Reducing Test Application Time Through Test Data Mutation Encoding

Sherief Reda and Alex Orailoglu

Computer Science & Engineering Department
University of California, San Diego
La Jolla, CA 92093
e-mail: {sreda, alex}@cs.ucsd.edu

Abstract

In this paper we propose a new compression algorithm geared to reduce the time needed to test scan-based designs. Our scheme compresses the test vector set by encoding the bits that need to be flipped in the current test data slice in order to obtain the mutated subsequent test data slice. Exploitation of the overlap in the encoded data by effective traversal search algorithms results in drastic overall compression. The technique we propose can be utilized as not only a stand-alone technique but also can be utilized on test data already compressed, extracting even further compression. The performance of the algorithm is mathematically analyzed and its merits experimentally confirmed on the larger examples of the ISCAS'89 benchmark circuits.

1 Introduction

As VLSI technology moves to nanometer scales, fabrication facilities have been able to cram more logic into digital devices than ever before. At the same time, designers have been utilizing advancements in fabrication technology to achieve higher levels of unprecedented integration of digital circuits. The ability to manufacture with such high integration has also resulted in increased test volume. The increased test volume necessitates a test time increase which hinders volume manufacturing in a demanding market.

The problem of test time has been exacerbated by the need to test multiple cores in System-on-Chip (SoC) designs. The increasing number of cores, each with its own test data, have resulted in a dramatic increase in the amounts of storage in the Automatic Test Equipment (ATE), well exceeding gigabit levels. The ATE not only has to store prodigious amounts of data but also needs to supply them to the chip in rapid succession in order to shorten test time. These demands have resulted in dramatic increases in current ATE prices. An ATE, at over $3.5 million a piece for a 512 pin, 400-MHz version, constitutes a striking component of overall test costs.

The aforementioned factors have resulted in active research towards improving test costs. There are two main themes that could be followed in order to reduce test costs. The first approach, that of incorporating BIST to the SoC, attempts to reduce the investment in the ATE and decreases test time as well. Yet frequently pseudorandom resistant faults limit the fault coverage attainable by BIST and consequently its applicability. The second approach, that of an on-chip decompression coupled with compressing test data at the ATE, attempts to reduce memory requirements and to alleviate timing constraints on the ATE.

In order to be able to compress the test data, researchers have suggested various compression techniques coupled with building on-chip decompression circuits to decompress the test data stream into test slices to be injected into the scan chains [9], [1], [7], [3], [8]. The overriding goals in the design of compression techniques are comprised of a superior compression ratio and minimal hardware overhead. These often conflicting goals are both simultaneously achieved in the technique we herein propose through cost effective encodings of the flips in the test data.

In the proposed approach, the test data stream is used to indicate which bits need to be flipped in the current test slice to obtain the subsequent one. In order to decompress the test data stream, a decompression hardware consisting of a decoder with its inputs forming a shift register is constructed. Through the use of a decoder and a shift register, we are able to specify which bits need to be flipped in order to obtain the next mutated test slice.

In order to obtain optimal compression ratios, we explore the problem of finding the minimal number of bits needed to traverse a combination of states in the state transition diagram of the decompression circuit shift register. We also develop the mathematical means to analyze the state transition diagram and obtain *a priori* the possible achievable compression ratios. The mathematical analysis is backed up by an extensive set of experimental results on practical

benchmark circuits.

The flexibility of the proposed approach allows its applicability as a stand-alone technique or as an extra layer of compression on top of existing techniques. The successful application of the proposed approach is attributed to the correlation of the test slices in practical circuits which decrease the number of changes between consecutive test slices.

The paper organization is as follows. Section 2 provides a brief overview of the state-of-the-art testing compression techniques. Section 3 illustrates our approach for compression of test data through the use of decoders. Section 4 analyzes the compression ratio that could be achieved using our methodology. Section 5 discusses how to overcome synchronization problems. Experimental results are given in section 6, while section 7 briefly summarizes the contributions of the paper.

