Concurrent Application of Compaction and Compression for Test Time and Data Volume Reduction in Scan Designs

Ismet Bayraktaroglu and Alex Orailoglu, *Member*, *IEEE*

Abstract—A test pattern compression scheme for test data volume and application time reduction is proposed. While compression reduces test data volume, the increased number of internal scan chains due to an on-chip, fixed-rate decompressor reduces test application time proportionately. Through on-chip decompression, both the number of virtual scan chains visible to the ATE and the functionality of the ATE are retained intact. Complete fault coverage is guaranteed by constructing the decompression hardware deterministically through analysis of the test pattern set.

Index Terms—Test pattern compression, test pattern compaction, scan chains, on-chip decompression, deterministic decompression.

1 Introduction

As the complexity of today's circuits keeps increasing, the applicability of sequential test generation to such designs diminishes. The current design-for-test paradigm consequently has shifted largely toward full scan to ensure easy diagnosis and manageable test generation times. Alternative schemes, such as the utilization of functional patterns, are still employed for detection of speed-related faults; however, the computational resources required for fault grading restrict their applicability to very high volume manufacturers who can amortize the associated cost over the high number of chips that they produce.

While scan improves the quality of test generation, even a relatively small number of patterns applied to the multitudinous scan cells typical in current designs results in inordinately high test data volume and tester time. The increase in test data volume coupled with the high test application times in turn necessitates prolonged utilization of increasingly expensive testers.

As the sizes of the circuits continue to increase, the pin to gate ratio diminishes. As the number of scan chains is limited by the number of I/O pins, the number of cells per scan chain has to increase, directly affecting test application time. As the number of scan cells and scan chains are fixed design-dependent parameters, algorithms to reduce the number of test patterns have constituted the focal points of test pattern generation research.

Reduction of test pattern counts through compaction schemes at no compromise in fault coverage levels have been commonly utilized for quite a while. Test cubes generated for a target fault by ATPG usually consist of a low number of specified bits enabling a set of compatible test cubes to be subsequently combined. The remaining unspecified bits are then randomly set to 0 or 1 to attain a fully specified test vector, enabling ancillary fault dropping through fault simulation. Randomly setting unspecified bits improves the effectiveness of fault dropping, but reduces the effectiveness of compression schemes. Yet, the effectiveness of compression schemes is an increasingly important issue as it helps reduce the size of compacted test sets.

In this work, we propose the utilization of a linear on-chip decompression circuitry based on a compression scheme that enables reduced test volume and application times for circuits with a high number of internal state elements. The number of scan chains visible to the tester is kept constant while increasing the number of internal scan chains, thus providing test data volume and application time reductions at no test pin count increase. The proposed decompression hardware is composed of XOR gates; its hardware complexity is bounded by three XOR gates per scan chain.

While the earlier part of the paper focuses on identifying the proposed approach and benefits of the scheme when applied with a prespecified decompressor, the latter part details the construction of a case-specific decompressor network that precludes any possibility of fault coverage loss. We start by summarizing, in Section 2, the previous work in the area of pattern compression for scan-based designs. Section 3 provides a summary of parameters affecting test application time and test data volume in scan based designs, concluding that reduction in both test application time and test data volume without increased power consumption and ATE cost can only be achieved through the utilization of an on-chip decompression hardware. Section 4 summarizes how the proposed decompression network can be employed in order to drive a higher number of internal scan chains through a set of fixed external scan inputs visible to the ATE.

- *I. Bayraktaroglu is with Sun Microsystems, MS: USUN03-315, 430 N. Mary Ave., Sunnyvale, CA 94085. E-mail: ismet.bayraktaroglu@sun.com.*
- *A. Orailoglu is with the Computer Science and Engineering Department, University of California, San Diego, La Jolla, CA 92093-0114. E-mail: alex@cs.ucsd.edu.*

Manuscript received 30 Nov. 2001; revised 22 Aug. 2002; accepted 12 Dec. 2002.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 115469.

In Section 5, we discuss a basic compaction scheme and the incorporation of the proposed compression into the compaction scheme. We show that the modification to the ATPG process is limited to the augmentation of the compatibility check of the compaction algorithm. Therefore, any given ATPG tool can be modified with minimal effort to incorporate the proposed compression scheme.

We follow this up in Section 6 by discussing various alternatives to the implementation of the linear decompression network. Randomly generated decompression networks provide high compression efficiency, yet introduce a slight risk of causing certain circuit faults becoming untestable. Deterministically generated decompression networks may result in a slight reduction in compression efficiency, but guarantee that no fault whatsoever in the circuit becomes redundant in the context of the decompression network. The experimental results in Section 7 are followed up by conclusions in Section 8.

2 Previous Work

A number of schemes have been proposed for test data volume reduction of scan-based deterministic test by improving the effectiveness of test compaction [21], [1], [3], [10], [22], [23] and compression schemes [18], [14], [11], [15], [16], [17], [5], [8], [6]. While both compaction and compression schemes aim at reducing test data volume, the manner in which they attack the problem of test data volume reduction differs. While compaction schemes aim at reducing the number of test patterns by increasing the number of faults detected by each vector, compression schemes aim at reducing the number of bits required to represent each test vector. Compression schemes may also end up increasing the number of test patterns slightly.

Both compaction and compression schemes have been widely researched. Since, in this work, we propose a compression scheme that aims at reducing the test data volume for scan-based deterministic patterns, we provide a summary of the previous research in this area.

The scan vector compression schemes proposed so far operate either on randomized test vectors or test cubes with unspecified bits. Compression schemes that work on test vectors utilize various coding schemes such as run-length coding [18], Golomb coding [6], statistical coding [14], [15], [5], or geometric shape-based encoding [8]. While compression of the original test vectors provides some level of test volume reduction, a number of these schemes also try compressing the difference between the successive vectors to improve compression efficiency [18], [15], [5], [6]. A cyclic register is then utilized to generate the test vectors from the difference sequences. Such schemes operate directly on test vectors and therefore necessitate no modifications to the test pattern generation process. A set of alternative compression schemes, on the other hand, work in conjunction with test pattern generation. Such schemes include driving multiple scan chains with the same inputs [11], utilization of weighted random patterns [16], and encoding deterministic test vectors as LFSR seeds [17].

