Characteristic Faults and Spectral Information for Logic BIST

Xiaoding Chen and Michael S. Hsiao {*xiaoding, hsiao*}*@vt.edu*
The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA

Abstract

We present a new method of built-in-self-test (BIST) for sequential circuits and system-on-a-chip (SOC) using characteristic faults and circuit-specific spectral information in the form of one or more Hadamard coefficients. The Hadamard coefficients are extracted from the test sequences for a small set of characteristic faults of the circuit. By extracting a few characteristic faults from the circuit, we show that detection of these characteristic faults is sufficient in detecting a vast majority of the remaining faults in the circuit. The small number of characteristic faults allows us to reduce the coefficients necessary for BIST. State relaxation is performed on the compacted test sequences to reduce the spectral noise further. Since we are targeting only a very small number of characteristic faults, the execution times for computing the spectra are greatly reduced. Our experimental results show that our new method can achieve high BIST coverage with both lower computational efforts and storage, with very few characteristic faults.

1 Introduction

Built-in-self-test (BIST) is playing an increasingly important role in VLSI testing. As the operating speed in modern VLSI technology increases, it becomes more difficult for test equipments to keep pace with the growing speed. Furthermore, BIST is a promising solution for testing embedded cores in the SOC environment, whose testabilities are greatly reduced due to limited accessibility.

Many BIST schemes have been proposed in literature. Pseudo-random test pattern generator (TPG) in the form of a linear-feedback-shift-register (LFSR) has been widely used. Weighted pseudo random TPG [3, 4] takes into consideration the subset of random-pattern-resistant faults to yield better results than random methods [3, 4, 5, 6]. The basic idea for weighted pseudo random is to bias the probability at each input based on the information gathered on the circuit. The weights can be obtained by using counter-based schemes [7], performing bit-fixing [8], or employing LFSR with good starting seed and feedback polynomial [9, 10, 11]. Many hardware pattern generators [6] often round-off optimal weights, hence producing patterns that are sub-optimal for certain circuits at a much lower computational cost.

While (weighted) pseudo-random BIST architecture may not provide a satisfactory fault coverage in reasonable pattern length (especially for sequential circuit), deterministic BIST techniques attempt to solve this problem by storing deterministically-generated test patterns. However, the cost associated with the storage of the pre-calculated test patterns may be high if there is a large number of patterns.

An embedded controller or processor on an SOC makes it possible for the processing unit to generate patterns to test the rest of the chip [12, 13, 14, 15, 16]. This alleviates the need for additional hardware such as LFSRs to test the other embedded cores/peripherals, and by employing efficient algorithms, it is also possible to reduce storage.

In [17], temporal correlation was found to exist within test sets, and this information allows for the application of spectral techniques to ATPG [18]. In [16], spectrum-based BIST technique was proposed for SOCs with embedded processor. It generates vectors by using the spectral characteristic of each embedded core. To do this, it views each core as a digital system that is described by input-output system, described as functions of time. Any such function can be represented and reproduced in the time-domain, using its frequency-domain spectrum. The approach uses Hadamard transform to compute the spectral information of digital circuits. As noise-free signals are desired to capture the spectral characteristic of the signal, static test set compaction is employed to remove any unnecessary vectors and thus filter some of the unwanted noise. As a result, a test sequence can be described by the corresponding spectral coefficients, and it preserves the fault detection criterion. At this point, self-testing becomes the problem of determining the spectrum of the circuit under test (CUT).

In this paper, we address the problem of refining the spectrum of the CUT, by picking a small set of *"characteristic faults"* from the circuit and employing state relaxation. Unlike [16], where spectral information is extracted from the entire fault set, only a very small number of the faults is used in this work. In essence, the set of characteristic faults can be viewed as representative faults of the entire circuit. In [19], test cubes are computed for hard faults such that many other easy faults can be detected simultaneously; however, only combinational circuits are considered. State relaxation [20] is performed on the compacted test sequences of the characteristic faults to further reduce noise. We present efficient algorithms on selecting characteristic faults.

The remainder of the paper is organized as follows. Section 2 gives an overview and motivation for the new spectral BIST. Section 3 explains the algorithm of computing characteristic faults and its application to BIST. Section 4 discusses the details of state relaxation. Section 5 reports the experimental results, and Section 6 concludes the paper.

