Test Vector Decompression via Cyclical Scan Chains and Its Application to Testing Core-Based Designs

Abhijit Jas and Nur A. Touba

Computer Engineering Research Center
Department of Electrical and Computer Engineering
University of Texas, Austin, TX 78712-1084
E-mail: {jas, touba}@ece.utexas.edu

Abstract

A novel test vector compression/decompression technique is proposed for reducing the amount of test data that must be stored on a tester and transferred to each core when testing a core-based design. A small amount of on-chip circuitry is used to reduce both the test storage and test time required for testing a core-based design. The fully specified test vectors provided by the core vendor are stored in compressed form in the tester memory and transferred to the chip where they are decompressed and applied to the core (the compression is lossless). Instead of having to transfer each entire test vector from the tester to the core, a smaller amount of compressed data is transferred instead. This reduces the amount of test data that must be stored on the tester and hence reduces the total amount of test time required for transferring the data with a given test data bandwidth.

1. Introduction

Testing systems-on-a-chip containing multiple cores is a major challenge due to limited test access to each core [Chandramouli 96], [Zorian 97]. The test vectors for each core must be applied to the core's inputs and internal scan, and the test response of the core must be observed at the core's outputs and shifted out of its internal scan. Some means for getting the test data from the tester to each core and getting the test response from each core to the tester is required. The best possible situation is to have full parallel access to the inputs and outputs of the cores [Immaneni 90]. However, this requires multiplexing all of the core I/Os to the chip pins. The routing complexity and overhead for this can be enormous. A more efficient means for providing test access to the cores is to use scan chains. The number of scan chains that are used and the way in which they are organized determines the *test data bandwidth* for each core (i.e., rate at which test vectors can be scanned in and test response scanned out). The number of scan chains and their organization typically depend on the capabilities of the tester being used and on the scan routing costs. The total test time required for testing a core-based design depends on the amount of test data that must be transferred between the tester and the chip and the test data bandwidth for transferring the data.

Figure 1 shows a general block diagram for how test data is transferred from the tester to the cores. The amount of test data that must be transferred from the tester to a particular core is equal to the number of test vectors (T) for the core times the number of input bits and internal scan elements for the core (m), i.e., $T \times m$. For systems-on-a-chip that contain many complex cores, both the amount of test data and the test time can become *very large.*

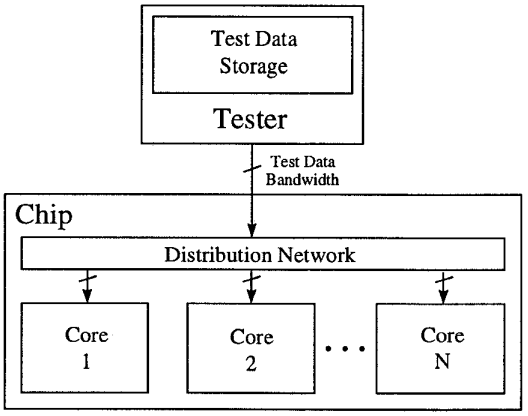

Figure 1. Block Diagram for Transferring Test Data between Tester and Embedded Cores

One solution to this problem is to use built-in self-test (BIST) where on-chip hardware is used to test the cores. However for logic cores, this is only practical if the core is made "BISTable" by the core vendor. Currently, there are few cores that include BIST features. Usually, only a set

0-7803-5092-8/98 $10.00

of test vectors for the core is given. The amount of BIST hardware required to apply a large set of specified test vectors is generally prohibitive.

This paper presents an efficient compression/ decompression scheme to reduce the amount of test data that must be stored on the tester and transferred to a core. A small amount of on-chip circuitry is used to reduce both the test storage and test time required for testing a core-based design. The fully specified test vectors provided by the core vendor are stored in compressed form in the tester memory and transferred to the chip where they are decompressed and applied to the core (the compression is lossless). Instead of having to transfer each entire test vector from the tester to the core, a smaller amount of compressed data is transferred instead. This reduces the amount of test data that must be stored on the tester and hence reduces the total amount of test time required for transferring the data with a given test data bandwidth. Thus, the technique presented in this paper can be used to reduce the test time required for testing a system-on-a-chip given a tester's limited memory and channel capacity.