2 Previous Work

In response to the increasing testing time of digital circuits, researchers have suggested a variety of techniques to compress the test data [9], [1], [7], [3], [8]. One of the suggested techniques utilizes Huffman encoding [7] to compress the most frequently occurring patterns. The patterns to be compressed are selected so as to minimize the area overhead by the on-chip decompression network. The technique delivers acceptable compression ratios but suffers from synchronization problems, since the bit-output from the decompression network has to be injected into the scan chain at a rate faster than the incoming test data in order to avoid buffer overruns.

LFSRs have been utilized in the work of [9] and [1] to build the required decompression network. The proposed idea in [9] is to reduce the scan chain length visible to the tester to well below the actual scan chain length by dividing the original scan chain into a number of scan chains. The test data stream is used to initialize the seeds of all but one of the LFSR generators. After the LFSRs are initialized, they are run in autonomous mode to fill the scan cells of all but one of the scan chains. The input test data stream is used to initialize the remaining scan chain.

Run length encoding has been proposed in [8] to reduce the test volume. In this work, the authors suggest compressing the difference vectors instead of the test vectors in order to obtain longer runs of 0s, thus achieving higher compression. Golomb coding has been utilized in the work of [3] to produce the necessary test data compression for embedded cores in SoCs. In this work, the authors have proposed the use of variable-to-variable-length Golomb codes and interleaved core testing to achieve effective results.

A novel compression technique has been proposed [1] for reducing the number of scan chains visible to the tester by building a decompression network based on LFSR sequences. In this scheme a test data vector is used as a seed to the decompression network to produce a test bit for each scan chain. The decompression XOR-based network is constructed in a such a way so as to reduce the linear dependencies between the seed bits, enabling superior encoding of the test data vectors. The experimental results demonstrate considerable compression over the method suggested by [9], but at the cost of extra hardware overhead for the decompression network.

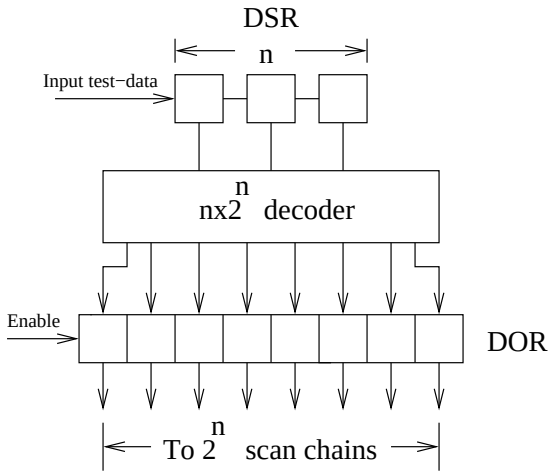

Figure 1. Hardware organization of the proposed decompression technique

3 Compression using decoders

In this work we propose the use of decoders to indicate which bits need to be flipped in the current test slice to obtain the subsequent one. In our approach the test data is used to indicate which bits need to be flipped in the output register of the decompression network, rather than directly loading the scan chain with the test data. Under this scheme the technique can be used as a stand-alone compression scheme or as an augmentation to existing compression techniques with a small hardware overhead, consisting of a small decoder for most practical cases.

The hardware decompression circuit consists of a decoder that receives its inputs from a shift register, the *Decoder Shift Register* (DSR), and delivers its outputs to an output register, the *Decoder Output Register* (DOR). The hardware organization of such a scheme is illustrated in Figure 1. As illustrated in the figure, the DSR receives its inputs from the test data input stream, and a control signal enables the flipping of the DOR bits. After loading the DSR with the position of the DOR bit to be flipped, the control signal is enabled to flip the required DOR bit. The process is repeated until all the required DOR bits are flipped. The

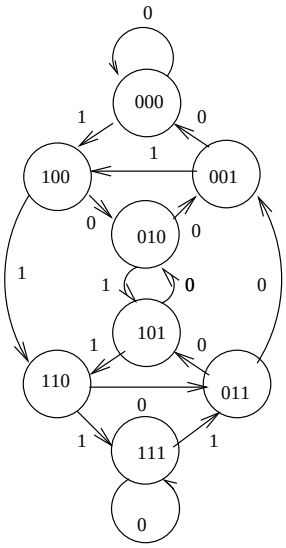

Figure 2. State Transition Diagram of the DSR

DOR result is then loaded into the scan cells by enabling the clock.

