Sequential Circuit BIST Synthesis using Spectrum and Noise from ATPG Patterns*

Nitin Yogi and Vishwani D. Agrawal
Auburn University, Department of Electrical and Computer Engineering, Auburn, AL 36849, USA
yoginit@auburn.edu, agrawvd@auburn.edu

Abstract

ATPG patterns of a digital sequential circuit contain temporally and spatially ordered bits as well as random (or don't care) bits. We synthesize BIST hardware that mimics these characteristics by controlled mixing of spectral components and noise. A Hadamard digital wave generator circuit produces all required spectral sequences and a weighted pseudorandom bit generator provides random bits. While these two blocks serve the entire circuit under test, specific to each primary input are two small blocks, one that combines the required Hadamard sequences in proper proportions and the other a randomizer that adds the required amount of noise if necessary. As an example, the FlexTest ATPG produced 55110 patterns for s38417, detecting 15472 of 31180 stuck-at faults. Sixty four-thousand of our BIST patterns detected 17020 faults as compared to 4244 detected by a previously reported spectral BIST method utilizing similar hardware overhead.

1. Introduction

For built-in self-test (BIST), scan-based testing has been the prevalent method mainly because it is easy to implement and can often give high fault coverage. However, it exhibits disadvantages like high area overhead, delay penalty, large number of test vectors and long testing times, unnecessary yield loss due to non-functional nature of the tests and the inability of at-speed testing.

Non-scan testing avoids any modification to the circuit and tests the circuit using only the primary inputs and outputs. The above mentioned disadvantages of scan-based testing are eliminated in this method. However, it suffers from two problems. First, a sequential automatic test pattern generator (ATPG) is required for generating vectors for non-scan circuits, whose test generation complexity is high [3, 10]. The achieved fault coverage can be low due to sequential untestability of faults or backtrack limits in ATPG. Second, generating vectors for sequential circuits by random pattern BIST can be nontrivial because of random pattern resistant faults and producing prespecified test sequences from BIST hardware can be expensive.

The proposed spectral BIST method addresses the second issue, which is test pattern generation for BIST. The goal of this work is to replicate the spectral characteristics of a given set of vectors so that BIST vectors have at least the same efficacy as the original vectors. In this work we consider ATPG vectors due to their high quality in terms of fault detection. Such BIST sequences, when longer than the original ATPG sequences, can even produce higher than ATPG coverages.

We use the commercial sequential ATPG tool FlexTest [11] to generate vectors for the circuit under test (CUT) using the stuck-at fault model. The proposed BIST methodology aims at designing hardware with minimum area overhead to recreate the essential spectral properties of ATPG vectors. ATPG vectors are analyzed for their prominent spectral components using Hadamard transform. These spectral components, generated by a Hadamard wave generator, are mixed in appropriate proportions and necessary amount of randomness is added. The major contribution of this work is a novel generalized BIST hardware synthesis procedure using spectrum and noise. It gives higher fault coverage with reduced hardware overhead as compared to existing published result. The proposed method is flexible and adaptable for other binary transforms like Haar transform. We propose a novel hardware approach for combining spectral components.

2. Prior Work

Most existing work on sequential BIST has used random and weighted random patterns. Flip-flops have been modified to increase the number of reachable

*This research is supported in part by the National Science Foundation Grants CCF-0429743 and CNS-0708962.

states [17]. Holding of selected input pseudo-random patterns is another way to increase coverage [13]. Several methods [1, 2, 17] are based on weighted random patterns. Alternatively, counters and comparators are used for test generation [14]. Such methods suffer, in general, from area or delay overheads, high complexity and inability to obtain consistently satisfactory results for a range of circuits.

Spectral analysis of sequential test patterns shows that they exhibit certain periodicities. Giani *et al.* [7] proposed a BIST scheme that replicates beneficial spectral components using an in-built microprocessor. Chen and Hsiao [4] applied that method only to hard to detect faults. Devanathan and Bushnell [5] proposed a Haar wavelet based BIST method for sequential circuits using modulation and correlation.

3. Background

Our BIST synthesis is based on the premise that the spectrum of vectors that detect faults in a circuit reflect important characteristics such as spatial and temporal correlations among the bits of primary input vectors along with some amount of noise or randomness, which corresponds to uncorrelated bits in the test.