Touba and McCluskey proposed the use of a transformation circuitry from a sequence of pseudo-random patterns to a set of deterministic patterns [25]. While the proposed scheme provides high fault coverage with a relatively small number of patterns in a BIST environment, subsequent research has proposed the utilization of a transformation circuitry for input width reduction [7], [4], [12], hence enabling utilization of pseudo-exhaustive testing. Even though the latter schemes result in a higher number of patterns, their hardware overhead is significantly smaller than that of the former scheme. Width reduction for BIST has been achieved by merging directly and inversely compatible inputs [7], by merging decoder(d)-compatible inputs [4], and, finally, by merging Combinational(C)-compatible inputs [12]. While d-compatible inputs require that no more than a single "1" be present at any of the test vectors, C-compatibility necessitates that an input can be driven by a combinational function of other inputs. While increasing the number of compatibility classes increases the efficiency of width compression, identification of such compatibility classes increases the complexity of the algorithm significantly.

3 Parameters Affecting Test Application Time and Data Volume

Test application time and test data volume is affected by a number of parameters, such as the number of patterns, T, the number of scan chains, N, the number of scan cells, S, and the scan frequency, F. The test application time, D, and the test data volume, V, can be calculated by the following equations:[1]

$$D = (T+1) \times \left(\left\lceil \frac{S}{N} \right\rceil + 1 \right) \times F^{-1}, \tag{1}$$

$$V = T \times S. \tag{2}$$

Test data volume can be reduced through various compression schemes. Application of compressed test patterns to the circuit may be performed by decompressing the test patterns on the tester or on chip via a decompression hardware. ATE-driven decompression of test patterns needs to utilize additional software capabilities on the ATE.

While test application time can be reduced by increasing either the scan frequency or the number of scan chains, neither affects test data volume. A slew of further complications detract from the appeal of either solution, furthermore. Increasing the number of scan chains may be simply infeasible as sufficient pins may not be available to accommodate the scan chains. Power issues limit the applicability of ramping up scan frequency. Both techniques necessitate significant increase in ATE costs, furthermore.

Consequently, an acceptable solution to the test time and data volume reduction problem needs to keep both of these parameters intact. We will propose a solution in which the internal scan chain count is increased with no change in either of these parameters, thus reducing both test application time and data volume simultaneously.

1. In these equations, we assume that the scan chains are perfectly balanced and, hence, the ceiling of the ratio of the number of scan cells to the number of scan chains equals the maximum number of scan cells in a scan chain.

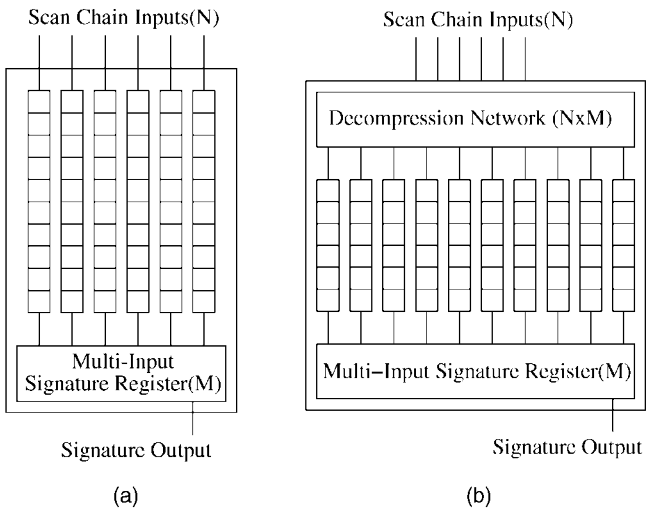

Scan Chain Inputs(N)
Multi-Input Signature Register(M)
Signature Output
(a)
Scan Chain Inputs(N)
Decompression Network (NxM)
Multi–Input Signature Register(M)
Signature Output
(b)

Fig. 1. Scan chain configuration. (a) Regular scan configuration. (b) Hidden scan configuration.

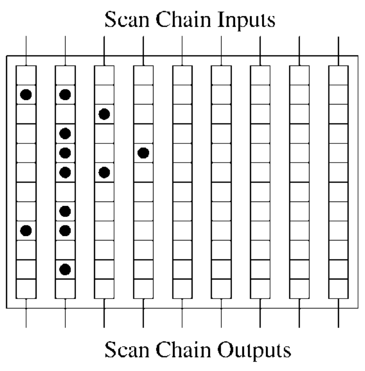


Fig. 2. Fault manifestation in scan chains.

Let's assume that, through a decompression hardware, the number of internal scan chains is increased to M and the number of test patterns increases slightly to T_c. Fig. 1b depicts a possible implementation of the decompression scheme in which the N bit data provided by the tester is decompressed to generate M bit data to the internal scan chains at every clock cycle.

In this case, the new test data volume and test application time can be calculated by the following equations:

$$D_c = (T_c + 1) \times \left(\left\lceil \frac{S}{M} \right\rceil + 1 \right) \times F^{-1}, \tag{3}$$

$$V_c = T_c \times \left\lceil \frac{S}{M} \right\rceil \times N. \tag{4}$$

The percentage reduction in test application time and test data volume therefore can be computed as $(D - D_c)/D$ and $(V - V_c)/V$, respectively. Equation (5) provides an approximation for both test data volume and test application time reduction.

$$1 - \frac{N}{M} \times \frac{T_c}{T}. \tag{5}$$

In the following section, we discuss the compression scheme that we propose in this work.

4 On-Chip Test Decompression

As routing complexity plays a critical role in current designs, ensuring that scan chain stitching imposes no further routing complexity assumes paramount importance. Consequently, scan chains are stitched after placement, inheriting the functional proximity of the underlying flip-flops. Functional proximity in the scan chains implies in turn that scan cells to test for a particular fault may lie in abundance within a scan chain, but may be unlikely to be represented in multiple scan chains. This observation can be illustrated in Fig. 2, wherein seven scan cells in the second scan chain are fully specified, while no more than two scan cells are ever fully specified at any row, thus ensuring that no more than 2 bits are fully specified at any time instance. The reduced number of specified bits helps in achieving high M/N ratios. Taking advantage of this observation, we propose the utilization of a linear decompression network that exploits the existence of the *don't care* bits in the test cubes generated by the ATPG.

