HAL
archives-ouvertes.fr

Evaluation of Error-Resilience for Reliable Compression of Test Data

Hamidreza Hashempour, Luca Schiano, Fabrizio Lombardi

To cite this version:

Hamidreza Hashempour, Luca Schiano, Fabrizio Lombardi. Evaluation of Error-Resilience for Reliable Compression of Test Data. DATE'05, Mar 2005, Munich, Germany. pp.1284-1289. hal-00181305

HAL Id: hal-00181305

https://hal.archives-ouvertes.fr/hal-00181305

Submitted on 23 Oct 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Evaluation of Error-Resilience for Reliable Compression of Test Data

Hamidreza Hashempour
IC Development and Applied Research
LTX Corp., San Jose, CA 95134, USA
hhashemp@ece.neu.edu

Luca Schiano, Fabrizio Lombardi
Department of ECE, Northeastern University
Boston, MA 02115, USA
lschiano, lombardi@ece.neu.edu

Abstract

This paper addresses error-resilience as the capability to tolerate bit-flips in a compressed test data stream (which is transferred from an Automatic Test Equipment (ATE) to the Device-Under-Test (DUT)). In an ATE, bit-flips may occur in either the electronics components of the loadboard, or the high speed serial communication links (between the user interface workstation and the head). It is shown that errors caused by bit-flips can seriously degrade the test quality (as measured by coverage) of the compressed data streams. The effects of bit-flips on compression are analyzed and various test data compression techniques are evaluated. It is shown that for benchmark circuits, coverage of test sets can be reduced by 10%-30%.

Index terms: error resilience, fault tolerance, yield, reliable operation of ATE, compression.

1. Introduction

The electronic industry has been highly affected by the widespread use of third party ready-to-use Intellectual Property (IP) cores [1] for assembling complex ICs. Recently, the integration of multiple cores has resulted in the manufacturing of System-on-a-Chip (SoC) on the same silicon die. A complete test set for a SoC includes test sets for each core (often provided by the core vendor). The aggregate volume of these test sets can be very large, in most cases well in the range of a few Gb. The application of such large volume of test data requires significant storage facilities on an ATE and may take long time to execute. Large storage translates into higher equipment cost while long test application time impacts the ATE throughput (DUTs tested per unit time), thus increasing the overall cost of test.

To address these problems, test data compression through so-called *resource partitioning* has been introduced. By compressing the initial test sets, a smaller volume of data is loaded on the ATE, so reducing the cost of storage. Additionally, the compressed test data is transferred to the DUT where it is decompressed and applied. As a smaller volume of data is transferred, then test application time can be reduced too. Traditionally, a reduction in data volume has been accomplished through vector compaction [2], however compression can be applied to already compacted data to further reduce its volume.

Current literature has addressed compression mostly with respect to its efficiency in reducing data volume (commonly quantified by the compression ratio), test application time, and overhead complexity (often for the on-DUT decompression circuitry). In [3], Run-Length coding has been proposed for compression. This has been extended by Golomb and Frequency-Directed-Run length (FDR) coding techniques in which variable length codewords are used for runs of 0 [4] [5]. In [6], Huffman coding has been applied to fixed length symbols. The application of Huffman coding to variable length symbols has been studied in the so-called Variable-Input-Huffman Coding (VIHC) [7]. To avoid the exponential complexity of a Huffman decoder with an increasing length of symbols, [8] has introduced Selective Huffman coding.

For manufacturing testing of VLSI systems such as SoC, the transfer of compressed data between the ATE and the DUT must be also considered. A recent work [10] has statistically evaluated the error-resilience of Huffman coding and proposed a modified version of Tunstall [9] coding to achieve a higher level of error-resilience against bit-flips. Test data has been modified by adding redundancy (one extra bit) to each codeword to preserve vector boundaries in the compressed stream [10]. This is referred to as *bit-padding*. The added bit is 1 if the codeword starts a new test vector, it is 0 otherwise. In this paper, we expand upon [10] by considering the following novel features:

- We show that the so-called "Shift and Propagation" effects discussed in [10] are closely related and Shift is a special case of Propagation.

- We expand our evaluation to 3 other compression techniques, namely, VIHC, Golomb, and FDR.

