Efficient Spectral Techniques for Sequential ATPG

Ashish Giani[†], *Shuo Sheng*[†], *Michael S. Hsiao*[†], *and Vishwani D. Agrawal*[‡]
[†]Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854
[‡]Bell Labs, Murray Hill, NJ 07974

Abstract

We present a new test generation procedure for sequential circuits using spectral techniques. Iterations of filtering via compaction and spectral analysis of the filtered test set are performed for each primary input, extracting inherent spectral information embedded within the test sequence. This information, when viewed in the frequency domain, reveals the characteristics of the input spectrum. These spectral characteristics are then used to generate future vectors. We develop a fault-dropping technique to speed up the process. We show that very high fault coverages and small vector sets are consistently obtained in short execution times for sequential benchmark circuits.

1. Introduction

Simulation-based test generation began with random test generation, which used a pseudo-random pattern generator [1]. However, random testing generally results in large test sets [2] and they are useful for circuits without *random-pattern-resistant* faults [3]. Weighted random patterns have been found to yield better fault coverages in circuits that contain such random-pattern-resistant faults [4, 5]. In these approaches, the probability of obtaining a 0 or 1 at a particular input is biased towards detecting random resistant faults. However, the difficulty arises when no one set of weights may be suitable for all faults. In sequential circuits, faults may need a biased internal state in addition to biased input values, making it more difficult to obtain a good set of weights.

Recently, static compaction [6, 7] has been used to aid test generation. A specific feature of vector-restoration based compaction [6, 7] is that the resulting compacted test set guarantees to retain the original fault coverage. Various test generation methods based on compaction have been proposed [8–12] in which repeated calls to static compaction on modified test sets are performed. During each iteration, the test set is first modified by appending new vectors, then static compaction is called to remove any unwanted vectors. This process is repeated until a satisfactory coverage or a maximum number of iterations has been reached. In [8, 9], new vectors are appended to the test set by randomly choosing vectors from the compacted test set obtained in the previous iteration, while in [10–12], spatial and temporal correlations among test vectors are used to append new vectors to the test set.

In this work, we view the sequential circuit as a black-box system that is identifiable and predictable from its input-output signals, instead of viewing it as a netlist of primitives. In studying a signal, what we care most is the predictability of the signal. If the signal is predictable, we can use a portion of it (the past and the current) to represent and reconstruct its entirety. Testing of sequential systems, then, becomes the problem of constructing a set of waveforms, which when applied at the primary inputs of the circuit, can excite and propagate targeted faults in the circuit. These input waveforms (spectra) have specific spectral characteristics, as exhibited in all signals. In order to capture the spectral characteristics of a given signal, a clean representation for that signal is desired (wider spectra lead to more unpredictable/random signals). Thus, any embedded noise should be filtered. Static test set compaction reduces the size of the test set by removing any unnecessary vectors while retaining the useful ones. In other words, static compaction *filters* unwanted noise from the derived test sequence, leaving a cleaner signal (narrower spectrum) to analyze. Taking this idea to test generation, the spectral information obtained not only helps to identify embedded spectral information, it also offers a *new* way for testing sequential circuits by predicting intelligent vectors based on the vectors we have so far. Vectors generated from the narrow spectrum have better fault detection characteristics.

We also developed a technique to speed up the test generation process. Instead of using fault sampling during compaction to reduce the execution time, previously detected faults are periodically removed from the target fault list with the corresponding compacted vector sequence saved. In other words, compaction is performed using only the remaining faults. This significantly reduces the work during each iteration of compaction.

2. Overview and Motivation

Because what we care about most is the information embedded in the input signals (test set we already have), we want to employ signal processing techniques to ex-

1530-1591/01 $10.00 ©2001 IEEE

tract this information. Frequency decomposition is the most commonly used technique in signal processing. A signal can be projected to a set of independent waveforms that have different frequencies. This set of waveforms, each represented as a vector, forms a basis matrix. The projection operation (a post-multiply to the basis matrix) reveals the quantity each basis vector contributes to the original signal. This quantity is called decomposition coefficient. Subsequently, enhancing the important frequencies and suppressing the unimportant ones ("noise"), we expect that we can have a new and higher-quality signal that will help test generation.

In choosing the projection matrix, Hadamard transform is a well-known non-sinusoidal orthogonal transform in signal processing. It consists of only 1's and -1's, which makes it a good choice for the signals in VLSI testing (1 = logic 1, -1 = logic 0). Each basis in the Hadamard matrix is a distinct sequence that characterizes the switching frequency between 1s and -1s. For these reasons, we will use Hadamard transform for our analysis.