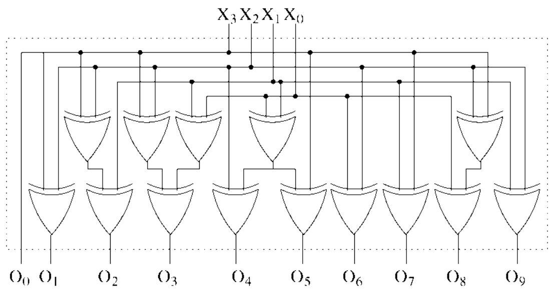
Another observation about the proposed linear decompression network is that the resultant compression ratio is constant. The proposed algorithm for test data compression constitutes a fixed length to fixed length code, consequently, and stands in contrast to previously proposed compression algorithms, which are either fixed length to variable length [18] or variable length to variable length [5], [6]. A variable length in encoding results in variable compression ratios, hence necessitating utilization of complicated synchronization methodologies between the ATE and the on-chip decompression hardware, significantly reducing the applicability of the associated techniques.

We illustrate the basic idea of the proposed technique by depicting a possible implementation in Fig. 3, consisting of a 4 to 10 decompression network. Fig. 3 also provides the linear equations governing this decompression network. The O_is represent known values determined through test generation corresponding to the test cubes, while the X_is are the values that need to be provided by the tester. The technique necessitates the identification of the unknown X_is given the known values of O_is. The equations depicted in Fig. 3 can be written in matrix form.[2] In case O_i is an unspecified bit, no constraint exists that needs to be satisfied. For particular values of O_i, the equations can be solved as long as the number of specified values does not exceed the rank of the matrix. Even when the number of specified values exceeds the rank number of the matrix, a solution may exist, depending on the values specified.[3]

Fig. 4 provides an example for the compression algorithm, wherein there are 10 internal and four external scan chains, each with length 11 scan cells. For each shift cycle in

2. In this matrix multiplication formulation of the equations, XOR operations replace the addition operations.

3. However, to keep the analysis simple, in this work, we only consider the cases wherein the number of specified bits is less than or equal to N and the resultant matrix has full rank. Exploitation of the remaining cases would increase the efficiency of compression at the expense of considerable increase in computational complexity.



$$
\begin{aligned}
O_0 &= X_3 & O_5 &= X_0 \oplus X_1 \oplus X_3\\
O_1 &= X_2 \oplus X_3 & O_6 &= X_0 \oplus X_2\\
O_2 &= X_1 \oplus X_2 \oplus X_3 & O_7 &= X_1 \oplus X_3\\
O_3 &= X_0 \oplus X_1 \oplus X_2 \oplus X_3 & O_8 &= X_0 \oplus X_2 \oplus X_3\\
O_4 &= X_0 \oplus X_1 \oplus X_2 & O_9 &= X_1 \oplus X_2
\end{aligned}
$$

Fig. 3. Decompression hardware and its equations.

this example, we need to generate linear equations and find the solution for X_i. The linear equations corresponding to the decompression hardware in Fig. 3 are provided in (6) in matrix form.

$$
\begin{bmatrix} 0&0&0&1\\ 0&0&1&1\\ 0&1&1&1\\ 1&1&1&1\\ 1&1&1&0\\ 1&1&0&1\\ 1&0&1&0\\ 0&1&0&1\\ 1&0&1&0\\ 0&1&1&0 \end{bmatrix}
\begin{bmatrix} X_0\\ X_1\\ X_2\\ X_3 \end{bmatrix}
=
\begin{bmatrix} O_0\\ O_1\\ O_2\\ O_3\\ O_4\\ O_5\\ O_6\\ O_7\\ O_8\\ O_9 \end{bmatrix}. \tag{6}
$$

Depending on the specified bits of the test cube, a subset of the rows of (6) is utilized for the solution. For the fourth scan cell of the scan chain, for example, only O_2, O_3, O_7, and O_9 are specified, with the corresponding values being $[1\ 1\ 1\ 0]$. Utilization of the rows corresponding to these four outputs results in matrix (7).

$$
\begin{bmatrix} 0&1&1&1\\ 1&1&1&1\\ 0&1&0&1\\ 0&1&1&0 \end{bmatrix}
\begin{bmatrix} X_0\\ X_1\\ X_2\\ X_3 \end{bmatrix}
=
\begin{bmatrix} O_2\\ O_3\\ O_7\\ O_9 \end{bmatrix}
=
\begin{bmatrix} 1\\ 1\\ 1\\ 0 \end{bmatrix}. \tag{7}
$$

This equation can be solved through the utilization of any Gaussian elimination [24] methodology. The solution to this equation is $X_0X_1X_2X_3 = [0\ 0\ 0\ 1]$. The rest of the six bits of the outputs cannot be arbitrarily assigned. They need to be determined through utilization of (6); the output sequence is determined to be $O_0 \cdots O_9 = [1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 0\ 0]$.

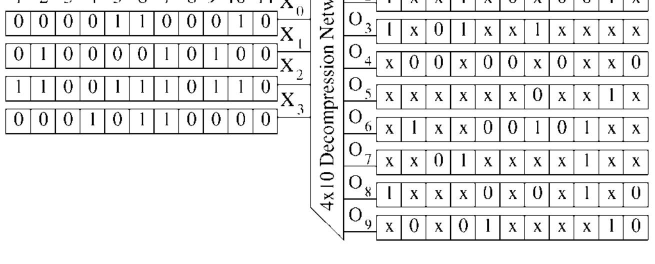


Fig. 4. A compression example for 10 scan chains, each with 11 cells.

The compression algorithm and its interaction with the compaction algorithm during test generation is discussed in the following section. Implementation details of the decompression network are discussed in the subsequent sections.

5 Inclusion of Compression into Test Pattern Generation

The ability of the fixed rate compression scheme that is proposed in this work to encode the test cubes stems from the limited number of specified bits. In order to guarantee complete fault coverage, the compression algorithm needs to be incorporated into the compaction algorithm. While a regular compaction algorithm only checks for compatibility of two test cubes before merging them, the combined algorithm proposed in this work needs to check for compressibility as well before merging the test cubes. The proposed technique necessitates only slight and well-contained modifications to current ATPG methods, thus retaining the significant investment in software in the IC industry. The modifications are limited to the way the compaction algorithm is utilized. To illustrate this point, we select a previously suggested algorithm, namely one by Ayari and Kaminska [1], and discuss the necessary modifications.[4] We improve the algorithm to enable utilization of randomized test patterns by limiting the working test cube size to one, thus reaping the benefits of fault dropping and significantly reducing computational cost. The improved variant of the algorithm is given in Fig. 5.