2 Overview and Motivation

Previously, spectrum-based BIST [16] has shown the important role spectral information can play. The spectral information it uses is extracted from the compacted test set for all detected faults. Because wider spectra contain more noise, pruning the spectra to rid the noise will help. Generally, sequences that detect hard faults frequently can detect many other faults. Based on this observation, we propose to focus on obtaining the spectral information for only the more difficult faults.

Our work begins with the test set that detects only the k latest detected faults. We perform relaxation on this test set to relax any unnecessary input bits to don't-care values (X's); the relaxed test set still ensures detection of the k faults. The relaxed sequence essentially removes additional noise from the original sequence, and the non-relaxed input bits in the relaxed sequence form the basic spectrum and are sufficient to traverse the state space such that the k hard faults are still detected. Surprisingly, many of the input bits can be relaxed. The states traversed may not be fully specified due to the don't cares in the input sequence. Furthermore, if detection of these k faults also detects a vast majority of the remaining faults, the derived spectrum can be viewed as the representative spectrum for the entire circuit. Following this assumption, our goal is to extract the spectrum from a *very small fault set*, and this spectrum is able to represent the characteristic of the circuit. As in [16], we employ signal processing

*This work is supported in part by NSF Grant CCR-0196470 and a grant from NJ Commission on Science & Technology.

0-7803-7607-2/02/$17.00 ©2002 IEEE

techniques to extract the spectral information and generate spectral BIST vectors. In addition, in our scheme, state relaxation is performed on the compacted test set for the k faults to reduce noise prior to extraction of spectral information.

The overall framework for extracting the spectral characteristics is presented in [16], however, characteristic faults were not identified. In this paper, by starting with random vectors, we first *filter* the random test set using static compaction for the k latest detected faults (not the whole detected fault set). Then, state relaxation is applied to the compacted test sequence to further reduce any additional embedded noise. The spectral coefficients are extracted from this relaxed, compacted test sequence through Hadamard transform, where the predominant Hadamard components are identified. New vectors are added to the test set based on the extracted spectrum. This process iterates until the stopping criterion is reached. At the end of this process, only the spectral characteristics for the k faults in the iteration where highest fault coverage was reached are collected and stored, and those k faults are called *characteristic faults*. The final spectrum is used to generate BIST vectors in the SOC. Note that in our process, the spectral coefficients from one iteration will be used by BIST. This is different from [16], where the coefficients from an iteration are appended to the final set if there is an increase in fault coverage.

3 Computation of Characteristic Faults

Instead of targeting the entire fault list to extract spectral information, we want to find "characteristic" faults within the fault set. Characteristic faults should have the property that a test set which detects them is also able to detect many other faults in the circuit. For this reason, the spectrum corresponding to the characteristic fault set is a representative spectrum for all the faults.

Because sequences that detect hard faults generally detects many other faults as well, our aim is on identifying the set of hard faults such that detecting them will maximize detection of the rest of the faults in the circuit. To do so, we modify the spectral ATPG process [16] in which the corresponding target faults in each iteration are associated with the derived spectrum. For instance, when the spectrum is extracted for the k last-detected faults in iteration i, the new vectors to be generated are simulated and the set of faults, S_i detected by these new vectors are recorded. The set S_i indicates the set of faults that have similar spectral characteristics as the k representative faults. In the next iteration, we would likewise compute S_{i+1}. The k faults in the jth iteration corresponding to the largest fault set S_j is chosen to be the characteristic fault set.

The characteristic-fault extraction algorithm is outlined below:

T_0 = random test vectors;
$i = done = S_{max} = 0$;
while (not $done$) and (i<max_iteration)
    fault simulate T_i;
    K_i = k last-detected faults;
    $C_i = T_i$ filtered for K_i;
    R_i = C_i relaxed;
    perform Hadamard Transform on R_i
        to obtain spectral coefficients H_i;
    E_i = new vectors generated using coeff. H_i;
    S_i = faults detected by E_i;
    if $S_i > S_{max}$
        characteristic fault set = K_i;
        $S_{max} = S_i$;
    i++;
    if no improvements in 3 consecutive iterations
        done = 1;

The above algorithm is simple to follow and we will show in experiments that it is also efficient in getting the characteristic fault set that achieves high fault coverages.