We use Walsh functions [16] contained in Hadamard matrices to spectrally analyze binary bit-streams. Walsh functions are a set of orthogonal functions that consist of +1s and -1s. Any bit-stream of k bits can be represented as a linear combination of Walsh functions contained in the Hadamard matrix, $H(\log_2 k)$. Hence, by spectral analysis of bit-streams supplied to primary inputs by test vectors the principal contributing Walsh functions can be determined, as described in the literature [7, 19].

4. Proposed Spectral BIST Method

One may use any commercial or home-grown sequential ATPG tool, FlexTest [11] in our case, to generate test vectors for the CUT. The goal is then to regenerate those vectors in hardware using minimum area overhead such that the original fault coverage of the deterministic ATPG vectors is achievable. The proposed method consists of two steps. In the first step the ATPG vectors are analyzed using Hadamard transform for prominent spectral components as well as their randomness (noise-like) content. Using this information, the spectral BIST is implemented in the second step.

4.1 Spectral Analysis

To analyze the ATPG vectors, bit-streams entering various inputs of the CUT are examined, separately. The 0s and 1s in a bit-stream are represented as -1s and +1s, respectively. Spectral analysis is performed using Hadamard transform [19] to obtain the components of Walsh spectrum in each bit-stream. The ATPG vectors are analyzed in sets, each of length N, where N is the dimension chosen for the Hadamard matrix. Later we shall discuss the selection of a value for N. Discrete sets of ATPG vectors can be analyzed or they can be overlapped. Overlapping of sets helps in sampling the vectors at closer intervals. Analyzing vectors with low overlapping may lead to loss of information, while a high overlapping may lead to sampling of noise in the spectrum. We chose the overlapping length O_V of $2^N/4$ as a compromise. The spectral analysis of a set i of length N for a primary input j of the CUT provides an original component spectrum 'C_{ij}' and a power spectrum (which is the square of C_{ij}) 'P_{ij}'. After the spectral analysis of all sets, the C_i spectra and the P_i spectra are averaged for each input j, separately, to obtain the averaged component spectrum C_j and averaged power spectrum P_j, respectively, for all inputs j. We then perform a threshold filtering on the spectra P_j and C_j using threshold values of T_H and $\sqrt{T_H}$, respectively. A high value of T_H will lead to information loss, while a low value will be ineffective in removing noise. We use a value of T_H equal to 0.5 times the average power of the resultant spectrum.

The averaged power spectrum P_j gives the prominent spectral components in the test vectors for the different inputs j of the CUT, from which we choose the top M prominent components for each input based on their magnitudes. Their signs are determined from the averaged component spectrum C_j. In our case we chose a value $M = 4$.

4.2 Spectral BIST Implementation

The goal of the BIST implementation is to design hardware that will generate vectors exhibiting similar component spectrum C and power spectrum P as the original ATPG vectors. This is achieved by combining the chosen M prominent components in appropriate proportions and phases using a "spectral component synthesizer". Randomness or noise if required is inserted in appropriate amounts to the generated vectors using a "randomizer circuit".

Figure 1 shows the proposed spectral BIST architecture for a CUT with three inputs. The Hadamard wave generator provides spectral components that are combined by the component synthesizer (generally, one per PI) using random bit-streams from the weighted pseudo-random bit-stream generator. The spectral component signals are shown in solid bold lines while the weighted random signals are in dotted bold lines.

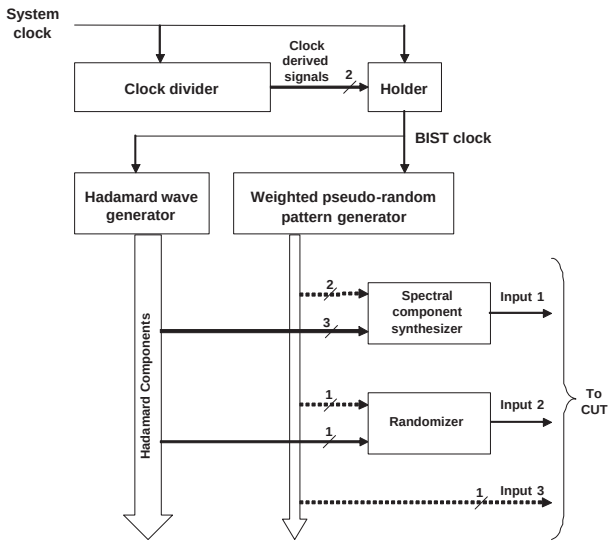

Figure 1. Proposed spectral BIST architecture.

