Test Volume and Application Time Reduction Through Scan Chain Concealment *

Ismet Bayraktaroglu
Computer Science & Engineering Department
University of California, San Diego
La Jolla, CA 92093
ibayrakt@cs.ucsd.edu

Alex Orailoglu
Computer Science & Engineering Department
University of California, San Diego
La Jolla, CA 92093
alex@cs.ucsd.edu

ABSTRACT

A test pattern compression scheme is proposed in order to reduce test data volume and application time. The number of scan chains that can be supported by an ATE is significantly increased by utilizing an on-chip decompressor. The functionality of the ATE is kept intact by moving the decompression task to the circuit under test. While the number of virtual scan chains visible to the ATE is kept small, the number of internal scan chains driven by the decompressed pattern sequence can be significantly increased.

1. INTRODUCTION

As the complexity of today's circuits limits the practicability of sequential test generation to small circuits only, scan has been commonly used to ensure easy diagnosis and test generation times within practical limits. Alternative schemes, such as utilization of functional patterns, are still employed for detection of speed-related faults; however, fault grading of such schemes cannot be easily performed.

While scan helps keep test generation times within limits, the large number of scan cells and patterns generated, as a result, increase the test data volume and the tester time requirements inordinately. The increases in turn boost test application cost by necessitating prolonged utilization of increasingly expensive testers.

A number of schemes have been proposed for test data volume reduction of scan-based deterministic test by improving the effectiveness of test compaction [2, 4, 13, 14, 17] and compression schemes [7, 8, 9, 10]. While compaction schemes try to reduce the number of patterns generated without compromising fault coverage levels, compression schemes in turn target reduction of the storage requirements of the compacted test patterns. Additionally, shifting the decompression task to the circuit under test reduces test application times.

ATPG tools initially generate a test cube for a target fault with many bits left unspecified. A set of compactable test cubes are merged in compaction schemes. The unspecified bits are then randomly set to 0 or 1 to attain a fully specified test vector. Fault simulation is performed to drop the faults that are detected by the test vector. Though a number of schemes have been proposed to compact test vectors [2, 4, 13, 14, 17], either statically or dynamically, the resultant test sets still remain inordinately large.

While randomly setting the unspecified bits improves the effectiveness of fault dropping, it reduces the effectiveness of compression schemes as compression algorithms are known to perform poorly on random sequences. 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 together with cyclic registers [10] or statistical coding [7]. Test cube compression schemes, on the other hand, employ pseudo-random pattern generators in order to embed deterministic scan vectors in pseudo-random sequences. As embedding all ATPG patterns is not usually possible, a way to apply deterministic patterns directly is also included in such schemes. In a test-per-scan scheme, one of the multiple scan chains is made visible to the ATE and the remaining scan chains are driven from the LFSRs, whose initial states also contribute to the visible scan chain [9]. In test per clock architectures, as a deterministic pattern is loaded to the scan chain serially, PRPGs continue to supply a pattern per clock [8].

While compression techniques usually require modification to the access mechanism of the internal test hardware, alternative techniques have been proposed to include additional DFT so as to enable higher compaction of test vectors [6, 15].

As the sizes of the circuits increase, the pin to gate ratio reduces. Since the number of scan chains, limited by the number of I/O pins, cannot increase, the number of cells per scan chain has to increase, which directly affects test time. While simple MISR compaction schemes have been proposed to eliminate the need for observing the scan chain outputs, it can be nevertheless easily observed that the benefits achieved by MISR compaction are strictly limited as only a halving of test data volume can be thus achieved.

In this work, we propose a compression scheme for circuits with a large number of scan chains. For such circuits, the number of scan chains visible to the tester is reduced, thus providing test data volume and application time reductions. The compression scheme utilizes a linear mapping network, based either on LFSR sequences or on heuristics that we propose, in order to drive a high number of internal scan chains through a limited number of external pins. Experimental results indicate that test data volume can by reduced by up to a factor of 5, reducing test application times analogously.