4 State Relaxation

4.1 Review of Support Sets

A test vector is *fully specified* if all inputs are specified to 0 or 1 (*i.e.,* no input assumes a value of X). Test vectors that contain X's are said to be *partially specified.* A *support set* (SS) for a primary output Z is any set of signals (including primary inputs) that satisfy all the following conditions:

1. All signals in the set assume a logic value 0 or 1.
2. The primary output Z is a member of the set.
3. The logic value on any signal (except PIs) in the support set is uniquely determined by values of other signals in the SS.

In the case of multiple primary outputs, condition 2 is modified to require that each of the POs be included in the support set. Likewise, support sets can also be computed for sequential circuits, where condition 2 is modified to include any next state variable v that satisfies the following condition:

- Next state variable v that is at a logic value 0 or 1.

The *support signals* for a gate are the smallest subset of required signal that uniquely determine the current logic value of the gate. For example, consider an AND gate a that has two inputs b and c. Suppose a=1, then both b, c must be 1 and be support signals for a. If a=0, b=1, and c =0, then support signal is simply b since the value on a is uniquely determined by signal b. In case that a=0, b=0, c=0, b, c are at level v(b) and v(c) respectively, and v(b)>v(c), we will use the following criteria:

- If one of the possible support signals of the gate has already been included in the support set of the circuit, then this signal is selected as the support signal of the gate;
- Otherwise, we choose a support signal at the lowest level;

In this case, if b is already in the support set of the circuit, then b will be the only support signal for a; otherwise, c will be the support signal of a because c is at a lower level.

A support set is *irredundant* if no signal in the set can be deleted without violating conditions 2 or 3. A *minimum* support set has the least cardinality among all possible support sets. It is desirable to compute a support set for PIs with small cardinality as this leads to a *cleaner* signal for later processing. However, attempting to compute the minimum support set for each input vector is computationally expensive.

By computing the support set for each time-frame of sequential circuit, both the intermediate state and input sequence can go from fully specified to partially specified. The procedure below [20] efficiently computes an irredundant support set, and takes a list of gates L with known logic values.

Procedure COMPUTE_SUPPORT_SET(L)
    support set $S = L$;
    while ($\exists$ unsupported gates in L)
        g = unsupported gate in L with maximal level;
        ss = minimal support signals for g;
        add support signals ss to S;
        for all unsupported gates i in ss
            add i to L;
        mark g supported;
    return support set S;

In the above procedure, we assume that the circuit is levelized and the input vectors have been simulated to determine the value of each gate in

the circuit. The support sets are, then, computed based on the logic values for each corresponding vector.

4.2 Relaxation in Sequential Circuit

For sequential circuits, we perform state relaxation in a reversed order, that is, starting from the last pattern. Let's consider benchmark circuit s27 [1] with gate 10 stuck at 0. The original compacted test set $S1$ for detecting this fault is {1011, 1001, 0101, 0111, 0011, 0110}, where the fault is detected by the last vector.