In order to incorporate the proposed compression scheme into the compaction procedure given above, Steps 2c and 3 of the aforementioned algorithm need to be augmented. After the compatibility check of Step 2c, the resultant test cubes need to be checked for encodability by the decompression hardware. This, in turn, requires setting up a set of linear equations. Unless the equations can be solved, no merging of test cubes can take place. While the additional condition reduces the effectiveness of the compaction algorithm, the increase in the number of resultant test patterns is shown to be

4. The original algorithm checks for compatibility of each newly generated pattern to the previously compacted test cubes, i.e., the current test set. In case a compatible test cube exists in the test set, it is merged with the new test cube and fault simulation is performed for fault dropping. In case no previous test cube is compatible with the new one, the new test cube is added to the test set. As a test cube in the current test set can possibly be merged with a new test cube at any time, the unspecified bits cannot be set randomly for additional fault dropping.

```
1. Get a fault from the fault list and generate a seed test cube ($STC$)
   for the fault, if not available.
2. Repeat forever
   (a) Get a fault from the fault list
   (b) Generate a test cube (or use already generated test cube)
   (c) If compatible with $STC$, merge them, and remove the fault
       from the fault list
   (d) If not compatible and the number of incompatible test cubes
       exceeds a threshold[a], exit the loop
3. Randomize the unspecified bits of the resultant test cube.
4. Perform fault simulation and drop all detected faults from the fault
   list.
5. If undetected faults remain, go to step 1.
```

[a]A lower threshold reduces the number of test cubes that need to be generated significantly, while slightly increasing the number of resultant test patterns. A higher threshold results in a lower number of test patterns generally, at the expense of increased computational complexity.

Fig. 5. Compaction algorithm.

quite small in our experiments. In Step 3 of the algorithm, the unspecified bits are set by simulating the decompression network, resulting in a pseudorandom fill. The resultant algorithm is given in Fig. 6.

The efficiency of the compression algorithm can be further improved by maximizing the number of unspecified bits in a test cube without losing the ability to test the target fault of the test cube.[5] The number of test patterns may be further reduced with no significant impact on the compression ratio through incorporation of more efficient dynamic test compaction algorithms and fault ordering schemes.

The inclusion of the decompression network may cause some faults to become untestable in certain situations. If a test cube generated for a fault cannot be encoded by the decompression network, the case may fall into either of the two categories, differentiated by whether another test cube for this fault exists that is encodable by the decompression network. If no such alternative test pattern exists, detection of the fault necessitates a change in either the number of inputs, N, or the structure of the decompression network. A simple optimistic solution consisting of skipping the fault and hoping that the fault will be gratuitously detected during the fault dropping step of the algorithm suffices if an alternative encodable cube exists. We utilize this simple solution in this work; its adequacy is confirmed by experimental results.

Even though this optimistic approach suffices for most of the cases, in some cases, the target fault becomes untestable due to the linear dependencies in the decompression network. The faults left undetected after the completion of the compression algorithm may need to be handled separately. A possible solution may consist of providing an alternative direct access scheme to the internal scan chains for the remaining faults.

While a direct access mechanism provides a solution to this problem, a deterministic decompression hardware that does not cause any faults to become untestable may be preferable. The following section discusses decompression network alternatives, including the deterministically

```
1. Get a fault from the fault list and generate a seed test cube ($STC$)
   for the fault, if not available.
2. Repeat forever
   (a) Get a fault from the fault list
   (b) Generate a test cube (or use already generated test cube)
   (c) If compatible with $STC$ and *compressible*, merge them, and
       remove the fault from the fault list
   (d) If either not compatible or *not compressible* and the number
       of incompatible test cubes exceeds a threshold, exit the loop
3. *Solve linear equations and determine unspecified bits.*
4. Perform fault simulation and drop all detected faults from the fault
   list.
5. If undetected faults remain, go to step 1.
```

Fig. 6. Compaction and compression algorithm.

generated one, and provides a comparison of their hardware overhead and performance.

6 Decompression Hardware Choices

In this section, we investigate various methodologies for the generation of the decompression network. We start with the possibility of utilizing an LFSR-based decompression network and show that a network with essentially identical performance can be generated at lower hardware cost by utilizing a random scheme. Finally, in this section, we provide a deterministic approach for the generation of the decompression network, which provides 100 percent fault detection efficiency.

6.1 LFSR-Based Decompression Network

Test cubes with a limited number of specified bits have been shown to be encodable with an LFSR seed [13], [17]. Encoding test cubes with LFSR seeds has been utilized in various contexts, such as storing a number of deterministic patterns on-chip [13] or compressing deterministic test cubes [17]. All such schemes are employed to encode a test cube corresponding to a scan chain, as noted previously. A test vector is generated by shifting the most significant bit of the LFSR through a scan chain starting from a precomputed seed.

The sequence generated by an LFSR can be computed through a combinational network in terms of the initial state of the LFSR. The linear equations given in Fig. 3 in Section 4, for example, correspond to the 10-bit output sequence of a 4-bit LFSR, with characteristic polynomial of $x^4 + x^3 + 1$.

LFSR generated sequences usually have a low number of linear dependencies, thus making it highly probable that the full rank property of the resultant matrices holds and making them, in turn, good candidates for decompression network generation. However, an $(N \times M)$ Decompression Network based on LFSR sequences in general requires $M \times (\frac{N}{2} - 1)$ XOR gates,[6] a significant cost for large N. Therefore, in the following section, we consider alternatives to an LFSR-based decompression network and provide a performance comparison of the schemes suggested.

5. The *Maximal Compaction* heuristic proposed by Pomeranz et al. [21], for example, achieves this by converting specified bits to *don't cares* one by one while checking for the detection of the target fault.

6. Computation of a bit value in the output sequence of an N-bit LFSR through a combinational network can be performed by XORing on the average of $\frac{N}{2}$ bits of the initial seed, which necessitates $(\frac{N}{2} - 1)$ 2-input XOR gates.