The proposed scheme operates by loading the DSR input with the binary representation of the bit positions to be flipped in the DOR. Typically, multiple positions will need to be flipped in order to attain the next mutated test slice; however, no order in the manner in which these indices are loaded in the DSR is preordained. The number of bits needed to be shifted into the DSR to encode the indices of the bits to be flipped depends on the order in which these indices are loaded. In order to calculate the minimum number of bits, we construct the State Transition Diagram (STD) of the DSR. The STD consists of 2^d states where d is the length of the DSR. Any state has exactly two possible next states, corresponding to shifting right and inserting the bit "0" or "1". [1] This STD is referred to commonly in the relevant literature as the DeBruijn diagram [5]. Constructing the STD enables the calculation of the minimal number of bits needed to traverse from one state to the other. This minimal number of bits needed to go from one state to the other is stored in a *distance matrix* with each entry indicating the minimum number of shifts needed to reach the column state from the row state.

Using the distance matrix, the problem of finding the minimum number of bits is transformed to calculating the tour that has the least total number of bits. For practical DSR sizes, this number of bits can be calculated optimally by using a brute-force enumeration algorithm [4]. The following example illustrates the use of the distance matrix in the computation of the optimal tour.

[1] An analogous scheme can be explored that relies on left shifting. However, in the average case, both schemes should produce identical results.

	0	1	2	3	4	5	6	7
0	0	3	2	3	1	3	2	3
1	1	0	2	3	1	3	2	3
2	2	1	0	3	2	1	2	3
3	2	1	2	0	2	1	2	3
4	3	2	1	2	0	2	1	2
5	3	2	1	2	3	0	1	2
6	3	2	3	1	3	2	0	1
7	3	2	3	1	3	2	3	0

Table 1. Distance matrix for state transition diagram in Figure 2

Example 1 Assume the STD given in Figure 2 with the corresponding distance matrix in Table 1. In order to calculate the optimal trip for visiting the states (2, 6), assuming that the initial state is (4), we calculate the total distance for the trips (4, 2, 6) and (4, 6, 2). Consulting the distance matrix, we find that the trip (4, 2, 6) results in a total distance of 3, while (4, 6, 2) results in a distance of 4. Evidently, the optimal trip is (4, 2, 6).

In the previous example, the enumeration was computationally feasible since there were only 2 states to visit. However, if the number of flipped bits is exceedingly large that attaining their optimal order of flipping is computationally infeasible, a greedy method can be utilized instead to compute the order of flipping. The greedy strategy is based on moving to the nearest state from the current state. Starting from the initial state, this greedy strategy is repeatedly applied until all the required states are visited. The following example illustrates an application of the proposed compression approach.

Example 2 Assuming we have an 8-bit DOR feeding 8 scan chains, we would require a 3x8 decoder. The state transition diagram of the 3-bit shift register acting as the decoder input is illustrated in Figure 2. The minimal number of shift bits needed to transition from one state to the other is given by the distance matrix in Table 1. Assume that the initial state of the shift register is "100" and that we would like to flip bits 2 and 6. Using a brute force enumeration algorithm to calculate the optimum trip yields the sequence $4 - 2 - 5 - 6$ with the control signal for flipping the LFSR enabled for all states except state 5. In this case the data shifted in would be "011". This constitutes a reduction of 62.5% in time compared to the original 8 shifts bits. [2]

[2] Though the control signal constitutes an additional increase in the test volume, this is not of much relevance in our case since we focus on test time reduction and the control signal is shifted in parallel to the test data.

