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, *vagrawal@eng.auburn.edu*

Abstract

ATPG test patterns of a digital sequential circuit contain temporally and spatially ordered bits as well as random 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 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 of testing. Thus, all flip-flops are made scanable using a scan chain. The hardware is easy to implement and can often give high fault coverage. This primarily avoids the high BIST cost of generating non-random sequential patterns. However, there are several disadvantages. The area overhead of scanned circuits can be as high as 25%, as all flip-flops must be scanned. Inserting scan flip-flops in the critical paths can affect the timing and the speed of the circuit. Since the flip-flops need to be scanned to apply the test vectors and to observe the responses, the test times grow very rapidly with the number of flip-flops. Scan testing can cause unnecessary yield loss due to the non-functional nature of the tests. The scan chain cannot be operated at the rated clock to test for delay faults due to high test power dissipation. Hence other scan-based methods like launch-on-shift and launch-on-capture are used to cover them which have their additional challenges.

Non-scan testing avoids any modification to the circuit and relies on testing the circuit using only the primary inputs and outputs. The above mentioned disadvantages of scan-based testing are eliminated in this method and at-speed functional testing of the circuit becomes possible. 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 [16]. The fault coverage achieved can be low, as some faults are either not detectable in the sequential mode or no tests are found due to backtrack and CPU time limits of the ATPG program. The second problem is that generating vectors for sequential circuits in a BIST environment can be nontrivial because many faults in sequential circuits are random pattern resistant and BIST circuits to produce prespecified test sequences 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 characteristics of the already obtained ATPG vectors so that the new generated vectors have at least the same efficacy as the ATPG vectors. The problem can be generalized to include any pre-existing set of vectors instead of ATPG vectors. In this work we consider ATPG vectors due to their high quality in terms of fault detection.

For the problem at hand, instead of reproducing the exact ATPG sequence, we synthesize BIST to generate patterns with similar spectral characteristics. These BIST patterns are not the same as the ATPG patterns but excite somewhat similar behavior in the circuit. Sequences longer than the original ATPG sequences can even produce higher coverages. The contribution of this paper is a spectrum and noise analysis of ATPG patterns and a generalized BIST hardware synthesis procedure.

*This research is supported in part by the National Science Foundation Grant CNS-0708962.

We use the commercial sequential ATPG tool FlexTest [17] to generate vectors for the circuit under test (CUT) using the stuck-at fault model. The aim of the proposed BIST methodology is then to design hardware with minimum area overhead which recreates the essential spectral properties of ATPG vectors. The ATPG vectors are analyzed for their prominent spectral components using Hadamard transform. These prominent spectral components, which are generated by a Hadamard wave generator, are mixed in appropriate proportions and randomness if necessary is inserted in appropriate proportions. The major contribution of this work is a novel approach of spectral BIST which gives high fault coverage with a reduced hardware overhead as compared to existing published works. The proposed method although uses Hadamard transform for spectral analysis, it is adaptable for other transforms like Haar transform. We also propose a novel hardware approach for combining spectral components.

In this paper, Section 2 gives a brief description of previously published work in the area of BIST for sequential circuits. Section 3, for completeness, gives a brief description of how binary bit-streams can be analyzed in the spectral domain, which is explained in details in the literature [11, 34]. Section 4 describes the proposed spectral BIST method and in Section 5 we present the results obtained on eight ISCAS'89 benchmark circuits. We conclude in Section 7.

2. Prior Work

Several articles have been published on sequential BIST using random and weighted random patterns [3, 4, 20, 21, 31]. Some methods [20, 31] are based on modifying the flip-flops to increase the number of reachable states of the sequential circuit to enhance the fault coverage. In another approach [21], selected input patterns from a set of pseudo-random patterns generated by an LFSR, are held for several time units to give high fault coverages. Several methods [3, 4, 19, 29, 30] are based on weighted random patterns. Some authors [2, 15] generate weighted random patterns using counters, while others [6] use bit fixing. In [22, 23], a parameterized structure for test pattern generation using counters and comparators was proposed. Although the reported fault coverages was close to the deterministic patterns, the area overhead was high. Generally, the above mentioned methods suffer from disadvantages like high area or delay overheads, high complexity and inability to obtain consistently satisfactory results for a range of circuits.