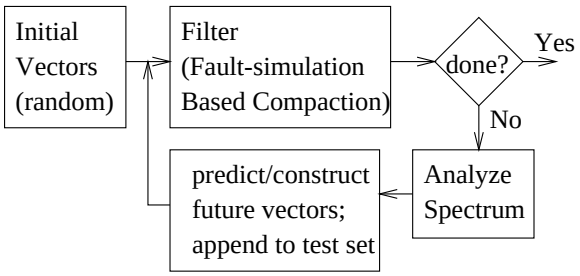


Figure 1. Test generation framework.

Figure 1 presents the overall framework of the spectral test generation procedure. Initially (iteration 0), the test set simply consists of random vectors. A call to static compaction will *filter* any unnecessary vectors. Using the Hadamard transform on the test set obtained, we *analyze* and *identify* the predominant pattern at each primary input. We generate test patterns based on this identified spectrum and filter out any unwanted random bits. At the same time, we can generate vectors spanning the *likely* vector space using only the basis vectors. This can potentially help drive the circuit into hard-to-reach states that require specific vectors at the primary inputs, making it easier to detect hard-to-detect faults faster. This process is repeated until either (1) desired fault coverage is obtained, or (2) maximum number of iterations is reached.

Hadamard matrices are square matrices containing only 1s and -1s, and can be generated using the following recurrence relation:

$$H_h(k) = \begin{bmatrix} H_h(k-1) & H_h(k-1) \\ H_h(k-1) & -H_h(k-1) \end{bmatrix}, \quad k = 1, 2, ..., n,$$

where $H_h(0) = 1$ and $n = log_2 N$. For example, with k=1 and k=2, above equation yields

$$H_h(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_h(2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

From this definition of the Hadamard matrix, we can observe that $H(n) \times H(n)^T = nI_n$, where I_n is the $n \times n$ Identity Matrix. Given only 1's and -1's in the matrices, multiplication can essentially be computed using additions and subtractions. Moreover, the inverse transform of a Hadamard matrix is the same as the forward transform, making reconstruction straight-forward.

Each row/column in a Hadamard matrix is a basis vector, carrying a distinct frequency component. Taking $H_h(2)$ for illustration, the four basis vectors are [1 1 1 1], [1 0 1 0], [1 1 0 0], and [1 0 0 1]. Any bit sequence of length 4 can be represented as a linear combination of these basis vectors. For instance, the vector [1 0 0 0] can be written as $-1 \times [1,1,1,1] + 1 \times [1,-1,1,-1] + 1 \times [1,1,-1,-1] + 1 \times [1,-1,-1,1]$. Therefore, what we can do is project the test sequence to the Hadamard bases, filter out certain frequencies and do an inverse transform to get the *de-noised* sequence.

3. Test Generation Approach

The goal of the spectral technique is to generate/construct intelligent vectors from one iteration to the next. Based on the predominant spectra identified using the Hadamard transform, we construct new test patterns. Algorithm 1 illustrates the construction of such vectors.

Algorithm 1:

Let a_i be the input bit sequence for primary input i.

---

for (each primary input i in test set)
  coefficient vector $c_i = H \times a_i$
for (each value in the coefficient matrix $[c_0, ..., c_n]$)
  if (absolute value of coefficient < cutoff)
    Set the coefficient to 0.
  else
    Set the coefficient to 1 or -1, based on its abs value.
for (each primary input i)
  extension vector $e_i = modified\ c_i \times H$
if (weight > 0)
  Extend the vector set with value 1 to PI i.
else if (weight < 0)
  Extend the vector set with value -1 to PI i.
else if (weight == 0)
  Randomly extend the vector set with either 1 or -1

We will illustrate this algorithm with an example. Consider a subsequence of eight 4-input vectors. We first replace each '0' with with a -1:

PI_0	PI_1	PI_2	PI_3		PI_0	PI_1	PI_2	PI_3
1	1	1	0		1	1	1	-1
1	0	0	1	replace 0	1	-1	-1	1
1	1	0	0	with -1	1	1	-1	-1
1	1	0	1	$\longmapsto$	1	1	-1	1
1	1	1	1		1	1	1	1
1	0	0	0		1	-1	-1	-1
1	1	0	0		1	1	-1	-1
1	0	0	1		1	-1	-1	1

Next, we perform spectral analysis. Consider the bit stream for primary input PI_1. Multiplying $H_h(3)$ with the bit stream for PI_1, we obtain its coefficient vector:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ -2 \\ 2 \\ 2 \\ -2 \\ -2 \\ 2 \end{bmatrix}$$