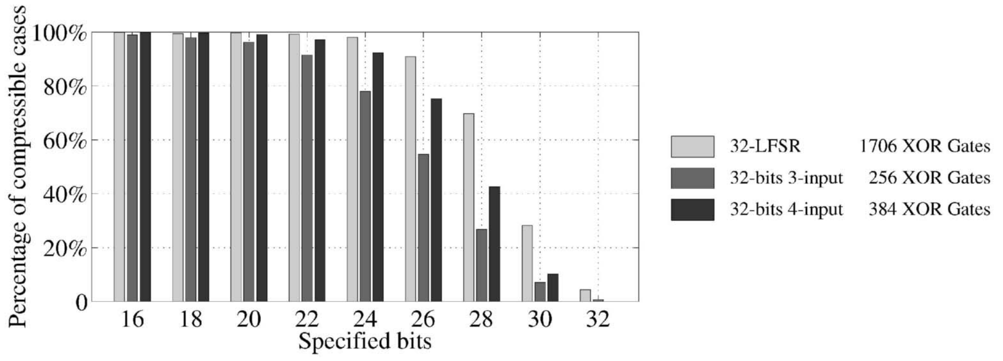


Fig. 7. Performance of decompression networks for 128-bit sequence.

6.2 Randomly Generated Decompression Network

Imposing a threshold on the number of XOR gates utilized per output can help reduce the hardware overhead for the decompressor; the risk in a possible increase in the number of linear dependencies needs to be examined though. We examine XOR networks in which every output is generated by always XORing three or four inputs to assess the possible loss in test vector encodability. In order to reduce the probability of gratuitous linear dependencies, the decompressor outputs are randomly selected out of all possible three (or four) input combinations. The number of outputs, M, supportable by an N-input network whose outputs are obtained by XORing I inputs can be as high as $\binom{N}{I}$.

In a multiple scan chain environment, the number of shift operations per pattern depends on the maximum length scan chain, denoted by K. In the proposed approach, for each test pattern, K linear equations, one per shift, are solved in order to represent M-bit test data with unspecified bits as an N-bit "seed," one per each shift; the resultant "seeds" are stored in the tester in a regular scan buffer. As far as the tester is concerned, there exist only N visible scan chains in the circuit under test. The compression ratio of the proposed scheme mainly depends on the attainable M/N ratio. However, the additional constraints introduced by the linear dependencies may increase the total number of test patterns slightly.

In order to compare the performance of the proposed network structure to an LFSR sequence-based network, simulations are performed to observe the probability that the subsets, ranging from 16 to 32 bits, of a 128-bit sequence are linearly independent. This is performed by randomly generating 10,000 subsets for each set cardinality.

The results provided in Fig. 7 indicate that, for all networks, the probability that a 128-bit sequence with at most 22 specified bits is representable with 32 bits exceeds 90 percent. As the number of specified bits converges to the number of bits available for representing the sequence, the probability of successful encoding diminishes. Additionally, it can be observed that an LFSR-sequence-based network outperforms the 3-input network. However, the 4-input network, at 75 percent lower hardware overhead, performs similarly to the LFSR-sequence-based network. As the performance of 3-input and 4-input decompression networks essentially matches that of LFSR-based ones, we restrict our subsequent examinations in this work to the utilization of the 3-input and 4-input decompression networks.

6.3 Deterministic Decompression Network

During the generation of the linear equations, the network outputs corresponding only to the specified bit positions are utilized. The condition that the resultant matrix have full rank necessitates the linear independence of the outputs of the decompression network corresponding to the specified bits of the test cube. If a more relaxed condition is employed while solving the equations instead of the full rank condition, linear independence would have been required for the incompatible positions of the test cube. However, as discussed earlier, we do utilize the full rank condition to keep the computational complexity of the proposed approach low. In this section, we examine the linear independence condition in detail and show that this problem can be converted to a graph coloring problem. While the graph coloring problem is known to be NP-complete, simple heuristics [9] that provide near minimal color counts exist.

In this work, we utilize a set of test cubes that detect all irredundant faults in the circuits. The decompression network needs to be able to generate all input combinations that are required by the test cubes. Let M denote the number of internal scan chains, K denote the maximum scan chain length, P the number of test patterns, and $T_p(m,k)$ denote the kth bit of the mth scan chain for pattern p, wherein p, m, and k range between 1 to P, 1 to M, and 1 to K, respectively.[7]

Problem Definition. We define a conflict graph with M nodes, each node corresponding to an internal scan chain, $G(V,E)$, with an edge E_{ij} between the nodes i and j if and only if $\exists p,k$ such that $T_p(i,k) \neq "x" \wedge T_p(j,k) \neq "x"$. The time complexity of generation of the conflict graph is $O(P*K*M^2)$. The existence of an edge E_{ij} between the nodes V_i and V_j necessitates that the outputs i and j of the decompression network be linearly independent.

Fig. 8 depicts the compatibility graph[8] for the test cube provided in Fig. 4. The chromatic number of this conflict graph provides the minimal number of inputs required for the decompression network, while the sets of outputs with

7. The value of absent scan cells are assumed to be *don't cares*.

8. In this example, we utilize a slightly different compatibility definition: Two inputs are assumed to be compatible if they have the same values or at least one of them is "x." This compatibility definition reduces the number of conflicts, thus making its visualization with edges less cluttered. While this compatibility condition could also be used in the approach we suggest in the paper, we choose to use the former version to keep the computational complexity low.

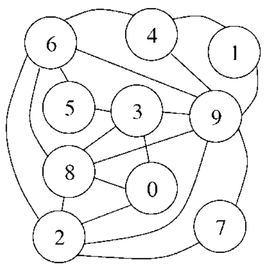


Fig. 8. Conflict graph.

the same color correspond to the sets of compatible outputs, i.e., the sets of outputs that can be linearly dependent. In the simplest case, all compatible outputs can be driven by a single input, resulting in an implementation of the decompression hardware comprised of a simple interconnect network.

The conflict graph in Fig. 8 has a chromatic number of 4 and the compatibility sets are $(0, 5, 9)$, $(1, 6, 7)$, $(2, 3, 4)$, and (8). Therefore, a decompression network with the outputs $O_0 = X_0$, $O_1 = X_1$, $O_2 = X_2$, $O_3 = X_2$, $O_4 = X_2$, $O_5 = X_0$, $O_6 = X_1$, $O_7 = X_1$, $O_8 = X_3$, and $O_9 = X_0$ is sufficient to compress this test cube. Such an implementation of the decompression network is similar to the *Parallel Serial Full Scan* (PSFS) implementation discussed in [11].

While such a deterministic decompression hardware guarantees that all the test cubes are encodable, hence introducing no redundancies to the circuit, the repetition of the inputs introduces high levels of linear dependencies to the decompression network. The introduced linear dependencies, though they do not generate any redundant faults, do reduce the compaction efficiency of the proposed compression algorithm significantly, resulting in a higher number of compressed test patterns, as confirmed by the results provided in Section 7. We examine the possibility, consequently, of alternative implementations that do not result in such a high number of linearly dependent input combinations.