Test vector compression/decompression techniques can be classified based on the amount of information they require. Four general classifications are described below:

Schemes Requiring ATPG - These are schemes that involve using special ATPG procedures in generating the test set. This includes techniques that try to compact test sets [Tromp 91], [Pomeranz 93], [Kajihara 93], or to find easy to encode test vectors [Reeb 96], [Hellebrand 95a].

Schemes Requiring Fault Simulation - These are schemes that do not decompress a particular test set, but rather use pseudo-random generators (e.g., LFSR's) to apply a large number of vectors to detect most of the faults, thereby reducing the number of deterministic test vectors that are required. These techniques require fault simulation of the circuit-under-test (CUT) to verify fault coverage.

Schemes Requiring Test Cubes - These are schemes that compress test cubes, which are ATPG generated vectors in which the unspecified inputs are left as don't cares. These schemes include LFSR reseeding [Koenemann 91], [Hellebrand 95b], [Zacharia 96], and width compression [Chakrabarty 97].

Schemes for Fully Specified Test Vectors - These are schemes that are able to compress fully specified test vectors. These schemes were developed for compressing test vectors stored in on-chip ROM's [Agarwal 81], [Aboulhamid 83], [Dandapani 84], [Edirisooriya 92], [Dufaza 93], [Iyengar 98].

For intellectual property cores where no information is given about the internal structure of the core, test vector compression/decompression techniques that require either ATPG or fault simulation cannot be used. The core integrator must test the cores with the set of test vectors given by the core vendor. Furthermore, in most cases, the test vectors that are given are fully specified. Thus, techniques which require test cubes also cannot be used. For this reason, compression/decompression techniques for fully specified test vectors are needed. Previous work in this area has been focused on reducing the size of an on-chip ROM needed to store the test vectors.

The cyclical scan chain decompression technique described in this paper can be used for fully specified test vectors and thus is applicable for intellectual property cores. It requires very little additional hardware. Rather it takes advantage of the fact that existing scan chains on the chip can be configured as cyclical decompressors.

A test data compression/decompression scheme for reducing the time for downloading test data from a workstation to a tester has recently been proposed by Yamaguchi, *et al.*, [Yamaguchi 97], [Ishida 98]. Note that this is a software based approach which targets a different problem then the one addressed here. It would be too complex and slow for an on-chip implementation as described here.

The paper is organized as follows: The basic idea of cyclical scan chain decompression is explained in Sec. 2. Section 3 describes how the tester transfers encoded data to cyclical scan chain decompressors. Section 4 discusses ways in which cyclical scan chain decompression can be implemented in systems-on-a-chip containing many cores. Experimental results indicating the amount of compression that can be achieved are shown in Sec. 5. Section 6 is a conclusion.

2. Cyclical Scan Chain Decompression

The section describes the basic idea of test vector decompression via cyclical scan chains. Practical issues on how to implement it in a core-based design will be described in subsequent sections. Cyclical scan chain decompression involves the use of two scan chains as shown in Fig. 2. One is the "test scan chain" where the test vector will be applied to the circuit-under-test (CUT), and the other is the "cyclical scan chain" where the decompression will take place. The serial output of the cyclical scan chain feeds the serial input of the test scan chain and also loops back and is XORed in with the serial input of the cyclical scan chain. There are two requirements for the cyclical scan chain:

1. It must have the same number of scan elements as the test scan chain.
2. Its contents must not be overwritten when the system clock for the CUT is applied.

When the system clock for CUT is applied, the test vector in the test scan chain is applied to the CUT and its

response is loaded back into the test scan chain. However, the contents of the cyclical scan chain must not be overwritten. The cyclical scan chain can be configured using the chip boundary scan, or using the boundary scan around a core, or using a scan chain in a different system clock domain. Note that if the test scan chain is a boundary scan that is driving the primary inputs of the CUT and is not capturing test response, then its contents are not lost when the system clock is applied and thus it can act as its own cyclical scan chain. This is illustrated in Fig. 3.