If the cutoff for coefficients is set to 4, then the coefficient vector c_i is modified by replacing every coefficient whose absolute value is less than 4 with 0. Thus, the new c_1 becomes $[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]^T$ Now, multiplying the new coefficient $[c_1]^T$ with $H_h(3)$ yields:

$$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0] \times H_h(3) = [1\ -1\ 1\ -1\ 1\ -1\ 1\ -1]$$

We will extend the test set for PI_1 with (1, -1, 1, -1, 1, -1, 1, -1). Here, we have filtered out the random pattern that appeared in the input bit stream, since bit #4 has been changed from 1 to -1.

Now let us consider a different primary input PI_3. Multiplying H with PI_3 yields coefficients shown below:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \\ 1 \\ 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -4 \\ 0 \\ 4 \\ 0 \\ -4 \\ 0 \\ -4 \end{bmatrix}$$

If the cutoff for coefficients is again 4, then the coefficient vector is changed to $[0\ -1\ 0\ 1\ 0\ -1\ 0\ -1]$. Multiply the new coefficient with H_8 yields the following extension vector:

$$[0\ -1\ 0\ 1\ 0\ -1\ 0\ -1] \times H_h(3) = [-2\ 2\ -2\ 2\ 2\ -2\ -2\ 2]$$

Extending the compacted vector set for PI_3, we obtain scaled vector (-1, 1, -1, 1, 1, -1, -1 1). This is the same as the input sequence, which means that there was no random noise that needed to be filtered out.

Similarly, we obtain extended vector sets for PI_0 and PI_2 as (1, 1, 1, 1, 1, 1, 1, 1) and (1, -1, -1, -1, 1, -1, -1, -1).

Thus, we append the following newly generated vectors to the test set:

```
 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
```

Extending the compacted vector set in the above manner would lead to the extended vector set having only twice as many vectors as the compacted vector set. In order to append more vectors, we modify Algorithm 1 as follows: ($H_h(4)$ is used in Algorithm 2 and the values of the coefficients range between +16 and -16)

Algorithm 2:

For each cutoff value of 6, 8, and 10 do
  Perform Algorithm 1.
  Randomly pick p vectors from the extended test set, and hold each vector an arbitrary number of cycles.

Increasing the cutoff value is equivalent to fine-tuning the filter, leading to bit patterns very similar to the original input stream. This technique leads to the generation of a sizeable number of "useful" vectors, which helps detect hard-to-detect faults rapidly.

4. Speeding up Test Generation with Fault Dropping

One of the ways to reduce the computation costs of simulation-based test generation is by reducing the number of faults during simulation. Fault sampling is used during the compaction process in [8,9], in which a sample of 128 or 256 randomly chosen faults were used as targets during fault simulation and compaction. As more faults become detected, the target fault list is replenished, until all the faults are included in the target fault list.

Instead of using fault sampling to reduce the test generation time, we developed a *fault dropping* technique to speed up test generation. In fault dropping, detected faults are periodically removed from the target fault list, reducing the time taken for compaction. Figure 2 illustrates the difference in the two concepts. In the fault sampling approach, test generation starts with a smaller number of target faults, and gradually add new faults to the target list. This is illustrated by the rising curve. On the contrary, the fault-dropping technique starts with the entire fault list, and remove the detected faults periodically to reduce the target fault list. This is illustrated by the falling curve. When faults are removed, the test sequence that detected those faults must be saved. Finally, these

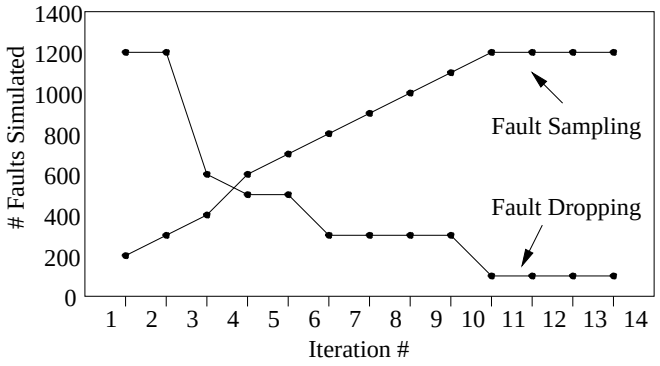


Figure 2. Fault sampling vs. fault dropping.

saved sequences are combined and a final compaction is called to produce the final test set. In smaller circuits that required only a few iterations to obtain the desired fault coverages, faults are removed every 1 or 2 iterations. For larger circuits, we remove faults less frequently - 4 or 5 iterations between removal of faults.