Let S_c denote the compatibility sets of a conflict graph, wherein c ranges between 1 and C, the chromatic number, and let $S_c(i)$ denote the elements of the compatibility sets. The linear independence condition imposed on the decompression network can be shown to be equivalent to the condition that the set of outputs $S_1(i_1), S_2(i_2), \cdots, S_C(i_C)$ be linearly independent for all possible $i_1, i_2, \cdots, i_C$ combinations. Assigning a single input to all of the elements of S_c constitutes a solution to this problem, as noted before. However, as pointed out before, such a solution significantly reduces compaction efficiency.

Another solution to this problem consists of XORing multiple inputs to generate decoded outputs, as performed in Section 3. In this case, however, guaranteeing that the condition holds requires significantly higher computational power. Therefore, we propose a constructive, computationally simple algorithm to generate a decompression network that satisfies the condition.

Let $X_1, X_2, \cdots, X_C$ denote the inputs to the decompression network and let S_cs be sorted in descending order according to their cardinality. We are going to prove that a network that is constructed in the following way is capable of satisfying the linear independence condition. The elements of S_1 are connected to $X_1, X_1 \oplus X_2, \cdots, X_1 \oplus X_{|S_1|-1}$. The elements of S_c are connected to $X_c, X_c \oplus X_{c+1}, \cdots, X_c \oplus X_{c+|S_c|-1}$.

$S_1 = (2, 3, 4)$	$S_2 = (1, 6, 7)$	$S_3 = (5, 9)$	$S_4 = (0)$	$S_5 = (8)$
$O_2 = X_0$	$O_1 = X_1$	$O_5 = X_2$	$O_0 = X_3$	$O_8 = X_4$
$O_3 = X_0X_1$	$O_6 = X_1X_2$	$O_9 = X_2X_3$		
$O_4 = X_0X_2$	$O_7 = X_1X_3$			

Fig. 9. Decompression network generation.

We are going to prove by contradiction that the linear independence condition is satisfied by this hardware. Let's assume that $S_\alpha(i), S_\beta(j), \cdots, S_\gamma(k)$ are linearly dependent and $1 \leq \alpha < \beta < \cdots < \gamma \leq C$. Linear dependence necessitates that $S_\alpha(i)$ be representable by the linear combination of the rest of the outputs. Due to the construction of the decompression network, the input X_α can be utilized by an output only if the output is in $S_c, c \leq \alpha$. Therefore, a linear combination of the rest of the outputs cannot result in $S_\alpha(i)$, implying that linear dependencies cannot exist in the proposed deterministic hardware implementation.

The proposed 2-input network, given a set of compatibility sets, can be generated if and only if $|S_c| \leq C + 1 - c, \forall c$, wherein S_cs are sorted in descending order in terms of their cardinality. While this constraint would not be consistently satisfied by a set of compatibility sets generated through graph coloring, the color assignment to the nodes of the incompatibility graph can be modified so as to satisfy the condition. In case the condition cannot be satisfied through color modifications, the chromatic number of the graph needs to be increased until the condition is satisfied.

Fig. 9 shows the augmented compatibility classes, corresponding input assignment and the equations for the decompression network attained from the conflict graph in Fig. 8. Utilization of additional terms, as in the 2-input network, reduces the possibility of linear dependencies and, therefore, increases compaction efficiency of the algorithm. A 3-input network can similarly be generated to improve the compaction efficiency even further. In case of a 3-input deterministic network, the constraints on the sizes of compatibility sets need to be augmented. The following section provides compression results for both random and deterministic decompression networks.

7 Results

The proposed compression algorithm is built on top of the Atalanta [19] ATPG tool; Hope [20] is utilized for fault simulation purposes. A C program is developed in order to implement the combined Compaction/Compression algorithm. The scheme proposed in this work mainly targets large circuits; therefore, we perform experiments and report on the larger ISCAS89 benchmark circuits. For each of these benchmark circuits, initially, we apply the compaction algorithm described in this work in a stand-alone mode in order to establish a baseline for our comparisons. The threshold for the number of incompatible test vectors before performing fault simulations is set to $1,024$ in order to reduce the computational complexity of the algorithm. The resultant number of test patterns is provided in Table 1. The number of faults in each circuit and the number of test cubes that need to be generated are provided in Table 1 as

TABLE 1
The Number of Compacted Patterns

	Faults	Pattern Count	Patterns Generated
s13207	9664	245	4072
s15850	11336	146	4178
s35932	35110	24	7379
s38417	31015	179	6014
s38584	34797	163	7399

well. The number of faults for which a test cube is generated is significantly reduced by limiting the search space of the compaction algorithm.

The number of test patterns (PC) with inclusion of the compression step for $N = 24$ and $M = 200$ is reported in Table 2 for 3 and 4 input random networks. The results for the deterministic network with $M = 200$ are provided in Table 3. In Table 3, N corresponds to the number of compatibility sets, i.e., the number of external scan chains. The decompression hardware in the special case of a 1-input network for the deterministic case reduces to an interconnect only network. Both of these tables also provide the number of test patterns generated (GTP), which is well below the number of faults in the circuit. In all cases, 100 percent of all irredundant faults are detected by the compressed test sets.

Table 2 provides the number of test cubes that are not encodable (NE) with the decompression network. While test cubes for a certain number of faults cannot be encoded, these faults are indeed detected and dropped from the fault list by compacted test patterns, as the test patterns that cannot be encoded are not the only test cubes that can detect these faults. In the case of s13207, only a few faults require highly specified test cubes, which results in a high chromatic number for the deterministic decompression network. However, a randomly generated 24-input network is able to detect all irredundant faults.

In the case of the deterministic decompression network, all initial test cubes are encodable. However, the highly deterministic nature of this decompression network, while guaranteeing complete fault coverage, reduces compaction efficiency, resulting in a higher number of test patterns.