IEEE COMPUTER SOCIETY

The rest of this paper is organized as follows: Section 2 presents the basic principles of error-resilience and its importance within the context of reliable ATE operation. Section 3 presents the effect of bit-flips on existing compression approaches; it also discusses "Shift and Propagation" effects, and shows their relationship. Section 4 presents the experimental framework of this analysis; results and comparison for four widely used compression techniques are presented for error-resilience. Section 5 concludes the paper.

2. Basic Concepts

In today's ATE architectures, the high volume and the speed of communication may result in the likely occurrence of errors in data transmission. These errors are caused by bit-flips which affect the test data stream. Bit-flips can also occur in the head of the ATE: the presence of a transmission line between the ATE pins and the DUT may be affected by noise in the so-called loadboard, leading to errors due to bit-flips. Moreover, the generation of test inputs as pin waveforms to the DUT, can also introduce an additional source of bit-flips (for example a ground bounce at the DUT receiver connected to the loadboard can result into bit-flips if the voltage levels are close to the margins such as for 3.3V or 1.5V DUTs). In general, once the ATE test program and loadboard are fully debugged, the occurrence of bit-flips can be reduced. However as testing requires very high speed (tens of millions of ATE cycles over hundreds of pins), bit-flips must be considered to be unavoidable in ATE environments [11]. Bit-flips can negatively affect the testing process over its entire spectrum: while developing test programs, bit-flips increase costs associated with debugging and development, while reducing productivity. During manufacturing test, a bit-flip can change data such that the coverage of the erroneous (decompressed) test set (as affected by bit-flips) can be reduced. Reduction in coverage translates into a lower yield and an increase in Defects Per Million (DPM) for products shipped to customers.

2.1. Bit-Flips and Compression

The effect of a bit-flip is significant when test data compression is required. On average, bits in a compressed bitstream carry more information compared with the initial uncompressed stream because compression techniques rely on removing redundancy. Consequently, on average a bit-flip in a compressed bit stream can destroy significantly more information.

Example 1: Consider a compressed test data with a compression ratio of 95%, i.e. a 20 times reduction in data volume. On average, each bit in the compressed bit stream is expanded into 20 bits of the initial uncompressed stream.

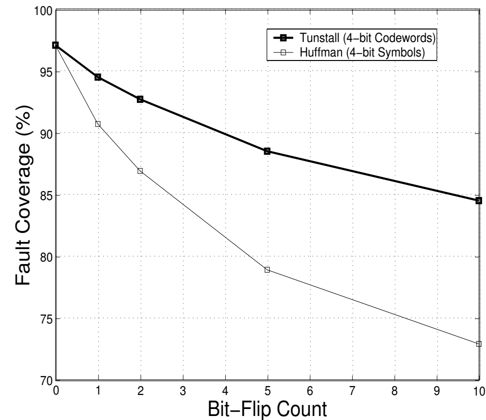

Figure 1. Average fault coverage of Tunstall and Huffman codes over the six largest ISCAS89 benchmarks (with Bit-Padding [10]) versus bit-flip count

If a bit-flip occurs in the compressed bit stream, then we can expect 20 bits to be affected in the uncompressed stream. These bits can be scattered over the uncompressed stream, thus affecting multiple vectors and seriously degrading the coverage of the decompressed test set.

In this paper, we refer to *error-resilience* as the ability of a data stream (encoded using a specified compression technique) to tolerate bit-flips. Error-resilience is measured by the *coverage* of the compressed test stream in the presence of bit-flips. The coverage of the uncompressed test set is equivalent to the error-resilience in the presence of no bit-flip. It also represents the (relative) maximum level of coverage. Other measures for error-resilience (such as the manufacturing yield and Defects-Per-Million (DPM)) can be statistically related to coverage using previously published techniques [12] [13].

An approach which addresses the negative effects of bit-flips, is based on the use of Error-Correcting-Codes (ECC), for example the data downloaded from the ATE workstation to the test head is often coded using additional bits such as parity or Cyclic-Redundancy-Codes (CRC). However, this is not readily applicable to test data compression due to its negative impact on different figures of merit such as the additional hardware resources required on the DUT, the time it may take to correct the flipped bits, and the added redundancy on the compression ratio. A different technique is based on the application of inherent error-resilient compression codes such as Tunstall [9]. Figure 1 compares the coverage reduction as a function of the bit-flip count for Huffman and Tunstall codes. Both techniques have been made partially error-resilient using the bit-padding technique of [10].