5. Experimental Results

All experiments were conducted on an Ultra SPARC 10 with 256 MB of RAM for ISCAS89 [16] and ITC99 [17] benchmark circuits. The 16×16 Hadamard matrix $H_h(4)$ is used for all circuits. Results for four test generators are compared: HITEC [14], STRATEGATE [15], PROPTEST [9], and the proposed spectral method. HITEC is a deterministic test generator, STRATEGATE is a genetic based test generator and PROPTEST is a compaction-based test generator.

Table 1 reports the results: the circuit name is first given, then, the number of faults detected, test set size, and execution time (in minutes) are reported for each ATPG method. Note that the different platforms were used for different test generators:. HP 9000 J200 for HITEC, Sun UltraSPARC 1 for STRATEGATE, Pentium II for PROPTEST, and Sun UltraSPARC 10 for the spectral technique. PROPTEST [9] used fault samples of 256 initially, and gradually increase the sample size until all the undetected faults are targeted in later iterations. On the other hand, the spectral technique used the proposed fault-dropping technique. We fixed an upper bound of 125 iterations as the terminating condition.

From Table 1, we observe that in all cases, the spectral technique was able to obtain very high fault coverages very quickly, and the test sets were also very compact. For instance, in circuit b12, our spectral technique detected 1645 faults, 175 more faults than best reported coverages. For the rest of the circuits, the spectral technique was able to reach the maximum coverages as well. In terms of test set sizes, the spectral technique results in smaller test sets for many of the circuits. The execution times were less than HITEC and STRATEGATE for most cases, and they were slightly longer than PROPTEST due to

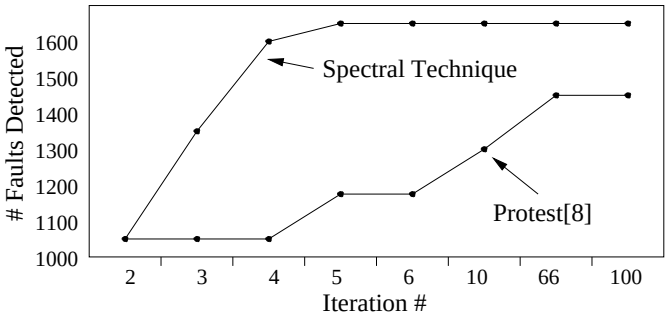


Figure 3. Faults detected per iteration for b12.

different platform, programming style, and the extra effort needed to obtain highly compact test sets and to perform spectral analysis.

To see whether the spectral technique is truly effective in generating intelligent vectors, we compare the number of faults detected at each iteration of the test generation process. We implemented a compaction-based test generator similar to [8], and used that test generator as a comparison. Figure 3 shows the number of faults detected in circuit b12 over the first 100 iterations by our spectral technique and by our implementation of [8]. As shown in the figure, the spectral technique detects 1600 faults by the fourth iteration, while [8] reached 1400 faults (200 fewer) after 60 iterations. Figure 4 shows comparisons of the number of iterations needed to reach maximum fault coverages by our spectral technique and by our implementation of [8]. As illustrated in this figure, the spectral technique consistently reached the final fault coverage in much fewer iterations than [8]. For example, in s1488, only 3 iterations were needed by the spectral technique, while the compaction-based technique [8] required 79 iterations. This demonstrates the effectiveness and potential of the spectral technique.

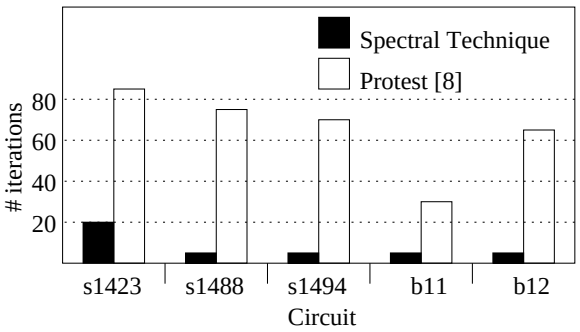


Figure 4. Number of iterations needed.