Table 4 provides the test data volume and test time for the compaction case and for the best compression case. The reductions reported in Table 4 indicate that reductions in excess of 80 percent in both test volume and application time are achievable. While the compression ratio could have

TABLE 2
The Number of Patterns for Compressed Patterns: Random Network

	3-Input Network				4-Input Network			
Circuits	N	PC	GTP	NE	N	PC	GP	NE
s13207	24	246	3698	55	24	248	3688	55
s15850	24	170	3731	0	24	175	3746	0
s35932	24	32	5373	0	24	32	5376	2
s38417	24	263	5423	0	24	267	5537	0
s38584	24	182	6012	67	24	180	6007	0

TABLE 3
The Number of Patterns for Compressed Patterns: Deterministic Network

	1-Input Network			2-Input Network			3-Input Network		
Circuits	N	PC	GTP	N	PC	GTP	N	PC	GTP
s13207	46	428	3600	46	259	3712	46	256	3796
s15850	20	376	3765	21	244	3789	21	196	3778
s35932	21	40	4890	23	38	5373	22	36	5215
s38417	20	801	5091	20	431	5163	20	356	5144
s38584	24	426	5356	24	190	5915	24	185	6010

been increased by searching for various combinations of N and M, in a realistic test environment, the number of scan chains and the number of tester pins available for driving the scan chains are usually predetermined.

We also provide the execution times in seconds for the proposed algorithm in Table 5. The execution time for the ATPG to generate the initial test cubes and the time spent on compaction and compression efforts are provided. Surprisingly, the total execution time in the case of the compaction plus compression scheme that we propose is lower than that of the compaction only case. The reason for this is that, in the case of compression, increased constraints cause the threshold for the fault simulation step to be reached earlier, resulting in a reduced number of test cubes being generated. Even without considering the time spent on ATPG, the compression algorithm increases the time of pure compaction by about 10 percent except for s38417. The last column of Table 5 provides the time required to generate the deterministic network, starting from the initial test pattern set.

Furthermore, we provide a comparison of the results to previously published data, as shown in Table 6. The results are taken from the five recent compression schemes, *Parallel Serial Full Scan* [11], *Golomb Coding* [6], *Virtual Scan Chains* [17], *Geometric Compression* [8], and *Frequency Directed Run-Length (FDR) Codes* [5]. The data volume in terms of the number of bits required to represent the test vectors (without the expected outputs) is provided in the table. The test volumes in bold denote the best test volumes for each circuit hitherto achieved among the five previously proposed schemes.

We summarize the results in two different ways. The last column of the table shows the percentage reduction that our approach attained compared to the best previous results for the particular circuit; of course, the best previous result may be achieved by different techniques, as can be observed in the case of s35932. While this summary in the last column provides a comparison against the best previous results, we follow this up by a summary of the improvements obtained against each previously suggested technique. The average percentage reduction over each previous technique, averaged accross all benchmark circuits, can be seen in the last row of the table. The comparison against the best previous result shows appreciable improvements, perhaps best exemplified by observing that the best possible previous result in the case of the largest benchmark circuit, s38584, exhibits more than a doubling in test application time. An interesting additional insight can be gained by noting that

TABLE 4
Reduction in Test Application Time and Data Volume

	Compactions		Compression		Reduction	
	Volume (bits)	Time (cycles)	Volume (bits)	Time (cycles)	% Reduction in Volume	% Reduction in Time
s13207	171,500	7626	23,616	1235	86.23	83.80
s15850	89,206	3969	16,320	855	81.71	78.46
s35932	42,312	1875	6,912	330	83.66	82.40
s38417	297,856	12780	56,808	2640	80.93	79.49
s38584	238,632	10168	34,560	1629	85.52	83.98

TABLE 5
Algorithm Times

	ATPG	Compaction	Total	ATPG	Compression	Total	Network
s13207	17.82	15.33	33.15	15.16	15.77	30.93	1.37
s15850	23.50	10.55	34.05	20.48	13.18	33.66	2.59
s35932	118.80	6.43	125.23	64.96	7.89	72.85	4.93
s38417	189.71	26.90	216.61	157.06	49.52	206.58	15.87
s38584	123.74	25.11	148.85	98.80	28.66	127.46	15.84

TABLE 6
Test Application Time Improvements with the Proposed Compression Scheme

Circuit	PIs	Parallel Serial Full Scan [11]	Golomb Coding [6]	Virtual Scan Chains [17]	Geometric Compression [8]	FDR Coding [5]	Proposed	Reduction (%)
s13207	700	82,546	35,122	60,894	24,446	**20,368**	24,096	-18.3
s15850	611	76,030	30,581	38,010	22,458	**21,590**	16,320	24.4
s35932	1763	**9,136**	26,885	NA	NA	20,946	6,912	24.3
s38417	1664	127,932	91,088	154,254	60,478	**57,066**	56,808	0.4
s38584	1464	129,580	89,884	101,185	NA	**70,328**	34,560	50.9
Average Reduction(%)		60.5	50.3	61.6	11.6	24.9		

the improvement over the average of the previous results, averaged over all the benchmark circuits, yields reductions in excess of 55 percent.

The experimental results attained in this work indicate that the proposed scheme is capable of providing more than 80 percent reduction in both test data volume and test application time with no case-specific fine tuning of its parameters. Additionally, comparison to previous work indicates that the proposed scheme outperforms previously proposed methodologies significantly, especially for larger circuits.

8 Conclusion

A test pattern compression scheme for large designs that necessitates a high number of scan chains is proposed. The number of visible scan chains is reduced through the utilization of an on-chip decompression network between the ATE outputs and scan chain inputs. The proposed scheme admits utilization of a higher number of scan chains in the circuit with no increase in the pin count requirements.

The storage requirements of ATEs are significantly reduced due to the high compression levels attained in this work. Additionally, test application times are proportionally reduced due to the unhindered increase in the number of scan chains.

A deterministic decompression hardware has been shown to guarantee detection of all irredundant faults. The computational cost of deterministic hardware generation is also shown to be insignificant. However, the highly deterministic nature of the the linear dependencies in such a decompression hardware reduces its compression efficiency compared to the randomly generated networks.

The proposed scheme does not require modifications to the test program and, from the ATE point of view, behaves in a matter identical to regular scan application. Reduction in pin counts and test data volume enables utilization of low cost testers. Consequently, significant cost reductions in manufacturing tests can be achieved by incorporating the proposed compression scheme into the current design, test, and ATE flows.

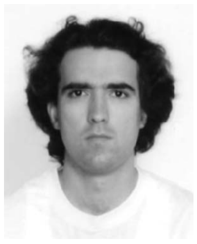
Acknowledgments

A preliminary version of certain sections in this paper can be found in [2].