IEEE COMPUTER SOCIETY

3. Analysis

A compression technique often partitions the uncompressed test data into a set of allowed symbols $\{s_i | 1 \leq i \leq S\}$, where S is the total number of different symbols. A coding technique maps each symbol s_i to a corresponding codeword c_i. The compressed bit-stream can be represented by an ordering π of codewords, $(c_{\pi(1)}, c_{\pi(2)}, \cdots, c_{\pi(N)})$ where N is the total number of symbols/codewords in the test data.

A bit-flip can change the compressed sequence in a complex manner; this is a function of different parameters such as compression ratio, number of symbols, length of symbols/codewords, and the ordering π. However, a bit-flip always affects a codeword in one of the following manners:

- The affected codeword is broken into $k \geq 1$ valid codewords.
- The affected codeword is broken into $k \geq 0$ valid codewords and a so-called "dangling suffix" which is not a valid codeword.

Example 2: Consider the set of Huffman codewords given by $\{0, 10, 110, 111\}$. A bit-flip affecting codeword 0 results into a zero valid codeword and a dangling suffix of 1 which is not a valid codeword. A bit-flip affecting the first bit of codeword 10, results into 00, i.e. two valid codewords. A bit-flip at the second bit of codeword 111, results into 101, i.e. one valid codeword and a dangling suffix of 1 which is not valid.

The significant feature of the first case is that if a codeword $c_{\pi(i)}$ in the compressed bit stream is affected by a bit-flip, then all codewords $c_{\pi(i+1)}$ to $c_{\pi(N)}$ occurring next, will be unaffected. The only effect of the bit-flip is to have k additional codewords between $c_{\pi(i-1)}$ and $c_{\pi(i+1)}$ as a result of $c_{\pi(i)}$ being broken. In this case, the correct sequence of codewords starting from $c_{\pi(i+1)}$ is shifted by k codewords. This is shown in Figure 2.

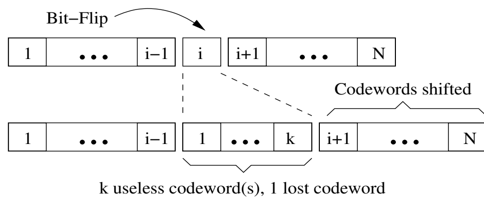

Figure 2. Conceptual view of the shift effect due to a bit-flip

The significant feature of the second case is that if a codeword $c_{\pi(i)}$ is affected by a bit-flip, the dangling suffix will form a valid codeword with a portion of $c_{\pi(i+1)}$. This effectively propagates the bit-flip to $c_{\pi(i+1)}$. If $c_{\pi(i+1)}$ is also left with a dangling suffix, the suffix will form a valid codeword with a portion of $c_{\pi(i+2)}$ and so on. If eventually $c_{\pi(i+j)}$ does not have a dangling suffix, error propagation will stop and $c_{\pi(i+j+1)}$ to $c_{\pi(N)}$ will be unaffected. However, the entire compressed stream between $c_{\pi(i+1)}$ and $c_{\pi(i+j)}$ is corrupted and therefore made useless. This is shown in Figure 3.

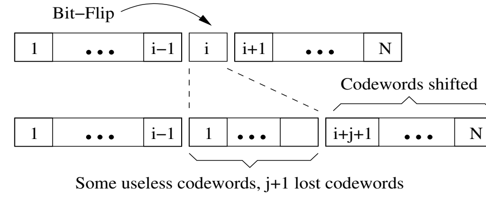

Figure 3. Conceptual view of the propagation effect due to a bit-flip

A simple comparison between these two cases, highlights the relationship between shift and propagation due to a bit-flip. In the first case, a codeword ($c_{\pi(i)}$) is lost and the remaining codewords are shifted by at least one codeword (since $k \geq 1$). In the second case, $j+1$ codewords (from $c_{\pi(i)}$ to $c_{\pi(i+j)}$) are lost, and the remaining codewords are shifted by at least one codeword. Propagation and shift only differ in the number of lost (corrupted) codewords. A shift is a special case of propagation in which only one codeword is lost, i.e. $j = 0$. Propagation always comes with shifts.