An interesting special case of our approach occurs in the case of compressing incompletely specified slices. In this case, the *don't care* values are initialized to minimize the number of flips between two consecutive slices. However, this does not result in the minimal number of shift bits, since it is possible to flip positions corresponding to intermediate states while traversing from an initial state to a final state. This flipping is used to specify a binary value for a *don't care* position in anticipation for a future change. This anticipation saves a shift bit or possibly more later on. This special case is illustrated in the following example.

Example 3 Assume that we are given the following three test slices to compress "11100110", "x0xxx0xx" and "0x0xxxxx", where the leftmost bit (bit 7) is the most significant bit. A straightforward *don't care* initialization yields the following test slices "11100110", "10100010", "00000010". Example 2 has shown that the trip (4, 2, 5, 6) is the optimal one to mutate test slice 1 into 2. In example 2, we had disabled the flipping of bit number 5, but in this example this is not a good choice since this flipping can be achieved at no extra cost and at the same time saves 2 bits in mutating test slice "10100010" to "00000010".

The potential of the proposed compression approach is that it could be also used to further augment most currently proposed compression approaches. In the worst case, we would need to tour all the STD states to flip all the bits. This would require 2^d bits which is equivalent to directly loading the existing decompression networks with the compressed data. However, in the common case the number of bits that need to be flipped is apparently low. From this perspective, our approach is able to achieve large compression ratios as analyzed in the next section.

4 Compression Ratio Analysis

In this section we derive the relation between the length of the DOR corresponding to the number of scan chains and the maximum compression ratio σ that can be achieved using the proposed approach. Assuming ν vectors of n bits to be compressed, with the average number of changed bits in two consecutive vectors denoted by s, we calculate η, the average number of input bits needed to invert s changed bits in a DOR of length n.

$\cap$	0	1	x
0	0	NULL	0
1	NULL	1	1
x	0	1	x

Table 2. Intersection operator semantics

i	j	c^i_j	New Reachable States
000	0	000	1
	1	x00	1
	2	xx0	2
	3	xxx	4
001	0	001	1
	1	x00	2
	2	xx0	2
	3	xxx	3

Table 3. New reachable states calculation

Let $d = \lceil \log_2 n \rceil$ represent the number of bits needed by the decoder to specify an inversion on any of the n bits of the DOR, and let the number of bits shifted into the DSR be denoted by j where $s-1 \leq j \leq 2^d - 1$. These j bits shall be used to flip the required s positions.

Let's assume that the initial state in the DSR register is i and that c^i_j denotes the cube that results from shifting right by j-bits the binary pattern of i. Using k shift bits, the cube c^i_k is able to reach 2^k states; however, some of these states could have been visited by $j \leq k$ bits. In order to calculate the exact number of new states that could be visited using k bits, we take the intersection, according to the semantics defined in Table 2, of c^i_k with all the cubes c^i_j where $j \leq k$. A NULL intersection in Table 2 indicates no common cube between the two cubes being intersected. We demonstrate the operation of the intersection operator by the following example for the simple case where $s = 1$.

Example 4 Let $n = 8$ as in Example 2 and $s = 1$. In order to calculate the average number of bits needed in transitioning from one state to the other, we calculate the number of the new states that can be reached using j bits where $0 \leq j \leq 7$. Assuming that the initial state is "001", we can construct the number of new reachable states using j bits by means of the intersection operator. The corresponding results are given in Table 3. In order to calculate the average number of bits η when $s = 1$, we repeat the previous calculation for each initial state and average the total number of bits. The various values of η for each initial state are given in Table 4.