Spectral analysis of sequential test patterns show that test sequences exhibit certain periodicities. Giani *et al.* [11] proposed a spectra based test generation scheme, where new vectors were generated replicating the enhanced spectral components of earlier vectors, which detected faults. For a BIST environment, they [12] proposed an extension of that method [11] using an in-built microprocessor to generate the vectors. Chen and Hsiao [7] modified that method [12] for only the hard to detect faults. Bushnell *et al.* [8] propose a Haar wavelet based BIST method for sequential circuits which gives fault coverages close to the deterministic patterns for several circuits.

3. Background

Our BIST synthesis is based on the premise that the spectrum of vectors that detect faults in a circuit reflect important characteristics of the circuit. These characteristics may include spatial and temporal correlations among the bits of primary input vectors and the necessary vector sequence length to sensitize paths between primary inputs and outputs of a sequential circuit. Along with the relevant spectra, some amount of noise or randomness is present, which corresponds to uncorrelated bits in the tests generated for some target faults. The noise-like behavior of these bits allows detection of untargeted faults that require similar, but not exact same, test sequences.

We use Walsh functions [27] to analyze the spectrum because they have been used for testing with effective results. These are a set of orthogonal functions consisting of trains of square pulses with +1s and -1s as the allowed states that can only change at fixed intervals of a unit time step. For order n, i.e., for a sequence of 2^n time steps, there are 2^n Walsh functions given by the rows of a $2^n \times 2^n$ Hadamard matrix $H(n)$ [26, 27, 28].

Hadamard matrices can be generated using the following recurrence relation:

$$H(n) = \begin{bmatrix} H(n-1) & H(n-1) \\ H(n-1) & -H(n-1) \end{bmatrix} \tag{1}$$

where $H(0) = 1$, and 2^n is the dimension of the nth order Hadamard matrix, $H(n)$. Any bit-stream of k bits can be represented as a linear combination of the Walsh functions obtained from the Hadamard matrix, $H(\log_2 k)$, where the multiplicand used for each function is the projection of the original bit-stream on that function. We shall refer to these multiplicands as components or coefficients. By analyzing the components we will be able to determine the major contributing Walsh functions in a given bit-stream.

4. Proposed Spectral BIST Method

We propose a novel BIST synthesis method using Hadamard transform and spectral techniques. One

$$
\begin{array}{c}1\\0\\1\\1\\1\\0\\1\\0\end{array}
\;\; 0 \rightarrow -1 \;\;
\begin{array}{r}1\\-1\\1\\1\\1\\-1\\1\\-1\end{array}
\quad \textit{Spectral decomposition}: \quad
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
1 & -1 & 1 & -1 & 1 & -1 & 1 & -1\\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1\\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1\\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1\\
1 & -1 & 1 & -1 & -1 & 1 & -1 & 1\\
1 & 1 & -1 & -1 & -1 & -1 & 1 & 1\\
1 & -1 & -1 & 1 & -1 & 1 & 1 & -1
\end{bmatrix}
\begin{bmatrix}1\\-1\\1\\1\\1\\-1\\1\\-1\end{bmatrix}
=
\begin{bmatrix}2\\\mathbf{6}\\-2\\2\\2\\-2\\-2\\2\end{bmatrix}
$$

bit stream — modified bit stream — Hadamard matrix $H(3)$ — modified bit stream — spectral component magnitudes

Figure 1: Spectral analysis of a stream of 8-bits. The prominent Walsh component in this bit-stream has magnitude 6 and is represented by the second row of Hadamard matrix, $H(3)$.

may use any commercial or home-grown sequential ATPG tool, FlexTest [17] 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 step two. The two steps are described below.