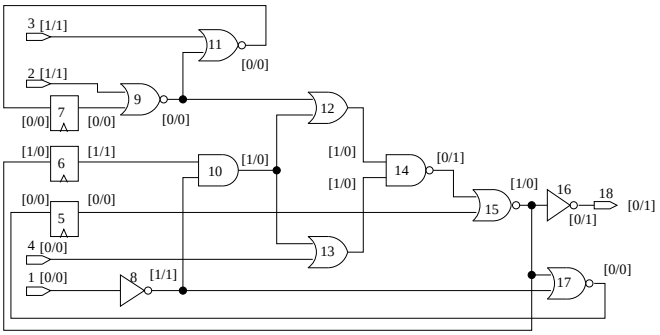

Figure 1: Good/Faulty Circuit Values at Time Frame 5

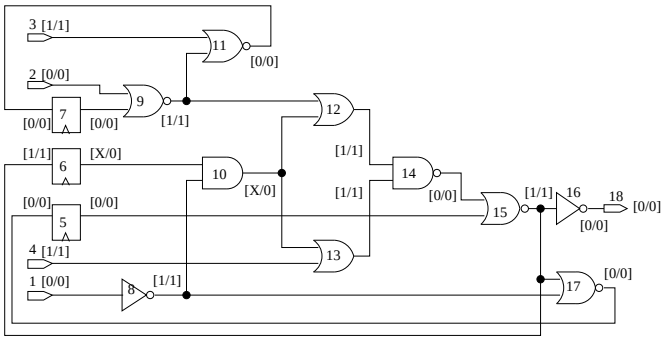

Figure 2: Good/Faulty Circuit Values at Time Frame 4

Gate values for both good and faulty circuits are shown in Figures 1 and 2 respectively for vectors 5 and 4. The values are shown in the form of [good/faulty]. Starting from pattern 5, the constraint list L contains only the PO which differentiates good circuit and faulty circuit; since pattern #6 (time frame 5) is the last vector, no next state FF is included in L. In this particular case, the fault is detected by pattern 5 on output gate 18, with good circuit of logic 0 and faulty circuit of logic 1. Thus, L for this vector is simply {18}. Calling procedure COMPUTE_SUPPORT_SET() with this constraint list L, we obtain the support set for good and faulty circuit respectively, listed below in the levelized fashion. The value "12[1/0]" indicates that gate # 12 = logic 1 is included as part of support set for the fault-free circuit, while gate 12 with logic 0 is included in the support set of the faulty circuit.

```
Support Set for Good and Faulty Circuits, vec #5
0  5[0/0] 6[1/1] 1[0/0] 4[X/0] 2[X/1]
1  8[1/1] 9[X/0]
2 10[1/1]
3 12[1/0] 13[1/0]
4 14[0/1]
5 15[1/0]
6 16[0/1]
7 18[0/1]
```

We combine the support sets for good and faulty circuits at level 0 and split that into two lists $L1$ and $L2$ by gate type. $L1$ consists only PIs and $L2$ contains only FFs. PIs and FFs which are not contained in either $L1$ or $L2$ can be set to X. By doing this, we have relaxed the input and state for the current time frame. In this example, $L1$ is {1 4 2} and $L2$ is {5 6}. $L2$ will be used to form the constraint list for the preceding pattern. We move on to relax pattern 4 the same way by calling the procedure, and this time, constraint list L will be the list of $L2$, which is {5 6}. Computing the support set the same way and we get the following support sets of pattern 4 for good and faulty circuit respectively:

```
Support Set for Good and Faulty Circuits, vec #4
0  5[0/0] 4[1/1] 7[0/0] 2[0/0]
1  9[1/1]
2 12[1/1] 13[1/1]
3 14[0/0]
4 15[1/1]
5 17[0/0]
```

The above procedure is repeated until we've reached the first vector. in reversed order. Shown below are the relaxed inputs (from input 1 to 4 in order) and intermediate states(from FF 5 to 7 in order) for our s27 example:

Vec #	Orig Input	Relaxed Input	Relaxed State
0	1011	XXXX	XXX
1	1001	XXXX	XXX
2	0101	XXXX	XXX
3	0111	0X1X	XXX
4	0011	X0X1	0X0
5	0110	01X0	010

5 Experimental Results

We conducted experiments of our approach on both ISCAS89 [1] and ITC99 [2] benchmark circuits. After computing the characteristic faults for each circuit, we allowed for a maximum of 70,000 vectors from the spectral information extracted for the characteristic faults after state relaxation. When performing the Hadamard transform, a relaxed '*X*' does not contribute to the spectrum extraction. Recall that in [16], the final coefficient set stored into the SOC is a concatenation of coefficient sets from different iterations. Once a new fault is detected in the iteration, the coefficient extracted from the whole compacted test set of that iteration will be included into the final coefficient set. That may result in larger storage requirement. We will show later in Table 3 how the two schemes differ in both computational effort and storage requirements. For all circuits, the number of iterations is set to 20.

Table 1 compares the results among STRATEGATE [21], weighted random BIST, spectral method [16] and our technique. Note that **only one** characteristic fault is used for our technique. In this table, the total number of faults is first listed for each circuit, followed by the coverage achieved by STRATEGATE. Next, BIST coverages are reported for ideal weighted random, rounded-off weighted random, [16], and finally our approach based on characteristic faults. For instance, in circuit s5378, STRATEGATE generated a test set that detected 3639 faults, ideal weighted random BIST detected 3127 faults, rounded-off weighted random BIST detected 3083 faults, 3596 faults were detected by [16], and we detected 3611 faults with spectrum for **only one** characteristic fault! Likewise, in circuit b12, our technique detected 1648 faults with spectral information using only one characteristic fault, while STRATEGATE, ideal, rounded-off weighted random BIST, and [16] detected 1488, 663, 636 and 1621 faults, respectively.