Noise if required is inserted in the combined spectral components by a randomizer (also one per PI) supplied with an appropriate weighted random bit-stream. In this example, the first input of the CUT has three prominent components, which are combined by the component synthesizer. The second input of the CUT has only one prominent component, hence no component synthesizer is required. A randomizer adds the required amount of noise. The third input of the CUT has no prominent components and hence a random bit-stream is directly fed from the pseudo-random bit-stream generator. Next, we describe the components of this BIST architecture.

4.2.1 Hadamard Wave Generator

There is much literature on Walsh function generators [6, 9], which provides implementations giving trade-offs between speed and area. We use a Walsh function generator proposed by Harmuth [9], which uses a counter and has low hardware overhead. A generator of order N, which generates 2^N Walsh functions, requires N flip-flops and $2^N - N - 1$ XOR gates.

The order N of the Hadamard matrix is used in spectral analysis and to implement the Hadamard wave generator. Higher order Hadamard matrices are constructed from lower order matrices [16]. Higher order matrices are better able to characterize a given bit-stream than lower order matrices. However, the area overhead for implementing the Hadamard wave generator increases with the order of the Hadamard matrix as

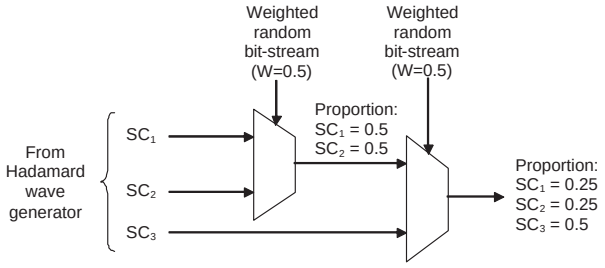

Figure 2. Spectral component synthesizer that combines three spectral components.

mentioned earlier. Considering tradeoffs for the value of N, we chose a lower bound of 4 and an upper bound of 8. The specific value was chosen such that the hardware overhead of the wave generator was approximately 5% of the total circuit area.

4.2.2 Spectral Component Synthesizer

To synthesize vectors, M prominent spectral components are combined in required proportions (equal to their relative power magnitudes) and phases (their signs), both obtained from spectral analysis of Section 4.1. This is achieved using the "spectral component synthesizer" shown in Figure 2. The inputs to the multiplexer are the M chosen spectral components, which are generated by the Hadamard wave generator, and its select line is driven by a weighted random bit-stream. The weighting of the bit-stream is the proportion in which the components are to be combined. Figure 2 depicts an example combining three spectral components SC1, SC2 and SC3 in proportions of 0.25, 0.25 and 0.5, respectively.

4.2.3 Randomizer

Along with the prominent spectral components, some randomness or noise is present in the ATPG vectors which needs to be inserted in the regenerated vectors. Noise is explicitly inserted for inputs for which just a single spectral component is selected. For other inputs, noise is inherently inserted by the spectral component synthesizer through the use of a weighted pseudo-random bit-stream. To estimate the amount of noise, we analyze the runs of 0s and 1s and pick the top 95% of them. The reciprocal of the average of the selected run-lengths gives the average amount of randomness or perturbation applied to the prominent spectral component. The perturbation is then inserted in the generated vectors using the randomizer, which performs an XOR operation with a weighted random bit-stream.

Table 1. Experimental results on fault detection by BIST patterns.