4.1 Spectral Analysis

As mentioned above, a spectral analysis is performed on the ATPG vectors using Hadamard transform [34]. The bit-streams entering various inputs of the CUT are analyzed separately. The 0s and 1s in a bit-stream are represented as -1s and +1s, respectively. To find the spectral components for a bit-stream, it is multiplied with the Hadamard matrix. Figure 1 shows an example of generation of spectral components using Walsh functions. In the example, an 8 bits bit-stream (0s and 1s represented as -1s and $+1$s) is analyzed by multiplying by a third order 8×8 Hadamard matrix. The corresponding result gives the spectral components.

The ATPG vectors are analyzed in sets, each of length N, where N is an appropriate dimension chosen for the Hadamard matrix. Later we shall discuss the selection of the value for N. The analysis of ATPG vectors can be performed either on discrete sets of vectors or the sets can be overlapped. Overlapping of sets helps in sampling the vectors at closer spaced intervals. The overlapping length (O_V) has to be chosen appropriately. Analyzing vectors with low overlapping may lead to loss of information, while a high overlapping may lead to sampling the noise in the spectrum. We choose a value of O_V equal to $2^N/4$ for optimum results. The spectral analysis of a set i of length N for a primary input j of the CUT provides an original component spectrum denoted by C_{ij} and a power spectrum (which is the square of C_{ij}) denoted by P_{ij}. After the spectral analysis of all the sets, all 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. The threshold value T_H like the overlapping length O_V needs to be chosen appropriately. 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, while the averaged component spectrum C_j gives the sign or phase of those components. From the P_j spectrum we choose the top M prominent components for each input j based on their magnitude. Their sign is determined by the C_j spectrum. For our experiment, we chose a value of $M = 4$. The M prominent components along with their power magnitudes and signs are then used for BIST implementation.

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. This is implemented using a spectral component synthesizer. Randomness or noise

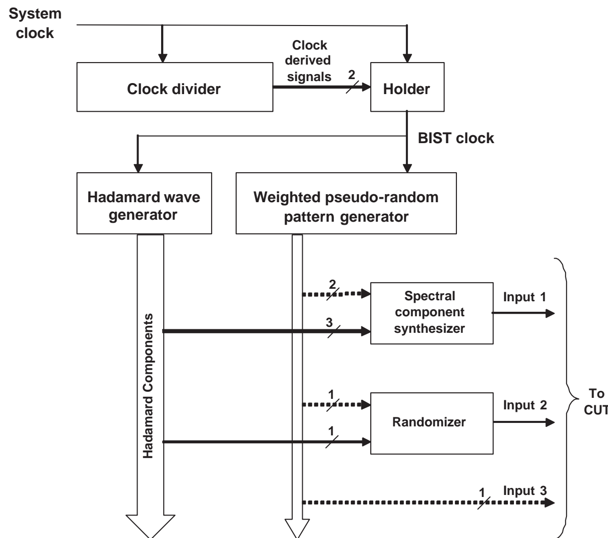

Figure 2: Proposed spectral BIST architecture.

if required is inserted in appropriate amounts to the generated vectors. This is achieved using the randomizer circuit.

Figure 2 shows the proposed spectral BIST architecture, which consists of: Hadamard wave generator, Spectral component synthesizer, Randomizer, Weighted pseudo-random bit-stream generator, Holder circuit and Clock divider circuit. Figure 2 is a BIST circuit designed for a CUT with three inputs. The Hadamard wave generator generates the spectral components which are combined by the component synthesizer (generally, one per PI) using random bit-streams provided by the weighted pseudo-random bit-stream generator. The spectral component signals are shown in dark bold lines while the weighted random signals are in dotted lines. Noise if required is inserted in the combined spectral components by 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

The heart of the proposed BIST hardware is the Hadamard wave generator, which generates the Walsh functions or Hadamard spectral components for the spectral component synthesizer. There is much literature on Walsh function generators [9, 5, 10, 14, 37]. Many implementations are based on an arithmetic or a Gray code counter and combinational logic. The speed of operation of such designs is restricted by the speed of the counter. Other fast implementations exist, but they require extra hardware. We selected a Walsh function generator proposed in [14], which uses an arithmetic counter and has low hardware overhead. The generator of order N, which generates 2^N Walsh functions, requires N flip-flops and $2^N - N - 1$ XOR gates. Figure 5 shows an example of a Walsh function generator of order 4, which is taken from [37].

