i=0}^{2^{|X|}-1} 1 - (1 - p_{X,i})^N. \quad (2)$$

Expected state coverage ranges from $\frac{1}{2^{|X|}}$ for a register that always generates the same test pattern to 1 for a register that generates an exhaustive set of test patterns. We will use a Markov Chain model to compute the underlying state probability distribution vectors, and then derive the randomness and expected state coverage values from the vectors.

Faults in a circuit manifest themselves as *state errors* in the registers; we say that a *state error* occurs at register X if under fault-free conditions, the register has some state i, but in the presence of some fault, the register has another state i', where $i \neq i'$. We define our measure of observability, the *transparency* $MT(X)$ of a register X, as the probability that an arbitrary state error in register X can be propagated to an observable point. Thus, transparency ranges from 0 for a register that is impossible to observe even indirectly to 1 for a register that is directly observable.

As a tool for evaluating transparency, we define a *state-based transparency vector* $\vec{t}_X$ for each register X, where the i^{th} element of the vector, $t_{X,i}$, is the transparency of register

1063-6404/95 $4.00 © 1995 IEEE

X given that the fault-free state of the register is i. We write the transparency of register X as a dot product of this vector with the state probability distribution:

$$MT(X) = \vec{t}_X \cdot \vec{p}_X. \quad (3)$$

The state-based transparency vectors can be computed in bottom-up fashion, starting with the observable points, and moving register by register towards the controllable points. Each observable point has perfect transparency, and therefore has $\vec{t} = [1\ 1\ 1 \ldots 1]$. We move up one register in the circuit by separating the transparency of a register X into two components: the probability of propagating a state error through a single arithmetic logic unit (ALU) to the next register Z in the circuit; and the probability of propagating the state error from that next register Z to an observable point. The first component is an indication of the *sensitivity* of the ALU to changes in values at the input port driven by register X, and the second component is an indication of the transparency of register Z. Both components depend on the state of the register Y that drives the other input port of the ALU.

We measure the sensitivity of the ALU in terms of a *sensitivity matrix* $S^{\otimes}$, where $\otimes$ is the function performed by the ALU. The matrix element $s^{\otimes}_{i,j}$ is the probability that a state error at the lefthand input of the ALU, from i to i', will cause a state error in the output of the ALU, given that the righthand input of the ALU has value j. Thus, $s^{\otimes}_{i,j}$ is the probability that $i \otimes j$ is not equal to $i' \otimes j$. For many functions, the sensitivity matrix can be found analytically; for example, for addition, all elements of the matrix are equal to one. For other functions, a Monte Carlo simulation can be done to evaluate the matrix. Once the sensitivity matrix is known, the state-based transparency vector at the input register X of the ALU can be written as:

$$t_{X,i} = \left(\sum_{j=0}^{2^{|Y|}-1} S^{\otimes}_{i,j}\, p_{Y,j} \right) \left(\sum_{j=0}^{2^{|Y|}-1} t_{Z,i\otimes j}\, p_{Y,j} \right). \quad (4)$$

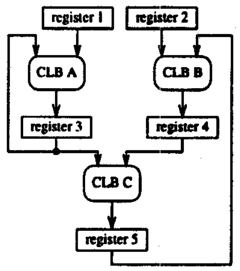
The two sums in this equation are the two components of transparency.

3 A Model for BIST Analysis

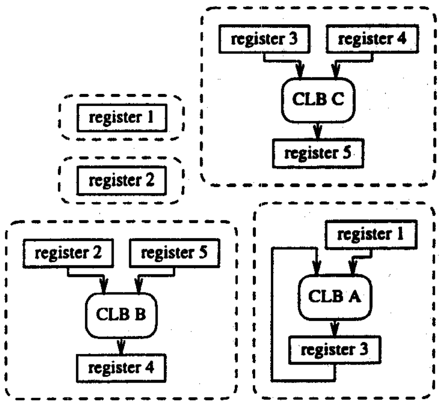
This section develops a Markov model to compute analytical values for the state probability distribution of each register in an RTL circuit; the testability metrics are computed from the probability distributions using the formulas of the previous section.