An important observation is that a coding technique with constant length codewords, will never have a propagation with $j > 0$, independently of the number of bit-flips or their location. This occurs because a bit-flip can never break a constant length codeword into more than one valid codewords and a dangling suffix is never encountered. This means that Run-Length and Tunstall codes will have at most one lost (corrupt) codeword and a possible shift effect.

3.1. Defect Level Considerations

With bit-flips occurring, portions of the test set can be lost at decompression. This translates into a reduction of coverage, i.e. some faults cannot be tested and defective DUTs with these undetected faults could pass the testing process. These falsely fault-free DUTs will increase the defect level (DL). Let y and DL be the initial yield and defect level respectively. For a coverage f, the following relation holds [13]:

$$DL = 1 - y^{1-f} \tag{1}$$

Assume that the coverages before and after bit-flips are

IEEE COMPUTER SOCIETY

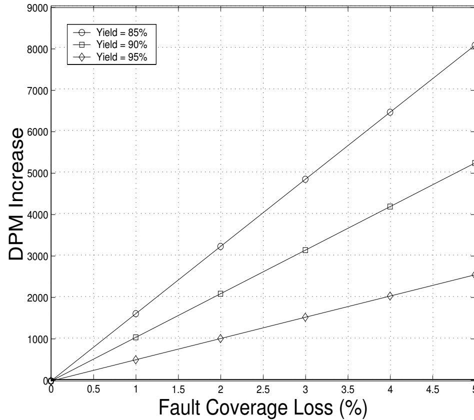

Figure 4. Increase in the number of defective DUTs per million versus fault coverage loss

given by f_1 and f_2 respectively. Then $DL_2 - DL_1$ is the increase in defect level due to bit-flips, i.e.

$$DL_2 - DL_1 = y^{1-f_1} - y^{1-f_2} \quad (2)$$

Figure 4 shows the increase in DPM[1] (the number of Defective parts Per Million DUTs) as a function of coverage loss $f_1 - f_2$ when $f_1 = 100\%$ for three initial values of yield (i.e. y). A very small change in coverage (even 1%) can significantly increase the DPM. This highlights the importance of error-resilience for data compression in a manufacturing environment.

4. Experimental Results

To evaluate error-resilience and related phenomena, the MINTEST dynamically compacted test-cubes for the full-scan versions of the 6 largest ISCAS89 circuits are considered [2]. All don't care bits are set to 0. Test vectors are reordered using the LKH hueristic [14]; however, they are not differentiated because it has been shown that differentiation can further reduce error-resilience [10]. Four widely cited compression techniques (Golomb, Huffman, FDR, and VIHC) are used to compress test vectors of each circuit. Huffman symbols are 4-bit, VIHC symbols are at most 16 bits long, the parameter m of Golomb was chosen for maximal compression per circuit, and the parameter k of FDR is set according to the longest run of 0.

Bit-flips are injected into the compressed test stream of each circuit. The measure of bit-flip count is considered because it is independent of the test data size. The location of the bit-flips is assumed to be *uniformly* distributed

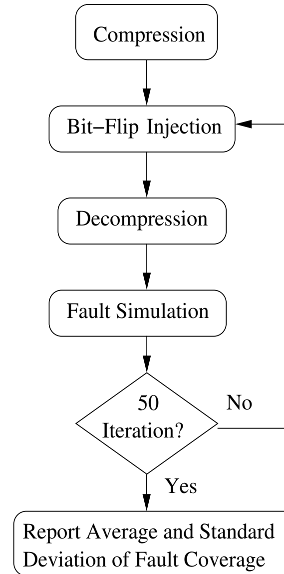

Figure 5. Experimental framework

over the entire compressed bit-stream. The erroneous data is then decompressed and its coverage is established. The reduction in coverage is the measure by which the effect of bit-flips on the error-resilience is evaluated. As the bit-flip location is statistical in nature, then each bit-flip count is repeated 50 times; the average coverage and standard deviation over all experiments are measured. Figure 5 shows the experimental framework for evaluating error-resilience.