The order of the Hadamard matrix N, used for spectral analysis, is also used to implement the Hadamard wave generator. Since higher order Hadamard matrices are constructed from lower order matrices [27, 28], lower order matrices form a subset of the higher order matrices. Thus higher order matrices are able to characterize a given bit-stream better than lower order matrices. However the area overhead for implementing the Hadamard wave generator increases exponentially with the order of the Hadamard matrix as mentioned earlier. Hence we have a tradeoff between the resolution of spectral analysis and the area overhead of the implemented hardware. For our implementation, for the value of N, we chose a lower bound of 4 and an upper bound of 8. These bounds were chosen so that they would provide the optimum tradeoff mentioned above. 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

As described earlier, M prominent spectral components are combined in required proportions and phases. The proportions are determined from the power magnitude of the spectral components (from spectrum P) and phases from their signs (from spectrum C), both of which are obtained from spectral analysis of Section 4.1. We combine the spectral components in a multiplexer called “spectral component synthesizer”, as shown in Figure 3. The inputs to the multiplexer are the chosen spectral components, which are generated by a Hadamard wave generator, and its select line is driven by a weighted random bit-stream. The weighting of the bit-stream is determined by the proportion in which the components are to be combined. Figure 3 depicts an example of the proposed structure which combines three spectral components SC1, SC2 and SC3 in proportions of 0.25, 0.25 and 0.5, respectively.

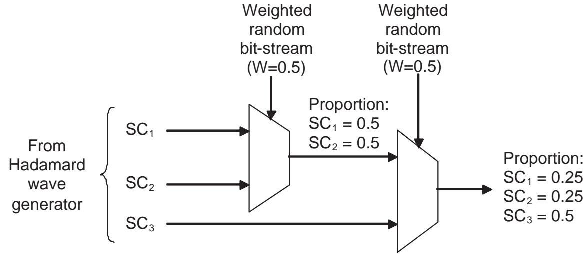

Figure 3: Spectral component synthesizer that combines three spectral components.

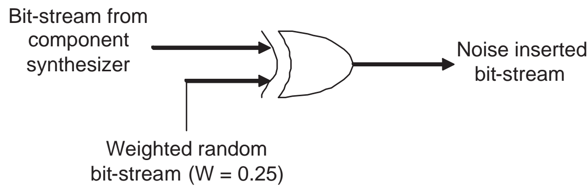

Figure 4: Randomizer XOR gate that randomly flips 25% of bits.

4.2.3 Randomizer

Along with the prominent spectral components, some randomness or noise is present in the ATPG vectors which need to be inserted in the regenerated vectors. Due to the filtering effect of the threshold T_H, for some of the inputs, the number of selected prominent components (M_S) could be less than M. For inputs with $M_S > 1$, it is observed that some amount of noise is inherently inserted by the spectral component synthesizer due to its use of a weighted pseudo-random bit-stream. However for inputs with $M_S = 1$, noise needs to be inserted explicitly. For such inputs the level of noise or perturbations to the prominent spectral component is generally very low although important and spread out in the long sequences of ATPG vectors which are not picked up by the short sequence spectral analysis. To estimate the amount of noise in such sequences, we analyze their runs of 0s and 1s. We pick the top 95% of the run lengths to eliminate the noisy effects of the short run-lengths. The reciprocal of the average of the selected run-lengths then gives the average amount of randomness or perturbation to be inserted in 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. Figure 4 shows an example of flipping 25% of the bits randomly by using a randomizer XOR gate.