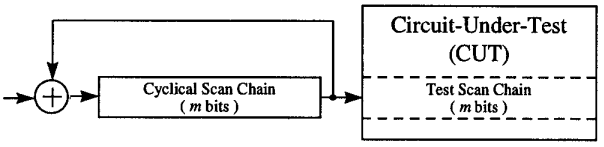

Figure 2. Cyclical Scan Chain Decompression Architecture

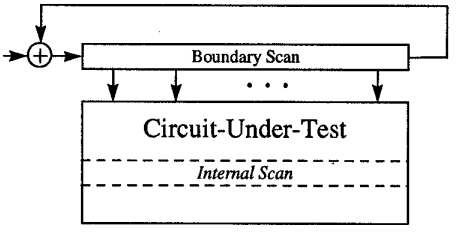

Figure 3. Cyclical Scan Chain Decompression Using Boundary Scan

The cyclical scan chain has the property that if it contains test vector t, then the next test vector that is generated in the cyclical scan chain will be the XOR of t and the "difference vector" that is shifted in. So generating a test set consisting of n test vectors, t_1, t_2,..., t_n, in a cyclical scan chain would involve first initializing the scan chain to all 0's, and then shifting t_1 into the scan chain followed by the difference vector $t_1 \oplus t_2$, followed by $t_2 \oplus t_3$, and so on up to $t_{n-1} \oplus t_n$.

The difference vectors that need to be shifted in to the cyclical scan chain depend on the way in which the test set is ordered. By carefully ordering the test vectors in the test set, the number of 0's in the difference vectors can be maximized. Test vectors tend to be very correlated. Faults in the CUT that are structurally related require similar input value assignments in order to be provoked and sensitized to an output. Thus, many pairs of test vectors in the test set will have similar input combinations such that the difference vectors have a lot of 0's. Ordering the test vectors in the test set such that correlated test vectors follow each other results in difference vectors with many more 0's than 1's.

Data which is skewed such that the probability of one value exceeds that of another can be efficiently compressed with a run-length code. An example of a *variable-to-block* run-length code is shown in Fig. 4. A variable number of bits is encoded by a fixed number of bits. In this example, the fixed number of bits is 3. If a difference vector was 0000010000001100001, then the encoded vector would be 101 110 000 100.

Note that if the last few bits at the very end of the *difference vector bit stream* (all difference vectors concatenated together) cannot be encoded, extra bits can be added to solve the problem. For example, if the last two bits were 00, then the codeword 111 could be used to encode the bits even though it generates 000000. The extra bits would simply be ignored.

000	1
001	01
010	001
011	0001
100	00001
101	000001
110	0000001
111	0000000

Figure 4. Example of Run-Length Code

The hardware required for decompressing an encoded vector for a run-length code is very simple. Each three bit block of encoded data is just a count of the number of 0's in the run, so a three bit counter can be used to decompress the data. The counter is loaded with a three bit block and counts down to zero. When it reaches a count of zero, it outputs a 1 (unless the initial state was 111) and then is reloaded with the next three bit block.

Ordering the test set to minimize the run-length encoding corresponds to forming a complete weighted graph and finding the minimum cost Hamiltonian path. Each node in the graph corresponds to a test vector and is connected by a weighted edge to every other node. The weight on the edge between two nodes is computed by forming the difference vector between the two corresponding test vectors and computing the number of bits needed to encode the difference vector. The minimum cost path through the graph that does not repeat any vectors corresponds to the optimum ordering of the test vectors to maximize the compression. Many efficient heuristic procedures for finding a good ordering exist.

So the basic idea of cyclical scan chain decompression can be summarized as follows. Given a test set that needs to be applied to a CUT, a cyclical scan chain is formed where the number of stages is equal to the number of bits in the test vectors. The test vectors are then ordered to

minimize the run-length encoding of the difference vectors. Rather than storing the full test vectors themselves, the compressed difference vectors (encoded with the run-length code) can be stored instead. To test the CUT, the compressed difference vectors are shifted in to a run-length decoder which decompresses them into the original difference vector bit stream one bit at a time which is fed into the cyclical scan chain to generate the test vectors.