3.1 Partitioning a circuit for analysis

A circuit with even a moderate number of registers has an unmanageably large number of system states. As a result, we must *partition* the circuit into smaller pieces, and do a probabilistic analysis of the partitions. Each circuit partition contains the information necessary to analyze a single register. The rules for partitioning are simple: create one partition for each register of the circuit, where the register serves as the output register of the partition. The partition consists of the register, any combinational logic that drives the register, plus the registers that serve as inputs to that combinational logic. The partitions overlap in the sense that the output register



(a) A simple RTL circuit.



(b) The circuit partitioned.

Figure 1: An example of partitioning for analysis.

of one partition may also be an input register to other partitions. The partition for a register is unique. Figure 1 shows a simple RTL circuit and its partitions. Note that partitioning does not remove feedback loops from the circuit. Direct feedback becomes feedback within a partition, as is the case with register 3 of this example, and is resolved by the Markov model. Indirect feedback loops manifest themselves as interdependency among the partitions; for this example, the partitions for registers 4 and 5 are interdependent, since register 5 is an input to the partition for register 4, and vice versa. Thus, analysis of register 4 requires knowledge of register 5's probability distribution, and vice versa. Indirect feedback is resolved by assigning all registers arbitrary initial probability distributions, and using an iterative process to repeatedly analyze registers until the probability distributions converge.

In Section 4, we will see that when this simple partitioning scheme is applied directly to circuits with reconvergent fanout, the Markov model is unable to accurately account for word-level correlation. Section 4 will present a circuit transformation to be used on circuits with reconvergent fanout before partitioning.

3.2 Analyzing a single register

The purpose of analyzing a single register is to find the register's state probability distribution $\vec{p}$. The steps for finding $\vec{p}$ are briefly described here.

Step 1. Find Q, a matrix that describes the register's state transitions when BIST is disabled. The matrix Q has elements

$$q_{ij} = \Pr\,\{\text{next state is } j \mid \text{current state is } i\}\ . \tag{5}$$

The register's next state is written in terms of the part of the circuit contained within the register's partition. The state transition matrix Q depends on the functionality of the partition's combinational logic, the probability distributions of the partition's input registers, and whether there is direct feedback within the partition. When direct feedback is not present, the equation for q_{ij} is reduced to $q_{ij} = \Pr\,\{\text{next state is } j\}$, and all rows of Q are identical.

Step 2. Modify Q to compute C, a matrix that describes the register's state transitions when BIST is enabled. The elements of C are defined by:

$$c_{ij} = \Pr\,\{\text{ next state is } j \mid \text{current state is } i\}, \tag{6}$$

keeping in mind that the BIST is enabled. How C is computed depends on the BIST methodology used. If the register is a left-shifted circular BIST test register, the next state is a bitwise XOR of what the next state would be if BIST were disabled and a shifted version of the register's current state, where a bit from the preceding register on the circular self-test path is shifted into the vacated bit position. Let ϱ denote the probability that this shifted-in bit is a 1; ϱ can be derived from the probability distribution for the preceding register. Each element of C can be expressed in terms of Q and ϱ by the following:

$$c_{ij} = \varrho\, q_{i,j\oplus \mathrm{SHL}(i,1)} + (1-\varrho)\, q_{i,j\oplus \mathrm{SHL}(i,0)} \tag{7}$$

if the register is left-shifted,[*,†] and

$$c_{ij} = \varrho\, q_{i,j\oplus \mathrm{SHR}(i,1)} + (1-\varrho)\, q_{i,j\oplus \mathrm{SHR}(i,0)} \tag{8}$$

if the register is right-shifted.

For the TPGRs and MISRs used in conventional BIST, the idea is the same; each element of C can be expressed in terms of Q by:

$$c_{ij} = q_{i,j\oplus \mathrm{SHL}(i,f(i))} \tag{9}$$

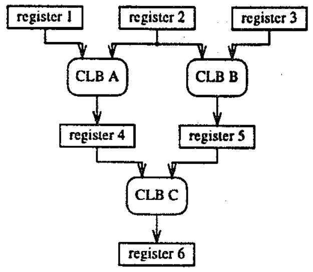
where $f(i)$ is the modulo-2 sum of the values at the feedback taps when the register is in state i.

Step 3. Use C to compute $\vec{p}$, a row vector describing the steady-state probability distribution over the register's state space. Finding $\vec{p}$ involves solving the equation $\vec{p} = \vec{p}C$; $\vec{p}$ is the left eigenvector of C corresponding to eigenvalue 1. We know that such an eigenvector exists and is non-trivial because C is a Markov matrix [Stra88].