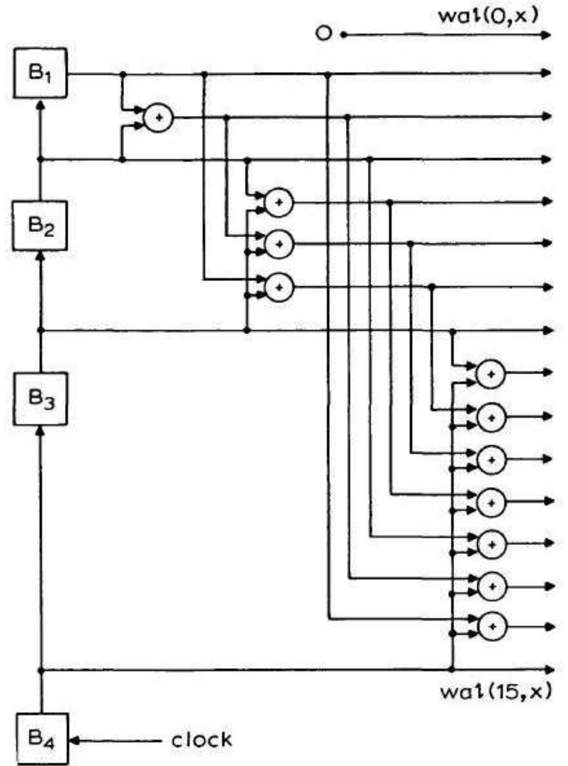

Figure 5: Walsh function generator of order 4 that generates 16 Walsh functions [37].

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. We use a 16-bit cellular automata register to generate the pseudo-random bit-streams. Weighted bit-streams are obtained using a combination of AND and OR gates as required. The different weights to be generated are determined by the proportions in which the spectral components are to be mixed and by the amount of randomness to be added for all inputs. We quantize the required weights into $W = 2^w$ fixed weights that take the form of $i \times 2^{-w}$ for $i = 0$ to $2^w - 1$. For reported experiments, we used a value of $w = 4$. We 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 our fault coverage, based upon its reported benefits [13, 18, 21, 38]. This method involves holding input vectors constant for several clock cycles while applying the system clock to the circuit under test. It was believed in [21] that holding

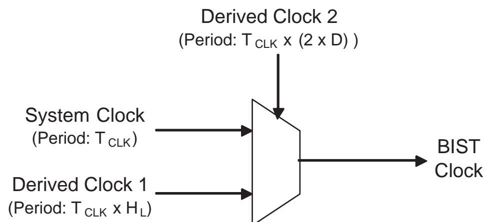

Figure 6: Holder circuit implemented using a multiplexer and clock derived signals.

proves effective due to several reasons. Holding a vector for several clock cycles helps fault effects latched in flip-flops to be observed at the primary outputs. Also if a hard-to-detect fault is activated by a random vector, then holding the vector for multiple cycles will activate the fault multiple times, thus increasing the probability of its detection. The number of clock cycles to hold the input vectors can be determined in different ways. In [21], the number of clock cycles to hold the input vectors is determined by using a deterministic sequential test pattern generator to detect the flip-flop output faults. In [13], the number of hold cycles is determined by logic simulation of the fault-free circuit with vectors having different hold cycles and determining their ability to set flip-flops to 0s and 1s and to traverse most states. It has been reported [18, 38] that "Holding a vector V_b for a testable stuck-at fault f_B in combinational circuit C_B d+1 times, where d is the sequential depth of the corresponding acyclic sequential circuit C created by adding flip-flops at any wire of C_B, gives a test sequence for all sequential faults in C corresponding to f_B."

In the proposed method, we determine the number of hold cycles from an upper bound on the sequential depth of the circuit [25], which we shall denote as H_L. The value of H_L is rounded to the nearest power of 2. The upper bound of the sequential depth [25] is defined as the minimum number of cycles required to initialize a flip-flop (for both cases of 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 we generate D/H_L vectors, each of which is held for H_L clock cycles. Although any appropriate value can be chosen for D, an effective value for D was found to be proportional to the length of the ATPG vectors. We choose the block length D equal to *length* (ATPG Vector Length/50) rounded to the nearest power of 2. Figure 6 shows the implementation of the holder circuit. It consists of a multiplexer whose inputs are clock signals with periods CLK and $CLK \times H_L$. The control signal to the multiplexer is driven by a clock signal of period

Table 1: FlexTest ATPG results.