Table 1, Table 2, Table 3, and Table 4 show the average coverage and standard deviation as representative of error-resilience of Golomb, Huffman, FDR, and VIHC respectively. Table 5 compares the average coverage over all benchmark circuits for different coding techniques. In all tables, a column corresponds to a specific bit-flip count, for example "1b" means only 1 bit-flip has been injected. Huffman coding results in a much lower coverage loss due to bit-flips. The coverage loss is about 10% for 1 bit-flip and increases in the worst case to 27% for 10 bit-flips in Huffman coding. For only 1 bit-flip, Golomb, FDR, and VIHC have approximately 14% coverage loss from the 97% maximum coverage level. For an initial yield of 85%, this translates into an increase in DPM of almost 22000, i.e. 22000 defective DUTs will be shipped to customers. This is obviously non acceptable[2] and stresses the importance of error-resilient in data compression methods for high quality manufacturing test.

Figure 6 shows the decrease in coverage from its maximum value as a function of the bit-flip count. The results are averaged over all 6 benchmarks examined in Table 1 to Table 4.

[1] $DPM = DL \times 10^6$

[2] Typical DPM values are 500 or less

IEEE COMPUTER SOCIETY

	Standard Deviation				Average Coverage				
Benchmark	1b	2b	5b	10b	1b	2b	5b	10b	Maximum
s5378	7%	8%	6%	5%	91%	86%	80%	77%	99.13%
s9234	15%	16%	11%	9%	72%	65%	50%	51%	93.47%
s13207	13%	11%	8%	6%	85%	77%	70%	65%	98.46%
s15850	10%	11%	7%	6%	78%	77%	66%	65%	96.68%
s38417	10%	10%	7%	6%	87%	82%	78%	73%	99.47%
s38584	10%	9%	6%	5%	81%	75%	72%	70%	95.85%

Table 1. Average and Standard Deviation of fault coverage for Golomb compression

	Standard Deviation				Average Coverage				
Benchmark	1b	2b	5b	10b	1b	2b	5b	10b	Maximum
s5378	9%	8%	7%	5%	90%	85%	80%	75%	99.13%
s9234	16%	15%	11%	6%	80%	68%	64%	52%	93.47%
s13207	13%	11%	8%	6%	86%	79%	70%	67%	98.46%
s15850	13%	12%	8%	4%	82%	76%	67%	66%	96.68%
s38417	11%	10%	6%	6%	87%	84%	75%	73%	99.47%
s38584	10%	9%	6%	5%	81%	76%	72%	68%	95.85%

Table 2. Average and Standard Deviation of fault coverage for FDR compression

	Standard Deviation				Average Coverage				
Benchmark	1b	2b	5b	10b	1b	2b	5b	10b	Maximum
s5378	9%	8%	8%	6%	93%	87%	83%	78%	99.13%
s9234	18%	19%	15%	11%	78%	73%	64%	57%	93.47%
s13207	6%	11%	10%	9%	92%	85%	79%	74%	98.46%
s15850	14%	13%	9%	7%	84%	76%	70%	64%	96.68%
s38417	12%	12%	9%	7%	89%	85%	80%	75%	99.47%
s38584	10%	11%	10%	6%	85%	78%	72%	71%	95.85%

Table 3. Average and Standard Deviation of fault coverage for Huffman compression

	Standard Deviation				Average Coverage				
Benchmark	1b	2b	5b	10b	1b	2b	5b	10b	Maximum
s5378	9%	8%	7%	6%	89%	85%	80%	76%	99.13%
s9234	16%	16%	12%	6%	75%	66%	59%	52%	93.47%
s13207	12%	11%	7%	5%	85%	77%	69%	66%	98.46%
s15850	17%	10%	8%	5%	79%	74%	68%	66%	96.68%
s38417	10%	12%	7%	5%	86%	82%	76%	73%	99.47%
s38584	9%	8%	5%	5%	85%	75%	73%	69%	95.85%

Table 4. Average and Standard Deviation of fault coverage for VIHC compression

IEEE COMPUTER SOCIETY

	Average Fault Coverage			
Coding	1b	2b	5b	10b
Golomb	82%	77%	69%	67%
FDR	84%	78%	71%	67%
Huffman	87%	81%	75%	70%
VIHC	83%	76%	71%	67%
Maximum	97%	97%	97%	97%