Section 2 outlines the scheme proposed in this work. While section 3 describes the decompression hardware utilized in this work, section 4 explains the compression algorithm in detail. Experimental results and comparative studies in section 5 are followed by conclusions in section 6.

*The work of the first author is being supported by an IBM Graduate Fellowship.

DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA.

2. REDUCING VISIBLE SCAN CHAINS

In a scan-based deterministic test environment, a number of parameters, such as the number of scan chains and the scan frequency, can be varied in order to trade off among test application time, power consumption, and ATE cost. A method has been proposed in order to transfer compressed data to the chip under test at a slower speed and then uncompressing them in the chip at a faster speed, thus essentially reducing the visible scan chain length to the tester [7]. Such a scheme reduces test application time by increasing the actual speed of scan shifting with no need for faster ATEs. At the same time, tester memory requirements are also reduced. However, the increased speed of test application increases the power consumption of the chip under test. The speed under test usually has to be significantly lower than the operational speed as scan shifting results in significantly higher switching activity in the circuit.

An alternative solution for reducing test application time in a scan-based deterministic test environment with no increase in power consumption is increasing the number of scan chains. Doubling the number of scan chains reduces test application time and the power consumption nearly by half. However, the increased scan chain count increases pin count requirements of ATEs, resulting in higher cost. A solution to pin count requirements, by reducing the number of scan chains visible to the tester, is proposed by Jas *et al* [9]. In such a scheme, unspecified bits of the test cubes are utilized to encode the test cubes with an LFSR seed. A visible scan chain is split into N virtual scan chains internally and test patterns are provided to $N-1$ scan chains from the LFSRs and the remaining scan chain is provided directly from the ATE. Reductions in excess of 50% in the number of test cycles are reported to be thus achieved. However, the hardware overhead is significantly high as each virtual scan chain requires its own dedicated LFSR.

In this work, instead of dedicating a decoder to each scan chain separately, we introduce a single decoder to the inputs of the scan chains. A reduced number, determined by the compression ratio, of scan chains is seen by the ATE. Increasing the number of scan chains reduces the average length of the scan chains, thus reducing test application time, in addition to power consumption.

Figure 1 depicts the implementation we propose in this work. The combinational decompression network decodes N inputs provided by the ATE and produces M outputs to feed M scan chains. Test time and tester memory requirements are reduced by the ratio of M to N.

In this work, scan chain inputs (*SCI*) are compressed instead of the contents of individual scan chains. Scan chains are usually stitched after placement and therefore scan cells inside a scan chain

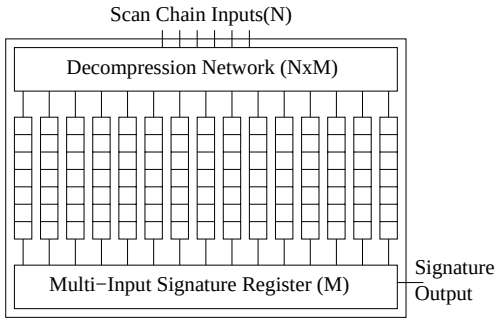


Figure 1: Hidden Scan Chains

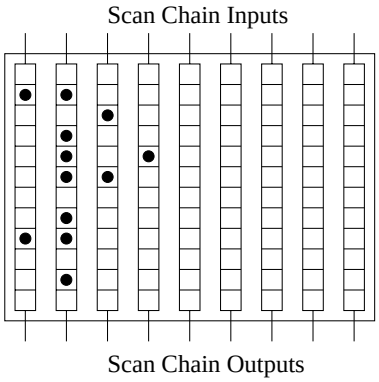


Figure 2: Fault Manifestation in Scan Chains

are usually driven by the same logic. A fault may manifest itself on a large number of scan cells in a scan chain; however, the fault can affect only a few scan chains, as illustrated in figure 2. The decompression scheme and its implementation is discussed in the next section.