Circuit	FlexTest ATPG result			
	No. of vectors	Total no. of faults	# faults detected	Fault cov. (%)
s298	153	308	273	88.64
s820	1127	850	793	93.29
s1423	3882	1515	1443	95.25
s1488	736	1486	1446	97.31
s5378	739	4603	3547	77.06
s9234	15528	6927	1588	22.92
s15850	61687	13863	7323	52.82
s38417	55110	31180	15472	49.62

$CLK \times (2 \times D)$. The output of the holder circuit is the BIST clock which drives the Hadamard wave generator and the weighted pseudo-random bit-stream generator blocks. All the input signals of the holder circuit are derived from the clock and are provided by the clock divider circuit.

4.2.6 Clock Divider Circuit

The clock divider circuit generates two clock derived signals by dividing the clock frequency by H_L and $(2 \times D)$, respectively, which are then provided to the holder multiplexer to generate the BIST clock. The clock divider circuit is constructed from an asynchronous binary counter consisting of $\lceil \log_2(2 \times D) \rceil$ flip-flops. If the BIST environment uses a pattern counter (which is found in many cases) then the clock derived signals can be conveniently obtained from appropriate outputs of the pattern counter.

5 Results

We emulated our BIST method using a MATLAB program such that the generated vectors would be the same as or very close to those generated by actual hardware. These vectors include the quantization errors involved in the BIST test generation method which would not be included in the software based methods like [12]. Thus software based methods would give results which are more optimistic than hardware based methods. We implemented BIST on eight ISCAS'89 benchmark circuits. We inserted a reset signal in the circuits to initialize the flip-flops. We activate it only once before beginning the BIST session. The initial reset can be performed at a slow speed and hence the reset signal can be implemented using minimum resources. ATPG vectors with an initial reset were generated to detect stuck-at faults in the circuits using Mentor Graphics

Table 2: **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 [8]
			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

commercial tool FlexTest [17]. Table 1 gives the results of this ATPG. These ATPG vectors were analyzed for prominent spectral components and noise as described in Section 4. Using this information in the MATLAB program, 64,000 BIST emulated test vectors were generated and then fault simulated again using FlexTest.

Table 2 gives the results for the number of faults detected by our method ('Hadamard BIST') for each circuit and they can be compared with results from random and weighted random vectors (both without and with holding), ATPG vectors and with the method in [8]. For random, weighted random and the proposed method we generated 64,000 vectors. For the weighted random vectors, weights for each circuit were used from their corresponding ATPG vectors without quantization. Random and weighted random vectors were generated using a software random number generator. For implementing holding for the random and weighted random vectors, we used the same holding scheme as used in our proposed method.

As shown in Table 2, the proposed BIST method detects equal or greater number of faults in six out of eight circuits than the existing methods of random, weighted random and [8]. Also in five out of eight circuits, our proposed method was able to detect at least as many faults as detected by ATPG vectors using 64,000 BIST vectors. Noticeable results were obtained for s38417, where our 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 by the method.

Table 3 shows the effectiveness of our proposed BIST method over longer number of vectors and by comparing its results with those obtained from ATPG vectors. 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 close to that of ATPG vectors.

Table 4 shows a comparison of the area overhead in terms of number of transistors and % area overhead of our proposed method with the method proposed in [8]. 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 of a clock divider circuit. The area overhead of our method, in the case where a clock divider circuit is used, is lower than [8] in three out of eight circuits. In the case where it is not used, our area overhead is lower than [8] in six out of eight circuits. On an average, the area overhead of our proposed method with clock divider circuit is 9.31%, without clock divider circuit is 8.67% and that of the method proposed in [8] is 8.42%. The cost of our slightly higher area overhead is compesated by the obtained higher fault coverages. Further research work is required in optimizing and reducing the area overhead.

6 Improving Testability and Identifying Untestability

As observed in table 1, ATPG gives low fault coverage for many sequential circuits. The undetected faults

Table 3: **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

can be classified into two groups. Some faults are truly untestable and they can be safely removed from the fault list without affecting the defect level [1, 24]. Some other undetected faults are aborted by the test generation program due to limited CPU and memory resources. Such faults can be helped to be detected by improving their testability using DFT techniques. We shall attempt to reason the undetectability of faults either by showing they are sequentially untestable or help them getting detected by improving their testability.