6. Conclusions

We have presented a novel spectral technique for test generation. Static compaction is first used to filter unwanted vectors. Then, Hadamard transform is used to analyze input spectra. This technique identifies inherent periodicity of the input bits. Vector sequences, then,

Table 1. Test Generation Results

Circuit	HITEC [14]			STRATEGATE [15]			PROPTEST [9]			Spectral-Based		
	Det	Vec	Time	Det	Vec	Time	Det	Vec	Time	Det	Vec	Time
s382	301	1463	90.0	364	1486	8.1	364	572	0.5	364	567	1.0
s400	341	1845	72.0	384	2424	15.5	382	677	0.7	382	588	1.5
s713	476	173	0.1	476	176	1.3	476	104	0.1	476	89	0.4
s1196	1239	435	0.1	1239	574	1.5	1239	224	0.4	1239	244	1.2
s1238	1283	475	0.1	1282	624	1.5	1283	235	0.4	1283	255	1.1
s1423	723	150	834.0	1414	3943	76.2	1416	1049	8.8	1416	927	16.3
s1488	1444	1170	16.5	1444	593	7.5	1444	426	1.9	1444	384	3.2
s1494	1453	1245	9.6	1453	540	7.6	1453	454	2.2	1453	388	2.9
s5378	3231	912	1104.0	3639	11571	2268.0	3643	672	35.6	3643	734	43.5
b01	-	-	-	133	86	0.1	-	-	-	133	50	0.1
b04	-	-	-	1168	2680	10.3	-	-	-	1168	158	0.8
b08	-	-	-	463	1567	0.7	-	-	-	463	387	1.5
b11	-	-	-	1003	4883	50.0	1004	419	1.6	1004	962	2.5
b12	-	-	-	1488	33113	9659.0	1470	3697	27.5	1645	4464	24.2

Time reported in minutes Different platforms were used for different test generators

can be represented as a linear combination of the basis vectors (columns of the Hadamard matrix). Experiments conducted using this spectral technique showed that very high fault coverage with small test sets can be rapidly obtained in a few iterations. Together with the use of the proposed fault dropping method, a considerable computation cost reduction was achieved. In circuits such as b12, we achieved a noteworthy increase in the fault coverage when compared to any previously reported technique.

References

[1] M. A. Breuer, "A random and an algorithmic technique for fault detection test generation for sequential circuits," *IEEE Trans. on Computers,* vol C-20, No. 11, pp. 1364-1370, Nov. 1971.

[2] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital System Testing and Testable Design,* New York, NY: Computer Science Press, 1990.

[3] V. D. Agrawal, "When to use random testing", *IEEE Trans. on Computers,* vol C-27, No. 11, pp. 1054-1055, Nov. 1978.

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

[5] F. Muradali, T. Nishada, and T. Shimizu, "Structure and technique for pseudo random-based testing of sequential circuits," *Journal of Electronic Testing: Theory and Applications,* pp. 107-115, Feb. 1995.

[6] I. Pomeranz and S. M. Reddy, "Vector restoration based static compaction of test sequences for synchronous sequential circuits," *Proc. Intl. Conf. Computer Design,* 1997, pp. 360-365.

[7] R. Guo, I. Pomeranz, and S. M. Reddy, "Procedures for static compaction of test sequences for synchronous sequential circuits based on vector restoration," *Proc. Design, Aut., and Test in Europe,* 1998, pp. 583-587.

[8] R. Guo, I. Pomeranz, and S. M. Reddy, "A fault simulation based test pattern generator for synchronous sequential circuits," *Proc. VLSI Test Symp.,* 1999, pp. 260-267.

[9] R. Guo, S. M. Reddy and I. Pomeranz, "PROPTEST: a property based test pattern generator for sequential circuits using test compaction," *Proc. Design Aut. Conf.,* 1999, pp. 653-659.

[10] A. Jain, V. D. Agrawal, M. S. Hsiao, "On generating tests for sequential circuits using static compaction," *Intl Test Synthesis Workshop,* March 1999.

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

[12] A. Giani, S. Sheng, M. S. Hsiao, and V. D. Agrawal, "Correlation-based test generation for sequential circuits," *Proc. IEEE North Atlantic Test Workshop,* 2000, pp. 76-83.

[13] A. V. Oppenheim, R. W. Schafer, J. R. Buck, *Discrete-Time Signal Processing.* Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1999.

[14] T. M. Niermann and J. H. Patel, "HITEC: A test generation package for sequential circuits," *Proc. European Design Aut. Conf.,* 1991, pp. 214-218.

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

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

[17] S. Davidson and Panelists, "ITC '99 benchmark circuits - preliminary results," *Proc. Int. Test Conf.,* 1999, pp. 1125. also at www.cerc.utexas.edu/itc99-benchmarks/bench.html