Circuit	Total no. of Faults	Number of faults detected						
		FlexTest	Random		Weighted Random		**Hadamard BIST [this paper]**	Haar BIST [5]
			Without Holding	With Holding	Without Holding	With Holding		
s298	308	273	269	273	273	273	**273**	273
s820	850	793	414	449	744	764	**777**	710
s1423	1515	1443	891	1217	1449	1469	**1468**	1468
s1488	1486	1446	1161	1369	1443	1443	**1443**	1441
s5378	4603	3547	3222	3424	3288	3537	**3603**	3609
s9234	6927	1588	1268	1305	1293	1303	**1729**	1413
s15850	13863	7323	5249	6270	5847	6696	**6844**	5888
s38417	31180	15472	4087	4185	4803	4949	**17020**	4244

4.2.4 Weighted Pseudo-Random Bit-Stream Generator

This circuit generates weighted pseudo-random bit-streams required by the component synthesizer and the randomizer. It uses a 16-bit cellular automata register and a combination of AND-OR gates. The different weights to be generated are determined by the mixing proportions of the spectral components and by the amount of randomness to be added. We quantize the weights as $W = (i \times 2^{-w})$ for $i = 0$ to $2^w - 1$. We used a value of $w = 4$ and also generated two additional weights of 2^{-6} and 2^{-8} for the randomizer.

4.2.5 Holder Circuit

We use vector holding to enhance the fault coverage [8, 12, 13, 23]. It involves holding input vectors constant for several clock cycles while applying the system clock to the circuit under test. Nachman *et al.* [13] give several reasons for the effectiveness of vector holding. A theorem on testing of acyclic sequential circuits gives a relation between the holding length and the sequential depth of the circuit [12, 23]. The number of clock cycles through which an input vector is held unchanged can be determined in several ways [8, 13]. We determine the number of hold cycles 'H_L' from an upper bound on the sequential depth of the circuit [15] rounded to the nearest power of 2. The upper bound on sequential depth [15] is defined as the minimum number of cycles required to initialize a flip-flop (to 0 and 1) and propagate its value to a primary output. In our BIST scheme, vectors are generated with and without holding. First, we generate D vectors without holding and then D/H_L vectors, each held for H_L clock cycles. Although any appropriate value can be chosen for D, an effective value for D was chosen as (ATPG Vector Length/50) rounded to the nearest power of 2. The holding circuit consists of a multiplexer whose inputs are clock signals with periods CLK and $CLK \times H_L$. Its control signal is driven by a clock signal of period $CLK \times (2 \times D)$. The output of the holding circuit is the BIST clock that drives the Hadamard wave generator and the weighted pseudo-random bit-stream generator blocks.

4.2.6 Clock Divider Circuit

Clock divider circuit generates two derived clocks of periods $CLK \times H_L$ and $CLK \times (2 \times D)$ supplied to the holding circuit. It consists of an asynchronous binary counter using $\lceil \log_2(2 \times D) \rceil$ flip-flops. If the BIST environment uses a pattern counter (which is found in many cases) then these clock signals can be obtained from appropriate outputs of the counter.

5 Results

We emulated our BIST method using a MATLAB program. The generated vectors are the same as or are very close to those generated by actual hardware and include quantization errors, which would not be included in a purely software based method [7], thus giving more pragmatic results. We implemented BIST on eight ISCAS'89 benchmark circuits. We inserted a reset signal in the circuits, which is activated once before beginning the BIST session. The reset signal can be a slow-speed signal and thus implemented using minimum resources. ATPG vectors with an initial reset were generated for stuck-at faults using Mentor Graphics tool FlexTest [11]. These ATPG vectors were analyzed for spectrum and noise as described in Section 4. Using this information in the MATLAB program, 64,000 emulated BIST test vectors were generated and then fault simulated again using FlexTest.

Table 1 gives the number of faults detected by our method (shown as Hadamard BIST) for each circuit and they can be compared with results from random and weighted random vectors (both without and

Table 2. Comparison of fault coverage and number of vectors with FlexTest ATPG.

Circuit	FlexTest		Hadamard BIST		
	Fault cov. (%)	No. of vectors	Fault cov. (%) at 64K vectors	Fault cov. (%) at 128K vectors	BIST vectors for FlexTest ATPG cov.
s298	88.64	153	88.64	88.64	757
s820	93.29	1127	91.41	91.88	(!)
s1423	95.25	3882	96.90	96.90	22345
s1488	97.31	736	97.11	97.11	(!)
s5378	77.06	739	78.27	78.67	8984
s9234	22.92	15528	24.96	25.25	8835
s15850	52.82	61687	49.37	52.15	198061
s38417	49.62	55110	54.59	63.07	43240

with holding), ATPG vectors and with the method in [5]. For random, weighted random and the proposed method we generated 64,000 vectors. Weights for the weighted random vectors were obtained from their corresponding ATPG vectors without quantization. Random and weighted random vectors were generated using a software random number generator. We used the same holding scheme as used in our proposed method for the random and weighted random vectors.