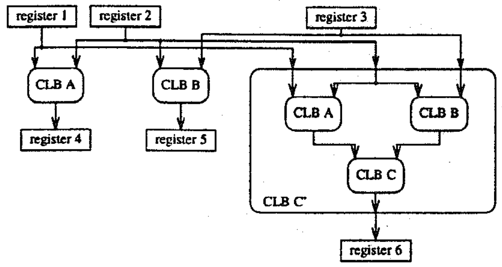
Example results in Section 5 will show how the Markov model is used to analyze the effectiveness of an RTL circuit with BIST. We now turn to a transformation that can be used to process the circuits before applying the Markov model, making the model applicable to a wider variety of circuits by enabling it to accurately handle word-level correlation.

[*] $\oplus$ denotes a bitwise XOR.

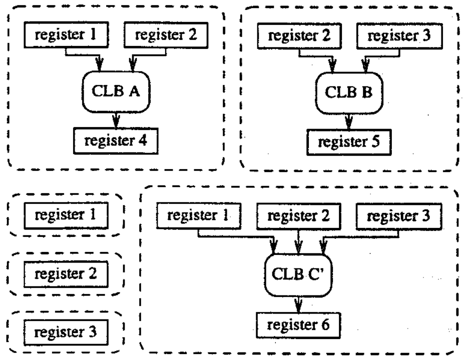
[†] SHL(i,b) and SHR (i,b) denote left and right bit shifts, respectively, of the binary representation for i, with the bit b shifted into the newly vacated bit.



(a) An RTL circuit with reconvergent fanout.



(b) The circuit transformed.



(c) A partitioning of the transformed circuit.

Figure 2: A simple transformation example.

4 A Transformation Technique

The probabilistic analysis described in this paper can be simplified by assuming that there is no correlation among the input registers of a partition, i.e., that one input register's state is independent of every other input register's state. In practice, *word-level correlation* occurs frequently, and is a direct result of fanout in the circuit structure. For example, consider the circuit of Figure 2(a). When a direct partitioning of the circuit is used, information about registers 4 and 5 is used to analyze register 6. The states of these two registers are heavily correlated; both are functions of the state of register 2 at the previous clock. Any probabilistic analysis operating directly on the original circuit should take this correlation into account.

In this work, a transformation that turns a general RTL circuit into a functionally equivalent RTL circuit with no reconvergent fanout is used prior to the BIST analysis. The transformation serves to subsume the reconvergent fanout within the combinational logic, where it can be taken into account more easily; Figure 2(b) shows how, for our example, the reconvergence is subsumed within combinational logic block C'. Part (c) of the figure shows the partitions for the transformed example; since the inputs to each partition are independent, subsequent probabilistic analysis may ignore correlation.

The algorithm for finding the partition for a register starts with the direct partition described in Section 3.1. It then enlarges the partition until all correlation among the input registers of the partition is eliminated. Whether two registers are correlated is determined by tracing through the circuitry driving the registers, looking for a common source register at a common sequential distance. For example, when finding the proper partition for register 6 of Figure 2(a), the algorithm starts by expressing the state of register 6 as a function of the states of registers 4 and 5. Since registers 4 and 5 have a common source, the partition is expanded around registers 4 and 5, so that register 6's state is written as a function of the states of registers 1, 2, and 3.

5 Examples and Results

In this section, the proposed model is used to analyze several example circuits. These examples *validate the model* by showing that the analytical predictions are close to actual results obtained by simulation, and *show how the metrics can be used* to compare different BIST configurations in terms of test quality. Analytical predictions are from the model; actual values are obtained from a computer program that simulates the functioning of an RTL circuit in test mode, computing the registers' states at each time step.

Layout area and critical delay figures given in this section are derived in the COMPASS Design Automation Tool. Critical delay reflects the speed at which the circuit can be clocked. Fault coverage curves are from AT&T's GENTEST fault simulator, and include only those faults within the ALUs. Traditionally, the signature is obtained by observing one element of the circular self-test path for a limited number of clocks at the end of the test session. Since we found this difficult to implement in GENTEST, we observed the element throughout the entire test session. As a result, our experiments neglect some of the probability of aliasing during circuit response compaction.