This is the basic theory of cyclical scan chain decompression. There are many ways in which it can be implemented and used in different applications. The remainder of this paper will focus on its application for testing core-based designs. There are a number of practical issues related to how the tester transfers encoded data to cyclical scan chain decompressors, and how a core-based design can be configured during testing to allow cyclical scan chain decompression.

3. Transferring Data to Cyclical Scan Chain Decompressors

Because a variable length code is used, the number of encoded bits transferred to the run-length decoder is less than the number of decoded bits that are transferred out. Since the run-length decoder shifts only one bit of data into the scan chain each clock cycle, there will be clock cycles when it will not be ready to receive data from the tester. These clock cycles can be overlapped with the clock cycles required to transfer data to another cyclical scan chain decompressor in order to reduce test time. One way to accomplish this is to use a single channel from the tester to transfer encoded data to multiple cyclical scan chain decompressors.

Consider the simplest case where a single tester channel is used to transfer data to two cyclical scan chain decompressors using the three bit code shown in Fig. 5. The code in Fig. 5 is a slight modification to the run-length code shown in Fig. 4 which provides some useful advantages. In our experiments, we found that in most cases it allows more compression because it is not as inefficient when runs of 1's occur. The other major advantage is that it takes no more than 6 clock cycles to decompress each encoded block of 3 bits. Thus, a single tester channel can shift in 6 bits of encoded data, 3 bits for each decompressor, and then load both decompressors at once (this is illustrated in Fig. 6). The decompressors can then start decompressing the encoded data while the tester takes 6 cycles to shift in the next 6 bits of encoded data. By the time the tester is ready to load the next block of encoded data, the decompressors are guaranteed to be finished decoding the previous block of encoded data. While the code in Fig. 5 is slightly more complicated to decode than the one in Fig. 4, it can still be decoded by a small finite state machine (FSM).

000	10
001	11
010	01
011	001
100	0001
101	00001
110	000001
111	000000

Figure 5. Modified 3-Bit Run-Length Code

When the decompressor is loaded with a 3 bit block of encoded data, it generates the appropriate sequence of decoded bits and advances the cyclical scan chain and test scan chain for each bit of the sequence. For the code in Fig. 5, the length of the decoded sequence varies from 2 to 6 bits. A scan counter is used to count the number of bits that are shifted into the test scan chain. When the test scan chain is full, the system clock is activated to apply the test vector to the CUT.

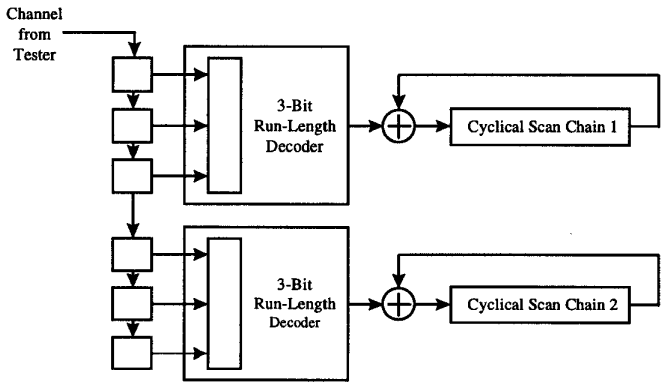

Figure 6. Tester Shifts in 6 Bits of Encoded Data and Loads Two Run-Length Decoders

This approach has a number of attractive features. One tester channel is used to load two scan chains through the decompressors. The test program is simple. The tester just shifts in 6 bits of encoded data and applies a control signal to load the decompressors. The decompressors and related control circuitry are very simple. In effect, the decompressors allow the tester to load two scan chains with compressed test vectors in close to the time normally required to load one scan chain with uncompressed test vectors. This increases the effective bandwidth of a single tester channel *and* reduces the amount of data that needs to be stored in tester memory.