The previous example illustrates an important key point. The number of shift bits needed to reach s states from an

i	0	1	2	3	4	5	6	7
η_i	2.125	1.875	1.75	1.625	1.625	1.75	1.875	2.125

Table 4. Average number of shift bits in the case of $s = 1$

	s							
n	1	2	3	4	5	6	7	8
4	1.13	1.92	2.50	3.00				
8	1.84	3.13	4.14	4.95	5.62	6.17	6.63	7.00
16	2.66	4.55	6.09	7.36	8.49	9.51	10.43	11.26
32	2.53	6.15	8.31	10.19	11.71	12.39	13.63	15.37

Table 5. Average number of shift bits needed

initial i is not only a function of the initial state but also of the particular combination of the s states to be visited. For the general case where $s \geq 1$, there are $\binom{n}{s}$ possible modifications of the DOR with s changed positions. The average number of bits needed to visit these s positions can be calculated by the following formula,

$$\eta(d,s) = \sum_{i=0}^{2^d-1} \sum_{j=s-1}^{2^d-1} j \cdot \frac{m(i,j,s)}{2^d \cdot \binom{2^d}{s}} \quad (1)$$

where the function $m(i,j,s)$ gives the number of s-state combinations that could be reached from state i using j bits. Table 5 gives the average number of shift bits needed to flip s positions for practical values of n and s. In this table, we have used brute force enumeration to calculate the values for the function m. The average number of shift bits in the last 4 columns of the last row are approximate values calculated by averaging the shift bits for the first 100,000 s combinations.

Using the data in Table 5 and calculating the compression ratio as given by equation (2),

$$\sigma = \frac{n}{\eta(n,s)} \quad (2)$$

we derive the expected compression ratio for some typically encountered values of n and s in practical circuits. The calculated results, given in Table 6, demonstrate that the approach is capable of achieving high compression ratios for small values of s; the experimental results in section 6 confirm this mathematical analysis.

		s			
n	*d*	0	1	2	3
4	2	4	3.54	2.08	1.60
8	3	8	4.35	2.56	1.93
16	4	16	6.02	3.52	2.63
32	5	32	12.64	5.2	3.85

Table 6. Achieved compression ratio

5 Synchronization of the Test and the Enable Signals

In this section we devise a method for the synchronization of the Test Data Input (TDI) and the Enable signal. To avoid an extra pin for the Enable signal, we configure the Test Data Output (TDO) pin as a bidirectional input/output pin. During the input mode, the pin is used as a source for the Enable data, and during the output mode, the pin is used as a means to extract the signature from the MISR.

In a typical test environment, a large number of test patterns are applied to the circuit under test in order to achieve the required fault coverage. After the conduction of these tests, the signature is ready to be shifted out from the MISR through the TDO pin. During the test session, the TDO pin is of no practical use and can be configured as the Enable pin. In order to configure the mode of operation of the TDO/Enable pin, a simple control circuitry is devised. Initially, the control circuit configures the TDO/Enable as an Enable pin, and the Enable signal is shifted in parallel with the test data. After the required DOR bits are flipped and the mutated test slice is attained, the clock is enabled to capture the DOR into the scan cells. This process is repeated until all the scan cells are loaded with the required test patterns.

After the application of all the required test patterns, the TDO/Enable pin should remain idle for a number of clock cycles equal to the width of the DSR of the decompression network; this idle period indicates that there are no more bits to be flipped and signals that the test session is completed. Upon detecting this, the control circuitry configures the TDO/Enable pin as TDO and starts shifting the MISR output for a number of cycles as dictated by the length of the MISR. The hardware organization of such a scheme is illustrated in Figure 3.

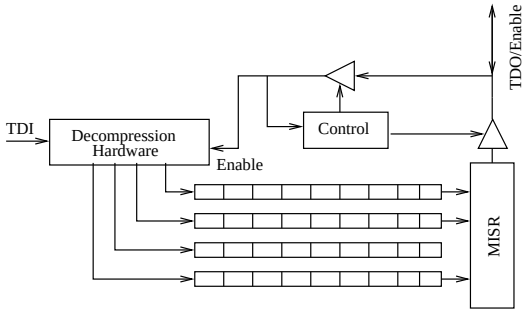

Figure 3. Overall hardware organization scheme of the proposed approach

Circuit	PIs	MinTest		Virtual Scan Chains		Golomb Coding		Proposed Approach		σ
		PC	SCC	PC	SCC	PC	SCC	PC	SCC	
s38584	1464	110	161,040	343	101,185	110	104,111	110	73,464	2.19
s38417	1664	68	113,152	547	154,254	68	92,054	68	45,003	2.51
s35932	1763	12	21,156	NA	NA	12	59,573	12	7,222	2.93
s15850	611	94	57,434	210	38,010	94	40,717	94	26,021	2.21
s13207	700	233	163,100	199	60,894	233	41,658	233	74,423	2.19