6.1 Improving testability

Several techniques can be used to improve the testability of sequential circuits. In [35] a DFT technique was introduced to improve the observability of so called RTL faults. RTL faults are defined as the faults on the boundary of the combinational logic in the sequential circuit. We shall use this method to improve the testability of the sequential circuits. In this method using a testability analysis, all the unobservable RTL faults are determined and they are connected using an XOR chain. The output of this XOR chain is added as a primary output of the circuit.

6.2 Identifying untestability

An undetected fault if proved untestable may be ignored in the test generation process. One emphasizes [1, 24] the importance of identifying the sequentially untestable faults to lower the test generation effort and the yield loss occurred by testing them using other techniques as scan. A single-fault theorem [1] allows identification of sequentially untestable faults by a combinational ATPG procedure. This procedure defines $C(n)$, a circuit consisting of n copies of the combinational logic of the sequential circuit connected as

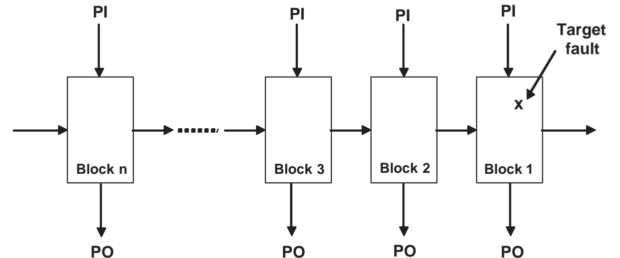

Figure 7: Combinational circuit $C(n)$ with a target single fault [1].

a linear array from right to left. Each copy represents a time frame of the sequential circuit. The rightmost copy is named as block 1 while the leftmost copy as block n. All single stuck faults in block 1 are targeted by a combinational ATPG. Figure 7 shows an example circuit $C(n)$ with a target stuck fault. The theorem states that "a target single-fault that is untestable in $C(n)$ is also untestable in the sequential circuit."

6.3 Example circuit

Consider the circuit s5378. ATPG detected 3547 out of 4603 faults using 739 vectors. To improve the testability we analyzed the unobservable RTL faults [35]. Outputs of 49 flip-flops from a total of 179 were made observable using an XOR chain. After DFT, the same 739 ATPG vectors detected 3785 faults. Spectral vectors which initially detected 3603 faults, detected 3739 faults in the DFT inserted circuit. In [1], 781 faults are reported as sequentially untestable. By inserting the DFT it can be observed from figure 7 that the only addition to each block of $C(n)$ is an XOR chain whose inputs come from the left side of each block and the output is a primary output. Since the untestability analy-

Table 4: **BIST area overhead in transistors.**

Circuit	No. of transistors in circuit	Hadamard BIST [this paper]				Haar BIST [8]	
		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

sis method inserts faults only in block 1 the minimum number of faults which can be conclusively labeled as untestable are $781 - 49 \times 2 = 683$. Table 5 gives the fault coverage and test coverage results for ATPG and spectral vectors with and without DFT. Test coverage is calculated as the ratio of the number of faults detected to the total number of detectable faults.

Table 5: Fault coverage and test coverage results with and without DFT for s5378.

Test method	Fault Coverage (%)		Test Coverage (%)	
	Without DFT	With DFT	Without DFT	With DFT
ATPG	77.05	82.22	92.80	96.55
BIST	78.27	81.23	94.27	95.38

7 Conclusion

We present a novel hardware test generation method for sequential circuit BIST. ATPG vectors are analyzed using Hadamard transform to obtain the spectrum and random noise level. The ATPG vectors can be derived from gate, register-transfer, or function level algorithm [32, 33, 34]. Target faults for these tests can be stuck-at or delay [35] faults, or even several fault models can be combined [36]. 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" that has not been reported before. An XOR circuit inserts noise in the generated bit-streams. Results show fault coverages equal or greater than those of ATPG vectors in six out of eight circuits. Area overheads are moderate compared to existing methods. Our method also achieved a maximum fault coverage in six out of eight circuits. Although we use Hadamard transform any other compatible transform of binary bit-streams, such as Haar transform, can be used in this methodology. We also use untestability analysis for DFT-based BIST coverage improvements.