As shown in Table 1, our proposed BIST method detects no lesser faults in six out of eight circuits than the existing methods compared. In five out of eight circuits, the proposed method detected at least as many faults as ATPG vectors using 64,000 BIST vectors. Noticeably better results were obtained for s38417, where the proposed method detected 17020 faults as compared to 15472 faults detected by ATPG vectors. The effectiveness of the holding scheme is evident from the results obtained for random and weighted random vectors where most circuits benefited.

The problem of low coverage for some circuits in Table 1 is quite typical of sequential mode testing and is the main reason for the popularity of the scan testing [3]. The coverage of sequential mode BIST can be improved by specific design for testability methods. Besides, it is also essential to identify the sequentially untestable faults to assess the test coverage [22].

Table 2 shows the effectiveness of the proposed BIST method over longer vector sequences. The last column gives the number of vectors required by our BIST method to achieve at least as much fault coverage as ATPG vectors. As observed from columns 4 and 5, the fault coverage gradually increases as more vectors are applied. Also we observe from column 6 that eventually six out of eight circuits were able to achieve at least as much fault coverage as the ATPG vectors. The cells marked with (!) represent cases where our BIST vectors were not able to achieve the ATPG fault coverage although the fault coverages were quite close.

Table 3 shows a comparison of the area overhead in terms of number of transistors and % area overhead of our proposed method with [5]. To calculate the area overhead, we assumed a fixed number of transistors for each type of gate: AND - 6, OR - 6, NAND - 4, NOR - 4, XOR - 6 (pass-transistor logic design), 2 input MUX - 4 (pass-transistor logic design), D flip-flop - 22. In the table we report results for two cases, one where a clock divider circuit is implemented and the other where an existing pattern counter is reused eliminating the need for a clock divider. As compared to [5], the area overhead of our method with a clock divider circuit is lower in three out of eight circuits. It is lower in six out of eight circuits where the clock divider is not used.

6 Conclusion

We present a novel hardware test generation method for sequential circuit BIST. ATPG vectors are analyzed for spectrum and random noise level. The ATPG vectors can be derived from gate, register-transfer, or function level algorithms [18, 19] for fault models like stuck-at, delay [20], or several combined fault models [21]. Our hardware patterns mimic the characteristics of ATPG vectors by controlled mixing of spectral components and noise. We propose a novel circuit for mixing spectral components called the "spectral component synthesizer". Noise is inserted using an XOR circuit. Results show fault coverages either equal to or greater than those of ATPG vectors in six out of eight circuits. Area overheads are moderate compared to existing methods. Comparing alternatives, our method achieved maximum fault coverage (sometimes exceeding that of ATPG) in six out of eight circuits.

References

[1] M. F. Ashaibi and C. R. Kime, "Fixed-Biased Pseudo-random Built-In Self-Test for Random Pattern Resis-

Table 3. BIST area overhead in transistors.

Circuit	No. of transistors in circuit	**Hadamard BIST [this paper]**				Haar BIST [5]	
		with clock divider circuit		without clock divider circuit			
		No. of Transistors	% Area overhead	No. of Transistors	% Area overhead	No. of Transistors	% Area overhead
s298	890	908	102.02	820	92.13	834	93.71
s820	1896	1472	77.64	1340	70.68	1612	85.02
s1423	4624	1637	35.40	1483	32.07	1555	33.63
s1488	4006	1069	26.68	959	23.94	1078	26.91
s5378	12840	2342	18.24	2210	17.21	2487	19.37
s9234	23356	2700	11.56	2502	10.71	2552	10.93
s15850	43696	4908	11.23	4666	10.68	4595	10.52
s38417	108808	3606	3.31	3364	3.09	2135	1.96

tant Circuits," in *Proc. International Test Conf.*, 1994, pp. 929–938.

[2] M. Bershteyn, "Calculation of Multiple Sets of Weights for Weighted Random Testing," in *Proc. International Test Conf.*, 1993, pp. 1031–1040.