References

[1] B. Ayari and B. Kaminska, "A New Dynamic Test Vector Compaction for Automatic Test Pattern Generation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 13, no. 3, pp. 353-358, Mar. 1994.

[2] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction through Scan Chain Concealment," *Proc. IEEE Design Automation Conf.*, pp. 151-155, June 2001.
[3] S. Bhatia and P. Varma, "Test Compaction in a Parallel Access Scan Environment," *Proc. Asian Test Symp.*, pp. 300-305, Nov. 1997.
[4] K. Chakrabarty and B.T. Murray, "Design of Built-In Test Generator Circuits Using Width Compression," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 10, pp. 1044-1051, Oct. 1998.
[5] A. Chandra and K. Chakrabarty, "Test Resource Partitioning for SOCs," *IEEE Design and Test of Computers*, vol. 18, no. 5, pp. 80-91, Sept./Oct. 2001.
[6] A. Chandra and K. Chakrabarty, "Test Data Compression and Decompression Based on Internal Scan Chains and Golomb Coding," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 6, pp. 715-722, June 2002.
[7] C.-A. Chen and S.K. Gupta, "A Methodology to Design Efficient BIST Test Pattern Generators," *Proc. IEEE Int'l Test Conf.*, pp. 814-823, Oct. 1995.
[8] A. El-Maleh, S. al Zahir, and E. Khan, "A Geometric-Primitives-Based Cmpression Scheme for Testing Systems-on-a-Chip," *Proc. IEEE VLSI Test Symp.*, pp. 54-59, May 2001.
[9] A.H. Gebremedhin and F. Manne, "Scalable Parallel Graph Colouring Algorithms," *Concurrency: Practice and Experience*, vol. 12, no. 12, pp. 1131-1146, Oct. 2000.
[10] I. Hamzaoglu and J.H. Patel, "Test Set Compaction Algorithms for Combinational Circuits," *Proc. Int'l Test Conf.*, pp. 283-289, Oct. 1998.
[11] I. Hamzaoglu and J.H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores.," *Proc. IEEE Int'l Symp. Fault-Tolerant Computing*, pp. 260-267, June 1999.
[12] I. Hamzaoglu and J.H. Patel, "Reducing Test Application Time for Built-In-Self-Test Test Pattern Generators," *Proc. IEEE VLSI Test Symp.*, pp. 369-375, May 2000.
[13] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois, "Built-In Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers," *IEEE Trans. Computers*, vol. 44, no. 2, pp. 223-233, Feb. 1995.
[14] V. Iyengar, K. Chakrabarty, and B.T. Murray, "Deterministic Built-In Pattern Generation for Sequential Circuits," *J. Electronic Testing: Theory and Applications*, vol. 15, nos. 1-2, pp. 97-114, Aug.-Oct. 1999.
[15] A. Jas, J. Ghosh-Dastidar, and N.A. Touba, "Scan Vector Compression/Decompression Using Statistical Coding," *Proc. VLSI Test Symp.*, pp. 25-29, Apr. 1999.
[16] A. Jas, K. Mohanram, and N.A. Touba, "An Embedded Core DFT Scheme to Obtain Highly Compressed Test Sets," *Proc. Asian Test Symp.*, pp. 275-280, Nov. 1999.
[17] A. Jas, B. Pouya, and N.A. Touba, "Virtual Scan Chains: A Means for Reducing Scan Length in Cores," *Proc. VLSI Test Symp.*, pp. 73-78, May 2000.
[18] A. Jas and N.A. Touba, "Test Vector Decompression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs," *Proc. Int'l Test Conf.*, pp. 458-464, Oct. 1998.
[19] H.K. Lee and D.S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Technical Report 12_93, Dept. of Electrical Eng., Virginia Polytechnic Inst. and State Univ., 1993.
[20] H.K. Lee and D.S. Ha, "HOPE: An Efficient Parallel Fault Simulator," *Proc. IEEE Design Automation Conf.*, pp. 336-340, June 1992.
[21] I. Pomeranz, L. Reddy, and S.M. Reddy, "COMPACTEST: A Method to Compact Test Sets for Combinational Circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 7, pp. 1040-1049, July 1993.
[22] I. Pomeranz and S.M. Reddy, "Static Test Compaction for Scan-Based Designs to Reduce Test Application Time," *Proc. IEEE Asian Test Symp.*, pp. 198-203, Dec. 1998.
[23] R. Sankaralingam, R. Oruganti, and N.A. Touba, "Static Compaction Techniques to Control Scan Vector Power Dissipation," *Proc. VLSI Test Symp.*, pp. 35-40, May 2000.
[24] G.W. Stewart, *Introduction to Matrix Computations.* Academic Press, 1973.
[25] N.A. Touba and E.J. McCluskey, "Transformed Pseudo-Random Patterns for BIST," *Proc. IEEE VLSI Test Symp.*, pp. 410-416, May 1995.



Ismet Bayraktaroglu received the BS and MS degrees in electrical engineering from Bogazici University, Istanbul, Turkey, and the PhD degree in computer engineering from the University of California, San Diego. He is currently a member of the technical staff at Sun Microsystems. He performed the work discussed in this article while pursuing the PhD degree at the University of California, San Diego. His research interests include BIST, diagnosis of BIST designs, test pattern compression, and concurrent test of DSPs.

Alex Orailoglu received the SB degree cum laude from Harvard University in applied mathematics and the MS and PhD degrees in computer science from the University of Illinois, Urbana-Champaign. He has been a faculty member at the University of California, San Diego since 1987, where he is currently a professor of computer science and engineering. He previously held the position of senior member of the technical staff at Gould Research Laboratories, Rolling Meadows, Illinois. His research interests include digital and analog test, fault-tolerant computing, computer-aided design, and embedded processors. He serves on the technical, organizing, and/or steering committees of the major VLSI Test and Design Automation conferences and workshops. He is an associate editor of *IEEE Design and Test*. He served as the technical program chair of the 1998 High Level Design Validation and Test (HLDVT) Workshop and as the general chair of HLDVT '99. He is the founding chair of the Workshop on Application Specific Processors (WASP '02). He is also a member of the IEEE Test Technology Technical Council (TTTC) Executive Committee and currently serves as vice chair of TTTC and as chair of the Test Technology Education Program subgroup. He has previously held the positions of Technical Activities Committee chair and planning cochair of TTTC. He is a Golden Core member of the IEEE Computer Society and a member of the IEEE.