3. DECOMPRESSION HARDWARE

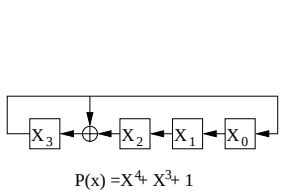
Test cubes with a limited number of specified bits have been shown to be encodable with an LFSR seed [5, 9]. Encoding test cubes with LFSR seeds has been utilized in various contexts, such as storing a number of deterministic patterns on-chip [5], or compressing deterministic test cubes [9]. All such schemes are employed to encode a test cube corresponding to a scan chain as noted previously. A test vector is generated by shifting the least significant bit of the LFSR through a scan chain starting from a precomputed seed.

In the proposed scheme, LFSR-based compression of the *SCI* can be achieved by shifting the LFSR for a number of cycles. However, such a process would necessitate the tester to wait idling for a number of cycles and would increase test application time! Instead, the LFSR sequence can be generated by using a simple XOR network. Figure 3 provides the 10-bit sequence generated by a 4-bit LFSR in terms of the initial state of the LFSR (X_3, X_2, X_1, X_0). A seed can be uncompressed in a single clock cycle by using such a simple XOR network, as depicted in figure 4, enabling a shift every clock cycle. Figure 5 provides an example, in which a test cube for 10 scan chains, each containing 11 scan cells, is encoded by 11×4 bits to be decompressed by the decompression network provided in figure 4.

While the test cubes in the example have been successfully compressed, linear dependencies at the output of the LFSR sequence may sometimes prevent successful compression, resulting in deteriorated fault coverages. The probability that s positions of a (2^n-1)-bit long sequence generated by an n-bit LFSR are linearly independent is given by the following formula [3]:

$$P_{2^n-1,s} = 1 - \prod_{i=0}^{s-1} \frac{2^n - 2^i}{2^n - i - 1} \quad (1)$$

While LFSR generated sequences usually have a low number of linear dependencies, short LFSR sequences may exhibit a high number linear dependencies [16]. Additionally, a $(N \times M)$ Decompression Network based on LFSR sequences in general requires



$$O_0 = X_3$$
$$O_1 = X_2 \oplus X_3$$
$$O_2 = X_1 \oplus X_2 \oplus X_3$$
$$O_3 = X_0 \oplus X_1 \oplus X_2 \oplus X_3$$
$$O_4 = X_0 \oplus X_1 \oplus X_2$$
$$O_5 = X_0 \oplus X_1 \oplus X_3$$
$$O_6 = X_0 \oplus X_2$$
$$O_7 = X_1 \oplus X_3$$
$$O_8 = X_0 \oplus X_2 \oplus X_3$$
$$O_9 = X_1 \oplus X_2$$

Figure 3: LFSR Sequence

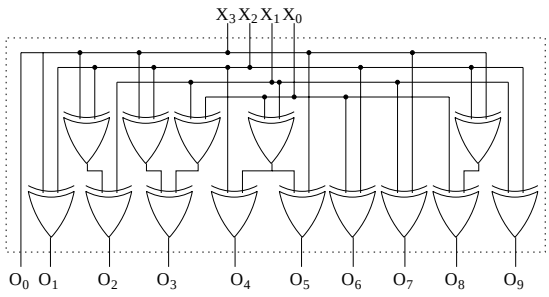


Figure 4: Decompression Hardware

$M \times (\frac{N}{2} - 1)$ XOR gates, a significant cost for large N. As we are not restricted to using only successive bits of the LFSR sequences in such an implementation, a search may be undertaken to select positions in the sequence to reduce the hardware requirement of the decompression network.

In this work we utilize a mapping function, instead of an LFSR, in which only a specific number of input signals are XORed together to generate the decompressed outputs. In general, the decompressor can be any linear mapping from N inputs to M outputs. We choose to utilize an XOR network in which every output is generated by XORing 3 inputs. In order to reduce the probability of linear dependencies, we enforce the constraint that any two output pairs have at most one input in common in their representation.