Table 7. Compressing the MinTest test vectors

Circuit	PIs	MinTest		Virtual Scan Chains		Proposed Approach			σ
		PC	SCC	PC	SCC	PC	SCC Before	SCC After	
s38584	1464	110	161,040	343	101,185	220	322,080	47,886	6.11
s38417	1664	68	113,152	547	154,254	231	384,384	55,848	6.88
s35932	1763	12	21,156	NA	NA	25	19,075	11,298	1.69
s15850	611	94	57,434	210	38,010	185	113,035	14,676	7.70
s13207	700	233	163,100	199	60,894	274	191,800	16,913	11.30

Table 8. Compressing incompletely specified test vectors

Circuit	PIs	Chain Concealment		Proposed Approach		σ
		PC	SCC	PC	SCC	
s38584	1464	203	38,976	186	22,636	1.72
s38417	1664	312	89,856	307	42,264	2.13
s35932	1763	33	7,128	29	3,972	1.79
s15850	611	178	22,784	176	10,798	2.11
s13207	700	264	25,344	259	15,783	1.61

Table 9. Results of proposed approach augmenting [1] on ISCAS'89 benchmarks

6 Experimental Results

After calculating the expected compression ratio in section 4, we now experimentally evaluate the achieved compression ratio for the larger circuits of the ISCAS'89 [2] benchmarks. For all the given experiments, it is required to calculate the optimal number of shift bits needed to flip the required positions. To achieve such a goal, we use a brute force enumeration algorithm [4] to pick the best tour, unless the number of bits to be flipped exceeds 10, in which case the greedy strategy previously discussed is used to tour the required states. The greedy strategy application may result in a slight reduction in compression compared to the one derived in the Section 4.

In the first series of experiments we set up our compression technique as the only compression hardware. The purpose of this experiment is to assess the compression ratio that could be attained by our approach as a stand-alone decompression technique. In this experiment, the scan chain is divided into 4 smaller scan chains, with its inputs driven from the decompression hardware. The inputs to the decompression network form the boundary scan chain driven by the Test Data Input (TDI) input pin. Figure 4 illustrates the organization of such hardware. We compress the test vectors generated by MinTest [6], and compare our results to MinTest, Virtual Scan Chains [9] and the results of applying Golomb Coding to MinTest [3]. Table 7 presents the experimental results. Columns 1 and 2 provide the circuit name and the number of primary inputs (PI) respectively. Columns 3 and 4 list the Pattern Count (PC) and Shift Clock Cycles (SCC) [6] needed to transfer the test data into the chip. Columns 5 and 6 list the PC and SCC in [9], while columns 7 and 8 list the results of [3]. Our results are presented in columns 9 and 10. Finally, column 11 lists the achieved compression ratio over MinTest.

In a second series of experiments, we compress incompletely specified test vectors. These vectors were obtained using Atalanta [10] and further compacted. These test vectors should provide an ideal case for our approach since unspecified bits in the test vectors dramatically reduce the number of bits to be flipped from one test slice to the other. However, as should be expected, the number of test vectors in this case should be slightly higher than the case of completely specified vectors. In this experiment, we compare our results against results of both [6] and [9] . A 4x16 decoder is used to derive the inputs of the scan chains. Table 8 gives the compression results. The compression ratio is calculated from the values of SCC before and after the compression.

In a third series of experiments, we augment existing decompression techniques with our decompression hardware.