4. Application to Testing Core-Based Designs

Cyclical scan chain decompression can be used for testing core-based designs. No knowledge of the internal structure of the cores is required. The test vectors given by core vendors can be encoded and stored on the tester and then decompressed with cyclical scan chains on-chip. There are typically many different scan chains in a core-based design. Each core may have an internal scan as well as a boundary scan collar, the user-defined logic (UDL) may contain scan chains, and the chip may have a boundary scan around its pins. The length of the internal scan chains in the cores cannot be changed, but the other scan chains are designed by the core integrator and thus can be configured in different ways.

The requirement for cyclical scan chain decompression as described in Sec. 2 is that a cyclical scan chain of equal length to the test scan chain is needed and the cyclical scan chain must not lose its contents during the test session. The simplest case for using cyclical scan chain decompression is for the boundary scan around the cores since the boundary scan can be configured as its own cyclical scan chain (assuming that it is not simultaneously used to capture the response of the logic surrounding the core). Using scan chain decompression in the internal scan of the core requires a separate cyclical scan chain of equal length. This cyclical scan chain can be configured in many different ways. Perhaps the simplest way is to use the boundary scan around the chip pins if one exists. If the chip boundary scan is longer than the internal scan of the core, it can of course be looped back at an intermediate point to form a cycle of the necessary length. In the same manner, the boundary scan around another core could also be used (as illustrated in Fig. 7). The internal scan in a different core whose system clock can be controlled independently from the core-under-test can also be configured as part of the cyclical chain provided it is the same length or shorter than the test scan chain. If it is shorter, than it can be configured with other boundary scan elements or scan elements in the UDL to form the correct length chain (as illustrated in Fig. 8).

There are many options for configuring the cyclical scan chain decompressors. There is a lot of flexibility for developing a test schedule that tests all the cores in a system-on-a-chip using cyclical scan chain decompression to maximize the test data bandwidth of the tester. The run-length decoders and scan counters can be reused in different test sessions for different cyclical scan chain configurations. The output of a run-length decoder need not directly connect to the cyclical scan chain, there can be any number of scan chain elements between the XOR at the input of the cyclical scan and the run-length decoder.

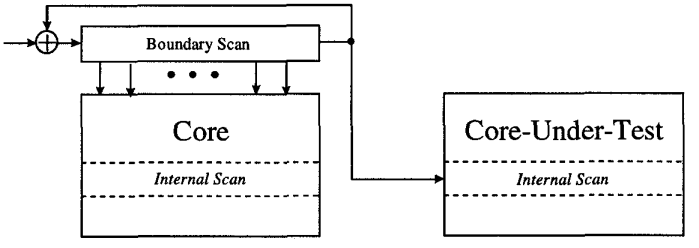

Figure 7. Configuring Boundary Scan as Cyclical Scan Chain for Internal Scan in Core

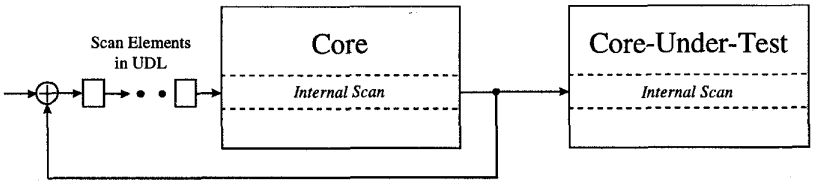

Figure 8. Using Internal Scan of a Core Plus Scan Elements in the UDL to Form Cyclical Scan Chain for the Internal Scan of the Core-Under-Test

5. Experimental Results

Experiments were performed for the ISCAS 85 [Brglez 85] and large ISCAS 89 [Brglez 89] circuits. The test set for each circuit was ordered to minimize the run-length encoding of the difference vectors, and then was encoded. Two different codes were tried. Code 1 is the 2-bit code shown in Fig. 9. Code 2 is the 3-bit code shown in Fig. 5. Table 1 shows the size of the scan chain for each circuit (for the ISCAS 85 circuits, it is assumed that their primary inputs are controlled by a scan chain). The original amount of test data is shown followed by the amount of compressed data for each code. The percentage of compression is computed as:

(Original Bits - Compressed Bits)/(Original Bits)

As can be seen, the 3-bit code provided better compression than the 2-bit code for most circuits. Results for codes having more than 3 bits were found to have much worse compression and thus are not shown.