Table 1: Comparison of Fault Coverages

Ckt	Total Faults	# Faults Detected by STRATEGATE [21]	# Faults Detected by Non-Scan BIST			
			Weighted-Random Patterns		Spectral Patterns	
			Ideal Weights	Rounded-off Weights	[16]	Ours
s382	399	**364**	329	116	**364**	357
s400	428	**384**	306	106	**384**	376
s526	555	**454**	95	94	**454**	442
s713	581	**476**	476	476	**476**	**476**
s1196	1242	**1239**	1233	1228	*1237*	1227
s1238	1355	**1282**	1276	1270	**1282**	1266
s1423	1515	**1414**	1319	1167	**1414**	**1414**
s1488	1486	**1444**	1442	1410	**1444**	**1444**
s1494	1506	**1453**	1451	1418	**1453**	**1453**
s5378	4603	**3639**	3127	3083	3596	*3611*
b01	135	**133**	**133**	**133**	**133**	**133**
b04	1346	**1168**	**1168**	**1168**	**1168**	**1168**
b08	489	**463**	461	438	**463**	**463**
b11	1089	1003	937	898	**1004**	**1004**
b12	3102	1488	663	636	1621	**1648**

Only **one** characteristic fault is used under "Ours" column

From this table, we can see that with only one characteristic fault, our fault coverages are very close to those obtained by state-of-the-art sequential ATPG for most circuits. In all circuits except for s1196 and s1238, the results of our technique either surpass or equal the results obtained for the ideal weighted random technique. For some hard-to-test circuits such as s5378 and b12, our technique is able to detect significantly more faults than the weighted random BIST approach and also outperforms [16], indicating that the spectra extracted by our new scheme are more efficient. The reason our approach did not perform as well in s1196 and s1238 is that these two circuits are randomly testable and thus the size of the compacted test set for a single characteristic fault is too small, typically only 5 vectors. Due to this reason, spectral information for the entire circuit cannot be fully captured by these few vectors.

To increase fault coverage, the immediate thing to do is to increase the number of characteristic faults. For those circuits which saw a loss in fault coverage with respect to [16], we increased the size of characteristic fault set, k, gradually until the fault coverage reaches that of STRATEGATE. The results are reported for increasing number of characteristic faults in Table 2. Once the fault coverage reached a desired level, we discontinue increasing the number of characteristic faults, and a "/" is placed under that column. In circuits s1196 and s1238, both of which are randomly testable, by increasing the number of characteristic faults, the fault coverages improved significantly as expected. For many other circuits, increasing the number of characteristic faults did not improve the results, since the single characteristic fault already achieved very high fault coverages.

Table 3 reports the speedup and storage reduction between our technique and [16]. The time reported is in seconds and the storage reported is the size of the final spectral coefficient set. Take s382 for example. The execution time for [16] and ours are 261 seconds and 212 seconds respectively. The speedup is due to the fact that we are filtering/compacting for the characteristic faults only. The number of spectral coefficients in our approach is 58, while the size by [16] would require 567. For this small circuit, nearly 1 order of magnitude reduction in storage is achieved. Results for s5378 showed that more than 2 orders of magnitude reduction in storage is achieved, with only 24 spectral coefficients needed for the circuit.

6 Conclusion

We have presented an effective approach for logic BIST using characteristic faults and spectrum information. We demonstrated that many faults in a circuit share similar spectral characteristics, which can be captured by a small number of characteristic faults. By intelligently selecting a small set of characteristic faults, we are able to spectrally characterize the circuit, and this information is sufficient in aiding BIST to obtain extremely high fault coverages. Our technique achieves the same or higher fault coverage than previously proposed BIST approaches, while the computational effort and storage needed are much less since our target changes from the total detected fault set to only a few target faults. Future work will further the characterization of the circuit and evaluation of different characteristic fault sets.

References

[1] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," *Int. Symp. on Circuits and Systems*, 1989, pp. 1929-1934.

[2] S. Davidson and Panelists, "ITC '99 benchmark circuits- preliminary results," *Proc. Int. Test Conf.*, 1999, pp. 1125.

[3] H. J. Wunderlich, "Multiple distribution for biased random test patterns," *Int. Test Conf.*, pp.236-244, 1988.

[4] F. Muradali, T. Nishada, and T. Shimizu, "Structure and technique for pseudo random-based testing of sequential circuits," *JETTA*, vol. 6, Feb. 1995, pp. 107-115.