Table 5. Average fault coverage over all benchmarks for compression techniques

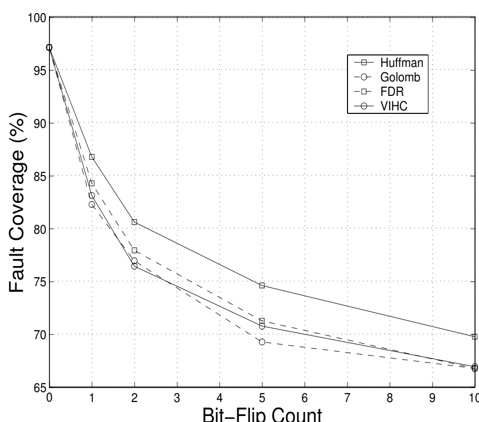

Figure 6. Average fault coverage over all benchmarks versus bit-flip count

5. Conclusion

Error-resilience for reliable test data compression has been introduced in this paper; it has been shown that bit-flips in an ATE can seriously affect compression resulting in corrupted test data for decompression. The reduction in fault coverage of the erroneous test data has been evaluated for four compression techniques. It has been reported that a fault coverage loss of 10%-30% can occur for large benchmark circuits due to the combined effect of data compression and bit-flips which can negatively affect DPM.

References

[1] Y. Zorian, "Test Requirements for Embedded Core-Based Systems and IEEE P1500," *Proc. IEEE Intl. Test Conf.,* pp. 191-199, 1997.

[2] I. Hamzaoglu and J. H. Patel, "Test Set Compaction Algorithms for Combinational Circuits," *Proc. IEEE Intl. Conf. On Computer-Aided Design,* pp. 283-289, 1998.

[3] A. Jas and N. A. Touba, "Test Vector Decompression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs," *Proc. IEEE Intl. Test Conf.,* pp. 458-464, 1998.

[4] A. Chandra and K. Chakrabarty, "System-on-a-Chip Test-Data Compression and Decompression Architectures Based on Golomb Codes," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 20, No. 3, pp. 355-368, March 2001.

[5] A. Chandra and K. Chakrabarty, "Frequency-Directed Run-Length (FDR) Codes With Application to System-on-a-Chip Test Data Compression," *Proc. IEEE VLSI Test Symp.,* pp. 42-47, 2001.

[6] V. Iyengar, K. Chakrabarty, and B. T. Murray, "Huffman Encoding of Test Sets for Sequential Circuits," *IEEE Trans. on Instrumentation and Measurement,* Vol. 47, No. 1, pp. 21-25, Feb. 1998.

[7] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici "Variable-Length Input Huffman Coding for System-on-a-Chip Test," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 22, No. 6, pp. 783-796, June 2003.

[8] A. Jas, J. G. Dastidar, N. M. Eng, and N. A. Touba, "An Efficient Test Vector Compression Scheme Using Selective Huffman Coding," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 22, No. 6, pp. 797-806, June 2003.

[9] B. P. Tunstall, "Synthesis of Noiseless Compression Codes," *Ph. D. thesis, Georgia Institute of Technology,* 1967.

[10] H. Hashempour, L. Schiano, and F. Lombardi, "Error-Resilient Test Data Compression Using Tunstall Codes," *Proc. IEEE Intl. Conf. On Defect and Fault Tolerance in VLSI Systems,* pp. 316-323, 2004.

[11] B. West, "Private Communication," *NPTest.*

[12] V. D. Agrawal, S. C. Seth, and P. Agrawal, "Fault Coverage Requirement in Production Testing of LSI Circuits" *IEEE Journal of Solid State Circuits,* Vol. SC-17, No. 1, pp. 57-61, Feb. 1982.

[13] T. Williams and C. Brown, "Defect Level as a Function of Fault Coverage," *IEEE Trans. on Computers,* Vol. C-30, No. 12, pp. 987-988, Dec. 1981.

[14] H. Hashempour and F. Lombardi, "ATE-Amenable Test Data Compression With no Cyclic Scan Registers," *Proc. IEEE Intl. Conf. On Defect and Fault Tolerance in VLSI Systems,* pp. 151-158, 2003.

IEEE COMPUTER SOCIETY