References

[1] V. D. Agrawal and S. T. Chakradhar, "Combinational ATPG Theorems for Identifying Untestable Faults in Sequential Circuits," *IEEE Trans. Computer-Aided Design*, vol. 14, no. 9, pp. 1155–1160, Sept. 1995.

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

[3] M. F. Ashaibi and C. R. Kime, "Fixed-Biased Pseudo-random Built-In Self-Test for Random Pattern Resistant Circuits," in *Proc. International Test Conf.*, 1994, pp. 929–938.

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

[5] P. W. Besslich, "Walsh Function Generators for Minimum Orthogonality Error," *IEEE Transactions on Electromagnetic Compatibility*, vol. EMC-15, no. 4, pp. 177–180, Nov. 1973.

[6] M. Chatterjee and D. K. Pradhan, "A Novel Pattern Generator for Near-Perfect Fault-Coverage," in *Proc. 13th IEEE VLSI Test Symp.*, 1995, pp. 417–425.

[7] 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.

[8] 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.

[9] B. J. Falkowski and T. Sasao, "Unified Algorithm to Generate Walsh Functions in Four Different Orderings and Its Programmable Hardware Implementations," *IEE Proceedings Vision, Image and Signal Processing*, vol. 152, no. 6, pp. 819–826, Dec. 2005.

[10] 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.

[11] A. Giani, S. Sheng, M. S. Hsiao, and V. D. Agrawal, "Efficient Spectral Techniques for Sequential ATPG," in *Proc. IEEE Design Automation and Test in Europe Conf.*, 2001, pp. 204–208.

[12] 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.

[13] 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.

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

[15] D. Kagaris and S. Tragoudas, "Generating Deterministic Unordered Test Patterns with Counters," in *Proc. IEEE VLSI Test Symp.*, 1996, pp. 374–379.

[16] 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.

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

[18] 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.

[19] F. Muradali, V. K. Agarwal, and B. Nadeau-Dostie, "A New Procedure for Weighted Random Built-In Self-Test," in *Proc. International Test Conf.*, 1990, pp. 660–669.

[20] F. Muradali, T. Nishida, and T. Shimizu, "A Structure and Technique for Pseudorandom-Based Testing of Sequential Circuits," *J. Electronic Testing: Theory and Applications*, vol. 6, no. 1, pp. 107–115, 1995.

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

[22] I. Pomeranz and S. M. Reddy, "Built-In Test Generation for Synchronous Sequential Circuits," in *Proc. Intl. Conf. on Computer-Aided Design*, 1997, pp. 421–426.

[23] 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.

[24] J. Rearick, "Too Much Delay Fault Coverage is a Bad Thing," in *Proc. International Test Conf.*, 2001, pp. 624–633.

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

[26] A. R. Thompson, J. M. Moran, and G. W. Swenson Jr., *Interferometry and Synthesis in Radio Astronomy*. New York: Wiley, 1986.

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

[28] S. Wolfram, *A New Kind of Science*. Champaign, IL: Wolfram Media, 2002.

[29] H. Wunderlich, "Self Test Using Unequiprobable Random Patterns," in *Proc. Fault-Tolerant Computing Symp.*, 1988, pp. 258–263.

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

[31] H.-J. Wunderlich, "The Design of Random-Testable Sequential Circuits," *Proc. 19th Fault-Tolerant Computing Symp.*, pp. 110–117, 1989.

[32] N. Yogi and V. D. Agrawal, "High-Level Test Generation for Gate-Level Fault Coverage," in *Proc. 15th IEEE North Atlantic Test Workshop*, May 2006, pp. 65–74.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] C. K. Yuen, "New Walsh-Function Generator," *Electronics Letters*, vol. 7, p. 605, 1971.

[38] 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.