When the test vectors are ordered, we noticed that some of the difference vectors require many fewer bits to encode with a run-length code than if they were not encoded while others required more bits to encode with a run-length code than if they were not encoded. Some test vectors are not very correlated with the others. Thus, one option for improving the compression would be to use the cyclical scan chain decompressor to generate all the test vectors that it is efficient for, and then shift in the remaining test vectors normally. We tried this for the large ISCAS 89 circuits and the results are shown in Table 2. On average, it increased the amount of compression by about 5%.

00	1
01	01
10	001
11	000

Figure 9. 2-Bit Run-Length Code

Note that the compression that is achieved is a two-fold advantage. Not only does it result in less test storage requirements, but it also results in less test time which translates directly into lower test costs. The amount of compression could be much greater if test cubes were compressed instead of fully specified test vectors.

Table 1. Compression Results for Compressing Whole Test Set

Circuit			Compressed Test Data			
Name	Scan Size	Test Data (bits)	Code 1 (2-bit)	Percent Compression	Code 2 (3-bit)	Percent Compression
c432	36	2124	1726	18.8%	1608	24.3%
c499	41	2378	1716	27.9%	1449	39.1%
c880	60	4800	4076	15.1%	3930	18.2%
c1355	41	4223	3268	22.7%	2910	31.1%
c1908	33	4290	3392	21.0%	3159	26.4%
c2670	233	30989	25748	17.0%	24405	22.3%
c3540	50	9900	8180	17.4%	7752	21.7%
c5315	178	38092	31932	16.2%	30501	20.0%
c6288	32	1152	974	15.5%	912	20.9%
c7552	207	56097	45638	18.7%	42528	24.2%
s9234	247	118313	100998	14.7%	98043	17.2%
s13207	700	372400	334904	10.1%	330975	11.2%
s15850	611	310388	277776	10.6%	274377	11.7%
s38417	1664	1995136	1786038	10.5%	1759206	11.9%

Table 2. Compression Results with Turning Off Compression for Part of Test Set

Circuit			Compressed Test Data	
Name	Scan Size	Test Data (bits)	Code 2 (3-bit)	Percent Compression
s9234	247	118313	96008	18.9%
s13207	700	372400	313666	15.8%
s15850	611	310388	260532	16.1%
s38417	1664	1995136	1673680	16.2%

6. Conclusions

This paper presents a new approach for compression/decompression of test vectors. Several key ideas are proposed:

1. Using existing scan chains on the chip to from a cycle to generate test vectors by shifting in a difference vector.
2. Ordering the test vectors in the test set to maximize the 0's in the difference vectors.
3. Encoding the difference vectors with a run-length code.
4. Using a single channel from the tester to load multiple run-length decoders which in effect increases the "effective" bandwidth of the channel.
5. Configuring scan chains in a core-based design to act as decompressors for other scan chains.

These ideas lay a ground work for further advancement in test vector compression/decompression. Some areas for further research would be to study the use of variable-to-variable length encoding for the difference vectors as well as other codes, and to look at ways to combine cyclical scan chain decompression with BIST.

Acknowledgements

This material is based on work supported in part by the Defense Advanced Research Projects Agency (DARPA) under Contract No. DABT63-94-C-0045, in part by the National Science Foundation under Grant No. MIP-9702236, and in part by the Texas Advanced Research Program under Grant No. 1997-003658-369.

References

[Aboulhamid 83] Aboulhamid, M.E., and E. Cerny, "A Class of Test Generators for Built-In Testing," *IEEE Transactions on Computers* , Vol. C-32, No. 10, pp. 957-959, Oct. 1983.

[Agarwal 81] Agarwal, V.K., and E. Cerny, "Store and Generate Built-In Testing Approach," *Proc. of FTCS-11*, pp. 35-40, 1981.