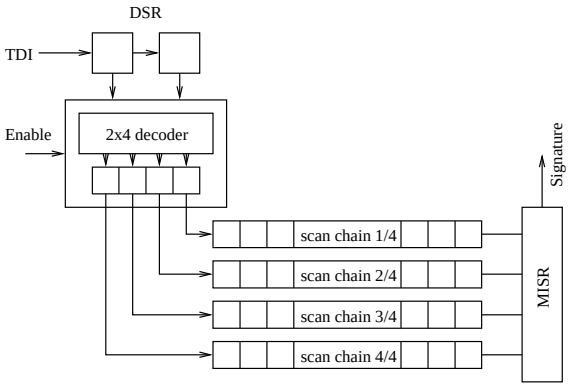

Figure 4. The proposed approach as the sole decompression hardware

We build our hardware as an extra layer of compression on top of the technique in [1]. We assume that the existing inputs to the decompression hardware form a boundary scan chain, and after adding our hardware, the DSR becomes the boundary scan chain driven by the TDI pin. Table 9 provides the comparative compression results for the proposed approach.

From the experimental results, we conclude that compressing incompletely specified vectors attains the best results. In order to utilize this, the core provider should provide the compressed test data to the system integrator, and integrate the decompression hardware to the core. This would make the compression scheme transparent to the system integrator.

The experimental results have clearly demonstrated that our technique results in a decrease in the number of bits shifted through the TDI input, and thus reduces the time needed to test the chip. Furthermore, we are able to achieve such a decrease with a very small overhead in hardware, i.e., 2x4, 4x16, 5x32 decoders for the experiments cited above.

7 Conclusions

As the complexity of chips keeps on increasing, the difficulty and amount of testing data needed for such chips keep on escalating. In this paper, we have proposed a new compression algorithm for reducing test data volume and time. The proposed method encodes the changes in the test vectors and utilizes decoders and shift registers to provide the necessary on-chip decompression. We have proposed the construction of the state transition diagram of the shift register in order to calculate the minimal number of shift bits needed to flip the required bits in the test slices. We have also analyzed the performance of the proposed technique, and proved that its use would be advantageous both as a stand-alone technique or as an augmentation to current compression techniques at the cost of an additional small amount of hardware. We have furthermore illustrated the effectiveness of the approach on the larger ISCAS-89 benchmark circuits. The results confirm the mathematical treatment. The low hardware overhead and the high compression ratios attained by the proposed approach establish it as an effective technique for test time reduction.

Acknowledgments

The first author would like to thank his colleague Ismet Bayraktaroglu for supplying the test vectors needed to carry out the second and third experiments.

References

[1] I. Bayraktaroglu and A. Orailoglu. Test volume and application time reduction through scan chain concealment. In *Proc. of Design Automation Conference*, pages 151–155, 2001.

[2] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Proc. of the International Symposium on Circuits and Systems*, pages 1923–1934, 1989.

[3] A. Chandra and K. Chakrabarty. Test data compression for system-on-a-chip using Golomb codes. In *Proc. of VLSI Test Symposium*, 2000.

[4] N. Dershowitz. A simplified loop-free algorithm for generating permutations. *BIT*, I5(1975):158–164.

[5] H. Fredricksen. A survey of full length nonlinear shift register cycle algorithms. *SIAM Review*, 24(2):195–221, 1982.

[6] I. Hamzaoglu and J. H. Patel. Test set compaction algorithms for combinational circuits. In *Proc. of International Test Conference*, pages 283–289, 1998.

[7] A. Jas, J. Gosh-Dastidar, and N. Touba. Scan vector compression/decompression using statisitical coding. In *Proc. of VLSI Test Symposium*, pages 25–29, 1999.

[8] A. Jas and N. Touba. Test vector decompression via cyclical scan chains and its application to testing core-based design. In *Proc. International Test Conference*, pages 458–464, 1998.

[9] A. Jas and N. Touba. Virtual scan chains: A means for reducing scan length in cores. In *Proc. of VLSI Test Symposium*, pages 73–78, 2000.

[10] H. K. Lee and D. S. Ha. On the generation of test patterns for combinational circuits. Technical report, Tech. Report No. 12-93, Department of Electrical Engineering, Virginia Tech.