5.1 A cascade

The first example, shown in Figure 5.2(a), is a four bit wide cascade of adders and multipliers. Testability metrics, both from analysis and simulation, are shown in Figure 5.2(b) for two different circular self-test path choices. Version I of the circuit has *minimal* circular self-test path, that is, only the primary input and primary output registers are included in the path. For this version of the circuit, registers 9 through 12 have low transparency, and registers 13 and 14 have low randomness. In order to boost these low values, we choose to insert additional BIST registers in place of registers 9 through 12 in version II of the circuit. Figure 5.2(b) shows that version II has improved randomness and transparency values. The table also shows good agreement between the analytical values predicted by our Markov model and actual values derived from simulation.

For the circular BIST methodologies, the insertion of a test register boosts both controllability and observability; that is why our primary inputs have high transparency. Note also that replacing registers 9 and 10 with test registers was sufficient to bring the randomness of register 13 up from a low value of .6389 to the more reasonable value of .9295. Figure 5.2(c), which shows fault coverage curves for the two circuit variations, shows that the version II is significantly more testable than version I.

5.2 A low pass filter

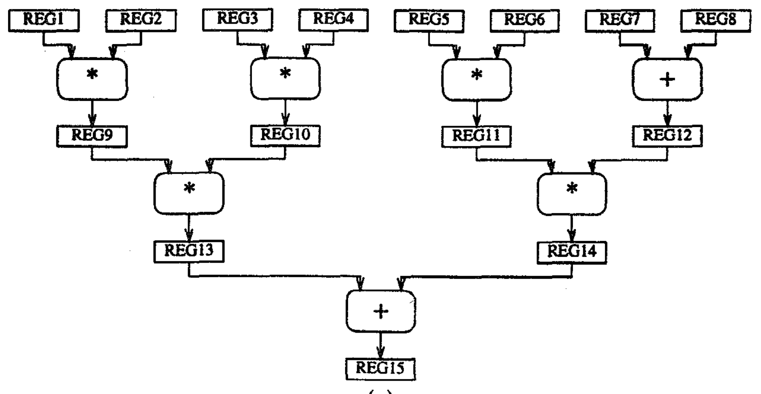
Our second example, shown in Figure 3, is adapted from a low pass filter in [COOS93]. The datapath is nine bits wide, using fixed point numbers with two bits after the binary point. The 25% and 75% ALUs multiply their single inputs by .25 and .75, respectively.

Version I of the circuit has a minimal circular self-test path. Because this circuit has chained ALUs, we calculate metrics not only at the registers, but also at points a and b. For this example, transparency values are trivially very high, because addition is highly sensitive to changes in an input, but the randomness of point b is quite low because the multiplication by 1/4 clears the two most significant bits. In order to improve the randomness at b, we insert a test register that is transparent in normal mode. Version II of the circuit inserts a full nine bit test register at point b. For this example only two of the nine bits actually have low entropy, so we can save area by using a test register of only two bits; we do this in version III. Fault coverage for all versions of the circuit are shown in Figure 3(c).

5.3 An elliptical wave filter

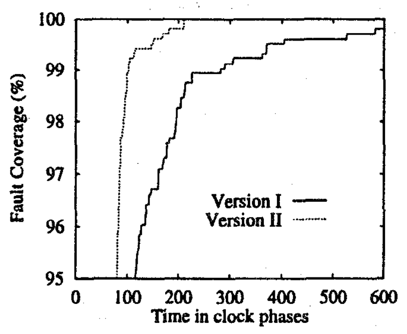
Our final example is derived from a *fifth order elliptical wave filter*[PaKn89]. First, a four bit wide version of the wave filter was synthesized using the high level synthesis system of [HPCN92]. Next, since our test effort focuses on the ALUs, the assumption that multiplexer control would be held constant during testing was made, and so a single path through each multiplexer was chosen. In effect, this allowed the removal of all multiplexers and some registers not used for testing from the circuit; the resulting circuit is shown in Figure 5.3(a). Next, a minimal circular self-test path was added. Testability metrics are shown in the table of Figure 5.3(b). In this case, the randomness and transparency values are quite high, and we elect not to insert additional test features. A fault coverage curve for the circuit is shown in Figure 5.3(c).