In a multiple scan chain environment, the number of shift operations per pattern depends on the maximum length-scan chain, denoted by *ML*. In the proposed approach, for each test pattern, *ML* 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

$O_0O_1O_2O_3O_4O_5O_6O_7O_8O_9$		$X_0X_1X_2X_3$
0 x 1 1 x x x x 1 x	→	0 0 1 0
x 1 x x 0 x 1 x x 0	→	0 1 1 0
x 0 x 0 0 x x 0 x x	→	0 0 0 0
x x 1 1 x x x 1 x 0	→	0 0 0 1
x x x x 0 x 0 x 0 1	→	1 0 1 0
x x 0 x 0 x 0 x x x	→	1 0 1 1
x x x 1 x 0 1 x 0 x	→	0 1 1 1
0 x 0 x 0 x 0 x x x	→	0 0 0 0
x x 0 x x x 1 1 1 x	→	0 1 1 0
x 1 1 x x 1 x x x 1	→	1 0 1 0
0 x x x 0 x x x 0 0	→	0 0 0 0

Figure 5: A Compression Example for 10 scan chains, each with 11 cells

Specified bits	32-bit LFSR	32-bits 3-input	34-bits 3-input	32-bits 4-input
16	99.82	98.87	99.50	99.81
18	99.31	97.80	98.68	99.57
20	99.60	96.17	97.42	99.04
22	99.15	91.39	94.05	97.09
24	97.99	77.99	84.79	92.17
26	90.88	54.69	67.22	75.26
28	69.68	26.71	40.24	42.59
30	28.14	7.09	16.48	10.17
32	4.43	0.68	4.48	0.00
Overhead (XOR Gates)	1706	256	256	384

Table 1: Performance of Decompression Networks for 128-bit Sequence

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 as discussed in the next section.

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 in table 1 indicate that the probability that a 128-bit sequence with at most 22 specified bits is representable with 32-bits is higher than 90% for all networks. 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% lower hardware overhead, performs similarly to the LFSR-sequence based network. The performance of a 3-input network, on the other hand, can be improved by increasing the number of bits available for encoding. The results in table 1 indicate that increasing the number of encoded inputs from 32 to 34 improves the performance of the 3-input network to the level of the 4-input network, at the cost of a 5.9% reduction in the compression ratio. As the performance of 3-input networks is usually acceptable, we utilize them in our experiments.

4. COMPRESSION ALGORITHM

In this section, we discuss the proposed compression algorithm. The ability of the compression scheme to encode the test cubes stems from the limited number of specified bits. Therefore, the compression algorithm needs to be incorporated into the compaction algorithm in order to guarantee that the compacted test cubes would be encodable. In this work, a variant of the dynamic compaction algorithm proposed by Ayari *et al* [1] is employed. 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 test set can possibly change at any time, the unspecified bits cannot be

set randomly for additional fault dropping. In order to enable utilization of randomized test patterns, we reverse the working order of the algorithm. The resultant algorithm is given below.

1. Perform fault simulations with a number of random patterns and remove detected faults from the fault list.
2. Select a fault and generate a seed test cube (*STC*) for the fault, if not available.
3. For all faults in the fault list,
   (a) Generate a test cube (or use already generated test cube).
   (b) If compatible with *STC*, merge them.
   (c) Perform fault simulation and remove detected faults from the fault list.
4. Randomize the unspecified bits of the resultant test cube.
5. Perform fault simulation and drop all detected faults from the fault list.
6. If undetected faults remain, go to step 2.

While the above algorithm without the final fault dropping step (5) performs similarly to the algorithm suggested by Ayari *et al* [1], the additional fault dropping step increases its effectiveness. Furthermore, a random pattern utilization step (1) reduces the number of faults that need to be handled by the compression algorithm.

In order to incorporate the suggested compression algorithm into the compaction procedure, steps 3 and 4 of the above algorithm need to be augmented. After a compatibility check of step 3, 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 has been shown to be quite small in our experiments. In step 4 of the algorithm, the unspecified bits are set by simulating the decompression network, resulting in a pseudo-random fill.

While step 3c of the algorithm enables dropping some faults, the simulations we have conducted indicate that it does not improve compaction and that at the same time it reduces only slightly the number of patterns that have to be generated, yet at the expense of introducing a high computational complexity due to repeated applications of the fault simulation procedure. Therefore, in this work, step 3c is skipped. Additional improvements can be attained in computational complexity by limiting the search in step 3 of the algorithm, for example, by stopping the mergeability analysis after a certain specified bit ratio is achieved or after a number of times the merge operation has failed.