[5] M. F. Alshaibi and C. R. Kime, "Fixed-biased pseudorandom built in self test for random pattern resistant circuits," *Proc. Int. Test Conf.*, pp. 929-938, 1994.

[6] M. Bershteyn, "Calculation of multiple sets of weights for weighted random testing," *ITC*, pp. 1031-1040, 1993.

[7] D. Kagaris and S. Tragoudas, "Generating deterministic unordered test patterns with counter," *IEEE VLSI Test Symp.*, pp. 374-379, 1996.

[8] M. Chatterjee and D. K. Pradhan, "A novel pattern generator for near-perfect fault coverage," *IEEE VLSI Test Symp.*, pp. 417-425, 1995.

Table 2: Fault Coverage for Varying Number of Characteristic Faults

Circuit	1 Chara Fault	4 Chara Faults	6 Chara Faults	10 Chara Faults	14 Chara Faults	16 Chara Faults
s382	357	357	364	/	/	/
s400	376	384	/	/	/	/
s526	442	444	452	/	/	/
s1196	1227	1230	1233	1237	/	/
s1238	1266	1273	1276	1280	1281	1282

Table 3: Computational Cost and Storage Requirement

Circuit	Execution Times			Storage Requirements		
	[16] Time(s)	Our Time(s)	Time Speedup	[16] Storage	Our Storage	Storage Reduction
s382	261	212	1.23	567	58	9.78
s400	267	216	1.24	567	75	7.56
s526	475	254	1.87	1106	62	17.84
s713	297	191	1.55	89	4	22.25
s1196	124	101	1.23	232	5	46.4
s1238	189	142	1.33	243	5	48.6
s1423	17400	1345	12.94	14421	34	424.15
s1488	731	435	1.68	413	20	20.65
s1494	768	514	1.49	413	20	20.65
s5378	7085	4841	1.46	3426	24	142.75
b01	66	54	1.22	62	8	7.75
b04	659	462	1.43	158	8	19.75
b08	193	145	1.33	428	21	20.40
b11	594	354	1.68	2003	90	22.26
b12	20700	3631	5.70	12866	115	111.88

[9] B. Koenemann, "LFSR-coded test patterns for scan designs," *IEEE European Test Conf.*, pp. 237-242, 1991.

[10] S. Venkataraman, J. Rajski, S. Hellebrand and S. Tarnick, "An efficient BIST scheme based on reseeding of multiple polynomial linear feedback shift registers," *IEEE Int. Conf. Computer-Aided Design*, pp. 572-577, 1993.

[11] C. Fagot, O. Gascuel, P. Girard and C. Landrault, "On calculating efficient LFSR seeds for built-in self-test," *Proc. European Test Workshop*, pp. 7-14, 1999.

[12] K. Radecka, J. Rajski, and J. Tyszer, "Arithmetic built-in -self-test for DSP cores," *IEEE Trans. CAD*, vol. 16, no. 11, Nov. 1997, pp. 1358-1369.

[13] S. Hellebrand and H. J. Wunderlich, "Mixed-mode BIST using embedded processors," *ITC*, pp. 195-204, 1996.

[14] R. Dorsch and H. J. Wunderlich, "Accumulator based deterministic BIST," *Proc. ITC*, pp. 412-421, 1998.

[15] L. Chen and S. Dey, "DEFUSE: A deterministic functional self-test methodology for processors," *Proc. VLSI Test Symp.*, pp. 255-262, 2000.

[16] 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," *Proc. VTS*, 2001, pp. 163-168.

[17] S. Sheng, A. Jain, M. S. Hsiao and V. D. Agrawal, "Correlation analysis for compacted test vectors and the use of correlated vectors for test generation,", *IEEE Intl. Test Synthesis Workshop*, 2000.

[18] A. Giani, S. Sheng, M. S. Hsiao, and V. Agrawal, "Efficient spectral techniques for sequential ATPG," *Design Automation & Test in Europe Conf.*, 2001, pp. 204-208.

[19] C. Fagot, P. Girard, C. Landrault, "On using machine learning for logic BIST," *Proc. ITC*, 1997, pp. 338-346.

[20] A. Raghunathan, S. T. Chakradhar, "Acceleration techniques for dynamic vector compaction," *Proc. Intl. Conf. Computer- Aided Design*, 1995.

[21] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Dynamic state traversal for sequential circuit test generation," ACM Trans. Design Automation of Electronic Systems, vol. 5, no. 3, pp. 548-565, July, 2000.