[3] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer, 2000.

[4] X. Chen and M. S. Hsiao, "Characteristic Faults and Spectral Information for Logic BIST," in *Proceedings IEEE/ACM International Conf. Computer-Aided Design*, 2002, pp. 294–298.

[5] S. K. Devanathan and M. L. Bushnell, "Test Pattern Generation Using Modulation by Haar Wavelets and Correlation for Sequential BIST," in *Proc. 20th International Conf. VLSI Design*, 2007, pp. 485–491.

[6] L. C. Fernandez and K. R. Rao, "Design of a Synchronous Walsh-Function Generator," *IEEE Trans. Electromagnetic Compatibility*, vol. EMC-19, no. 4, pp. 407–410, Nov. 1977.

[7] A. Giani, S. Sheng, M. S. Hsiao, and V. D. Agrawal, "Novel Spectral Methods for Built-In Self-Test in a System-on-a-Chip Environment," in *Proc. 19th IEEE VLSI Test Symp*, 2001, pp. 163–168.

[8] R. Guo, S. M. Reddy, and I. Pomeranz, "Proptest: A Property Based Test Pattern Generator for Sequential Circuits using Test Compaction," in *Proc. 36th ACM/IEEE Design Automation Conf.*, 1999, pp. 653–659.

[9] H. F. Harmuth, *Transmission of Information by Orthogonal Functions*. Springer-Verlag, 1972.

[10] T. E. Marchok, A. El-Maleh, W. Maly, and J. Rajski, "Complexity of Sequential ATPG," in *Proc. European Design and Test Conf.*, Mar. 1995, pp. 252–261.

[11] Mentor Graphics, *FastScan and FlexTest Reference Manual*, 2004.

[12] H. B. Min and W. A. Rogers, "A Test Methodology for Finite State Machines using Partial Scan Design," *J. Electronic Testing: Theory and Applications*, vol. 3, no. 2, pp. 127–137, 1992.

[13] L. Nachman, K. Saluja, S. Upadhyaya, and R. Reuse, "Random Pattern Testing for Sequential Circuits Revisited," in *Proc. Fault-Tolerant Computing Symp.*, June 1996, pp. 44–52.

[14] I. Pomeranz and S. M. Reddy, "Improved Built-In Test Pattern Generators Based on Comparison Units for Synchronous Sequential Circuits," in *Proc. International Conf. Computer Design*, Oct. 1998, pp. 26–31.

[15] L. Shen, "Genetic Algorithm Based Test Generation for Sequential Circuits," in *Proc. 8th IEEE Asian Test Symp.*, 1999, pp. 179–184.

[16] E. W. Weisstein. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com.

[17] H.-J. Wunderlich, "Multiple Distributions for Biased Random Test Patterns," *IEEE Trans. Computer-Aided Design*, vol. 9, no. 6, pp. 584–593, 1990.

[18] N. Yogi and V. D. Agrawal, "Spectral Characterization of Functional Vectors for Gate-Level Fault Coverage Tests," in *Proc. 10th VLSI Design and Test Symp.*, Aug. 2006, pp. 407–417.

[19] N. Yogi and V. D. Agrawal, "Spectral RTL Test Generation for Gate-Level Stuck-at Faults," in *Proc. 15th IEEE Asian Test Symp.*, 2006, pp. 83–88.

[20] N. Yogi and V. D. Agrawal, "Transition Delay Fault Testing of Microprocessors by Spectral Method," in *Proc. 39th IEEE Southeastern Symp. System Theory*, Mar. 2007, pp. 283–287.

[21] N. Yogi and V. D. Agrawal, "N-Model Tests for VLSI Circuits," in *Proc. 40th IEEE Southeastern Symp. System Theory*, Mar. 2008, pp. 242–246.

[22] N. Yogi and V. D. Agrawal, "Sequential Circuit BIST Synthesis using Spectrum and Noise from ATPG Patterns," in *Proc. 17th IEEE North Atlantic Test Workshop*, May 2008, pp. 104–113.

[23] J. Zhang, M. L. Bushnell, and V. D. Agrawal, "On Random Pattern Generation with the Selfish Gene Algorithm for Testing Digital Sequential Circuits," in *Proc. International Test Conf.*, 2004, pp. 617–626.