The overall execution time for analyzing a circuit is dominated by the time taken for the Markov analysis. The analysis process is iterative in nature, and the time per iteration grows at most linearly with the number of registers. An exact complexity analysis of the overall process has not been done because determining the number of iterations needed is difficult; however, in our experience, most circuits require only a few iterations. The time required to analyze a single register grows exponentially with the size of the register, since each possible register state is examined. The overall execution time is reasonably fast for the four and nine bit wide registers used here; our slowest example required six seconds of real ("wall clock") time on a SUN SPARC *IPC* workstation. However, the analysis is slow for larger 16 or 32 bit registers. It is



(a)

Circuit	Area in λ^2	Critical Delay in ns
I	2247x1447	10.77
II	2338x1519	12.56



(c)

		REG1-8	REG9	REG10	REG11	REG12	REG13	REG14	REG15
MR	Analysis	1	.9295	.9295	.9295	1	.6389	.7915	1
	Simul.	.9999	.9298	.9296	.9295	.9996	.6400	.7912	.9997
ESC	Analysis	.9981	.9756	.9756	.9756	.9981	.7525	.8875	.9981
(in 96 clks)	Simul.	1	.9375	.9375	1	1	.7500	1	1
MT	Analysis	1	.7	.7	.867	.7	1	1	1

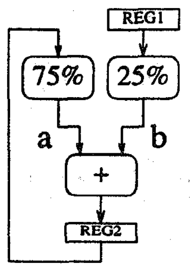
Version I

MR	Analysis	1	1	1	1	1	.9295	.9295	1
	Simul.	.9997	.9998	.9998	.9997	.9998	.9331	.9256	.9998
ESC	Analysis	.9981	.9981	.9981	.9981	.9981	.9756	.9756	.9981
(in 96 clks)	Simul.	1	1	1	1	1	1	1	1
MT	Analysis	1	1	1	1	1	1	1	1

Version II

(b)

Figure 3: The cascade example: (a) RTL structure; (b) testability metrics; (c) area, performance and fault coverage results.



(a)

		REG1	REG2	*a*	*b*
MR	Ana.	1	1	.9444	.7778
	Sim.	.9860	.9870	.9351	.7750
ESC (in	Ana.	.9975	.9975	.7488	.2500
3072 clks)	Sim.	.9961	.9980	.7500	.2520
MT	Ana.	1	1	1	1

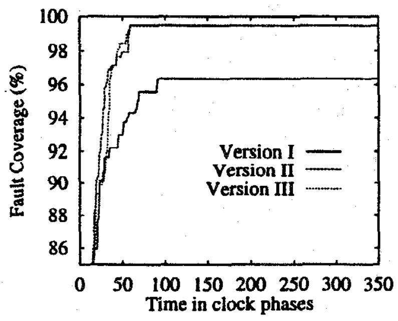
Version I

MR	Ana.	1	1	.9444	1
	Sim.	.9850	.9855	.9345	.9878
ESC (in	Ana.	.9975	.9975	.7488	.9975
3072 clks)	Sim.	1.000	.9941	.7500	1.000
MT	Ana.	1	1	1	1

Versions II and III

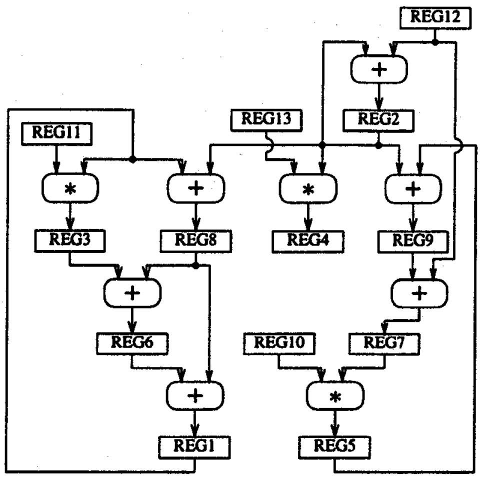
(b)

Circuit	Area in λ^2	Crit. Delay in ns
I	1418x741	18.14
II	1674x897.5	18.18
III	1306x905.5	18.48



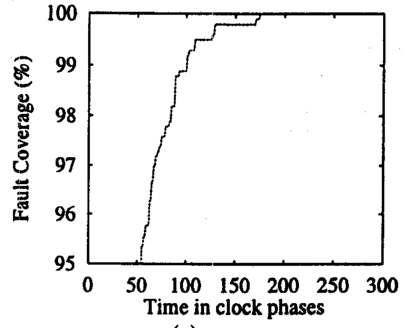
(c)

Figure 4: The low pass filter example: (a) RTL structure; (b) testability metrics; and (c) area, performance and fault coverage results.