[Brglez 85] Brglez, F., and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortan," *Proc. of International Symposium on Circuits and Systems*, pp. 663-698, 1985.

[Brglez 89] Brglez, F., D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," *Proc. of International Symposium on Circuits and Systems*, pp. 1929-1934, 1989.

[Chakrabarty 97] Chakrabarty, K., B.T. Murray, J. Liu, and M. Zhu, "Test Width Compression for Built-In Self-Testing," *Proc. of International Test Conference*, pp. 328-337, 1997.

[Chandramouli 96] Chandramouli, R., and S. Pateras, "Testing Systems on a Chip," *IEEE Spectrum*, pp. 42-47, Nov. 1996.

[Dandapani 84] Dandapani, R., J. Patel, and J. Abraham, "Design of Test Pattern Generators for Built-In Test," *Proc. of International Test Conference*, pp. 315-319, 1984.

[Dufaza 93] Dufaza, C., C. Chevalier, and L.F.C. Lew Yan Voon, "LFSROM: A Hardware Test Pattern Generator for Deterministic ISCAS85 Test Sets," *Proc. of Asian Test Symposium*, pp. 160-165, 1993.

[Edirisooriya 92] Edirisooriya, G., and J.P. Robinson, "Design of Low Cost ROM Based Test Generators," *Proc. of VLSI Test Symposium*, pp. 61-66, 1992.

[Hellebrand 95a] Hellebrand, S., B. Reeb, S. Tarnick, and H.-J. Wunderlich, "Pattern Generation for a Deterministic BIST Scheme," *Proc. of International Conference on Computer-Aided Design (ICCAD)*, pp. 88-94, 1995.

[Hellebrand 95b] Hellebrand, S., 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*, Vol. 44, No. 2, pp. 223-233, Feb. 1995.

[Immaneni 90] Immaneni, V., and S. Raman, "Direct Access Test Scheme - Design of Block and Core Cells for Embedded ASICS," *Proc. of Int. Test Conference*, pp. 488-492, 1990.

[Ishida 98] Ishida, M., D.S. Ha, T. Yamaguchi, "COMPACT: A Hybrid Method for Compressing Test Data," *Proc. of VLSI Test Symposium*, pp. 62-69, 1998.

[Iyengar 98] Iyengar, V., K. Chakrabarty, and B. T. Murray, "Built-in Self Testing of Sequential Circuits Using Precomputed Test Sets," *Proc. of VLSI Test Symposium*, pp. 418-423, 1998.

[Kajihara 93] Kajihara, S., I. Pomeranz, K. Kinoshita, S.M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," *Proc. of the 30th Design Automation Conf.*, pp. 102-106, 1993.

[Koenemann 91] Koenemann, B., "LFSR-Coded Test Patterns for Scan Designs," *Proc. of European Test Conference*, pp. 237-242, 1991.

[Pomeranz 93] Pomeranz, I., L.N. Reddy, and S.M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits," *IEEE Trans. Computer-Aided Design*, Vol. 12, No. 7, pp. 1040-1049, Jul. 1993.

[Reeb 96] Reeb, B., H.-J. Wunderlich, "Deterministic Pattern Generation for Weighted Random Pattern Testing," *Proc. of European Design & Test Conference*, pp. 30-36, 1996.

[Tromp 91] Tromp, G., "Minimal Test Sets for Combinational Circuits," *Proc. of Int. Test Conference*, pp. 204-209, 1991.

[Yamaguchi 97] Yamaguchi, T., M. Tilgner, M. Ishida, and D. S. Ha, "An Efficient Method for Compressing Test Data," *Proc. of International Test Conference*, pp. 191-199, 1997.

[Zacharia 96] Zacharia, N., J. Rajski, J. Tyszer, and J.A. Waicukauski, "Two-Dimensional Test Data Decompressor for Multiple Scan Designs," *Proc. of International Test Conference*, pp. 186-194, 1996.

[Zorian 97] Zorian, Y., "Test Requirements for Embedded Core-based Systems and IEEE P1500," *Proc. of International Test Conference*, pp. 191-199, 1996.