The compression algorithm is built on top of the Atalanta [11] ATPG tool; Hope [12] is utilized for fault simulation purposes. A C program is developed in order to implement the combined Compaction/Compression algorithm. A fault is selected from the fault list and a test cube is generated for the fault by Atalanta. Encodability of the pattern is checked by setting up a set of linear equations. In case the pattern cannot be encoded, either the number of inputs, N, needs to be increased or the mapping network needs to be changed. However, a simple optimistic solution consists of skipping the fault and hoping that the fault will be gratuitously detected during the fault dropping step of the algorithm. In case some faults remain undetected after the completion of the algorithm, the remaining faults need to be handled separately. A possible solution may be the incorporation of reconfigurability into the mapping hardware in that case.

The efficiency of the compression algorithm can also be improved by utilizing the *Maximal Compaction* heuristic proposed by Pomeranz *et al* [13], which aims at increasing the number of unspecified bits in a test cube without losing the ability to test the target fault of the test cube. Incorporation of more efficient dynamic test compaction algorithms together with fault ordering schemes proposed in the literature may help reduce the number of patterns with no significant impact on the compression ratio attainable by the proposed scheme.

5. RESULTS

The scheme proposed in this work mainly targets large circuits; therefore, we perform experiments on the larger ISCAS89 benchmark circuits. For each of these benchmark circuits, initially, we apply the compaction algorithm that is described in this work in order to establish a baseline for our comparisons. In order to reduce the computational complexity of the algorithm, we perform compaction experiments that utilize a number of random patterns to eliminate the easy to detect faults from the initial fault list. The resultant number of test patterns and the number of test cubes generated, i.e., the number of faults undetected by the random patterns, are reported in table 2. Increasing the number of random patterns, denoted by the number following the letter "R" in the column headers, increases the total pattern count though it decreases the computational complexity; the number of test cubes generated is reduced in the case of increased number of random patterns.

The number of test patterns with compression is also reported in table 2 for several combinations of N and M. The results indicate that quite high compaction ratios are possible. In all but two cases, 100% of the irredundant faults are detected. For the two remaining cases, fault detection loss is less than 0.2%. The compression ratio reported is the best compression attained for the particular circuits when 100% fault detection is achieved. It is calculated as:

$$1 - \frac{N}{M}\frac{\#\ of\ Patterns\ with\ Compression}{\#\ of\ Patterns\ without\ Compression} \qquad (2)$$

Experimental results reported in table 2 indicate that compression levels in excess of 75% are achievable. While the compression ratio could have 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 a comparison of the results to previously published data, as shown in table 3. The first two results are taken from the compaction work, *COMPACTEST* [13] and *MinTest* [4], and the third result is taken from a recent compression scheme, *Virtual Scan Chains* [9]. 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 achieved for a particular circuit in all three schemes. For the *Virtual Scan Chains* scheme, the length of the scan chains is also provided. For the proposed scheme, the number of bits required for encoding a single shift data and the maximum scan chain length is provided. The results indicate that the proposed scheme, even with no tuning of its parameters, is capable of providing a high compression ratio. For one of the test cases though, the compression ratio compared to the highly compacted test patterns of *MinTest* is significantly lower. The reduced compression ratio in that case is not due to the failure of our scheme, but rather to the highly specified test cubes for that particular circuit. In general, however, test compaction algorithms typically leave considerable space for further compression of the test patterns as seen in table 3.