(a)

Area in λ^2	Critical Delay in ns
2178x1258.5	12.49



(c)

		REG1	REG2	REG3	REG4	REG5	REG6	REG7	REG8	REG9	REGS10-13
MR	Ana.	1	1	.9295	1	.9295	1	1	1	1	1
	Sim.	.9998	.9998	.9268	.9998	.9303	.9998	.9998	.9998	.9999	.9999-1.000
ESC (in 96 clks)	Ana.	.9981	.9981	.9756	.9981	.9756	.9981	.9981	.9981	.9981	.9981
	Sim.	1	1	1	1	1	1	1	1	1	1
MT	Ana.	1	.8667	1	1	1	1	1	.9333	1	1

(b)

Figure 5: An example derived from an elliptical wave filter: (a) RTL structure; (b) testability metrics; (c) area, performance, and fault coverage results.

hoped that further research will speed up the analysis. Even as it stands now, the model provides insight into test quality, and thus is useful in validating faster heuristics; our research is exploring heuristics to compute the randomness and transparency metrics directly, without reliance on the underlying vectors and matrices.

6 Conclusions

A method for the analysis of conventional and circular BIST at the system level was presented. The method makes use of testability metrics for the controllability and observability of registers of a circuit, and provides a means of measuring the effect of a test insertion decision on test quality. In addition, a transformation technique, used to process the circuit before applying the method, was presented; this transformation technique strengthens the analysis by allowing accurate modeling of the effects of word-level correlation within the circuit. A number of examples were presented comparing analytical results to actual results obtained from simulation, showing that the proposed method is valid.

References

[AKS93] V.D. Agrawal, C.R. Kime, and K.K. Saluja, "A Tutorial on Built-In Self-Test, Part 1: Principles," *IEEE Design & Test of Computers*, pp. 73–82, March 1993.

[ChPa91] S. Chiu and C.A. Papachristou, "A Design for Testability Scheme with Applications to Data Path Synthesis," *Proc. Design Auto. Conf.*, pp. 271–277, June 1991.

[ChGu89] C.C. Chuang and A.K. Gupta, "The Analysis of Parallel BIST by the Combined Markov Chain (CMC) Model," *Proc. Int'l. Test Conf.*, pp. 337–343, October 1989.

[COOS93] W.B. Culbertson, T. Osame, Y. Otsuru, J.B. Shackleford and M. Tanaka, "The HP Tsutsuji Logic Synthesis System," *Hewlett-Packard Journal*, pp. 38–51, August 1993.

[HPCN92] H. Harmanani, C. Papachristou, S. Chiu and M. Nourani, "SYNTEST: An Environment for System-Level Design for Test," *Proc. European Design Auto. Conf.*, pp. 402–407, September 1992.

[KiHT88] K. Kim, D. Ha and J. Tront, "On Using Signature Registers as Pseudorandom Pattern Generators in Built-in Self-Testing," *IEEE Trans. on Computer-Aided Design*, vol. 7, no. 8, pp. 919–928, August 1988.

[PaKn89] P. Paulin and J.P. Knight, "Force-Directed Scheduling for the Behavioral Synthesis of ASIC's," *IEEE Trans. on Computer-Aided Design*, vol. 8, no. 6, pp. 661–679, June 1989.

[PiKK92] S. Pilarski, A. Krasniewski and T. Kameda, "Estimating Testing Effectiveness of the Circular Self-Test Path Technique," *IEEE Trans. on Computer-Aided Design*, vol. 11, no. 10, pp. 1301–1316, October 1992.

[POLB88] M.M. Pradhan, E.J. O'Brien, S.L. Lam and J. Beausang, "Circular BIST with Partial Scan," *Proc. Int'l. Test Conf.*, pp. 719–729, October 1988.

[Stra88] G. Strang, *Linear Algebra and its Applications*, Harcourt Brace Jovanovich, Inc., 1988.

[Stro88] C.E. Stroud, "Automated BIST for Sequential Logic Synthesis," *IEEE Design & Test of Computers*, pp. 22–32, December 1988.

[ThAb89] K. Thearling and J. Abraham, "An Easily Computed Functional Level Testability Measure," *Proc. Int'l. Test Conf.*, pp. 381–390, October 1989.