	Generated Test Cubes			Patterns Without Compression			Patterns With Compression									
							N=32, M=200			N=24, M=128			N=24, M=200			
Circuit	R0	R10	R20	R0	R10	R20	R0	R10	R20	R0	R10	R20	R0	R10	R20	Compression
s38584	34797	12807	10287	177	182	191	185	190	193	185	186	189	203	209	216	86.4%
s38417	31015	11966	9546	188	200	194	312	313	315	280	296	300	364*	375*	367*	75.0%
s35932	35110	17601	4343	22	28	32	30	33	35	30	32	38	33	36	39	85.4%
s15850	11336	4634	3910	155	161	176	178	179	188	181	183	194	201*	202*	203*	82.9%
s13207	9664	4116	3677	253	262	271	257	265	275	260	259	266	264	266	276	87.8%

* *100% fault detection is not achieved*

Table 2: Test Pattern Compression and its Impact on Compaction

		COMPACTEST		MinTest		Virtual Scan Chains			Proposed				
Circuit	PIs	PC	TV	PC	TV	PC	SCL	TV	PC	MSCL	EL	TV	Improvement
s38584	1464	125	183,000	89	130,296	343	295	**101,185**	203	8	24	38,976	61.5%
s38417	1664	95	158,080	62	**103,168**	547	282	154,254	312	9	32	89,856	12.9%
s35932	1763	13	22,919	9	**15,867**	NA	NA	NA	33	9	24	7,128	55.1%
s15850	611	123	75,153	91	55,601	210	181	**38,010**	178	4	32	22,784	40.1%
s13207	700	237	165,900	233	163,100	199	306	**60,894**	264	4	24	25,344	58.4%

PI: Primary Input, PC: Pattern Count, TV: Test Volume (number of bits), SCL: Scan Chain Length, MSCL: Maximal SCL, EL: Encoded Length

Table 3: Test Application Time Improvements with the Proposed Compression Scheme

6. CONCLUSION

A test pattern compression scheme for large designs that necessitate a high number of scan chains is proposed. The number of visible scan chains is reduced through 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, power consumption and test application times are proportionally reduced due to the unhindered increase in the number of scan chains.

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, the proposed scheme can easily be automated without impacting the design process, thus achieving significant cost reductions in manufacturing test.

7. REFERENCES

[1] B. Ayari and B. Kaminska. A new dynamic test vector compaction for automatic test pattern generation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(3):353–358, March 1994.

[2] S. Bhatia and P. Varma. Test compaction in a parallel access scan environment. In *Asian Test Symposium*, pages 300–305, November 1997.

[3] C. L. Chen. Linear dependencies in linear feedback shift registers. *IEEE Transactions on Computers*, 35(12):1986, December 1986.

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

[5] 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 Transactions on Computers*, 44(2):223–233, February 1995.

[6] Y. Higami, S. Kajihara, and K. Kinoshita. Test sequence compaction by reduced scan shift and retiming. In *Asian Test Symposium*, pages 169–175, November 1995.

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

[8] A. Jas, K. Mohanram, and N. A. Touba. An embedded core DFT scheme to obtain highly compressed test sets. In *Asian Test Symposium*, pages 275–280, November 1999.

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

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

[11] H. K. Lee and D. S. Ha. On the generation of test patterns for combinational circuits. Technical Report 12_93, Department of Electrical Eng., Virginia Polytechnic Institute and State University.

[12] H. K. Lee and D. S. Ha. HOPE: An efficient parallel fault simulator. In *IEEE Design Automation Conference*, pages 336–340, June 1992.

[13] I. Pomeranz, L. Reddy, and S. Reddy. COMPACTEST: a method to compact test sets for combinational circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(7):1040–1049, July 1993.

[14] I. Pomeranz and S. M. Reddy. Static test compaction for scan-based designs to reduce test application time. In *Seventh Asian Test Symposium*, pages 198–203, December 1998.

[15] I. Pomeranz and S. M. Reddy. PASTA: partial scan to enhance test compaction. In *Great Lakes Symposium on VLSI*, pages 4–7, March 1999.

[16] J. Rajski and J. Tyszer. On linear dependencies in subspaces of LFSR-generated sequences. *IEEE Transactions on Computers*, 45(10):1996, October 1996.

[17] R. Sankaralingam, R. Oruganti, and N. A. Touba. Static compaction techniques to control scan vector power dissipation. In *VLSI Test Symposium*, pages 35–40, 2000.