Using Statistical Transformations to Improve Compression for Linear Decompressors

Samuel I. Ward

IBM Systems &Technology Group
11400 Burnet RD
Austin TX 78758
E-mail: siward@us.ibm.com

Chris Schattauer, Nur A.Touba

Computer Engineering Research Center
Dept. of Electrical & Computer Eng.
University of Texas, Austin, TX 78712
E-mail: touba@ece.utexas.edu

Abstract

Linear decompressors are the dominant methodology used in commercial test data compression tools. However, they are generally not able to exploit correlations in the test data, and thus the amount of compression that can be achieved with a linear decompressor is directly limited by the number of specified bits in the test data. The paper describes a scheme in which a non-linear decoder is placed between the linear decompressor and the scan chains. The non-linear decoder uses statistical transformations that exploit correlations in the test data to reduce the number of specified bits that need to be produced by the linear decompressor. Given a test set, a procedure is presented for selecting a statistical code that effectively "compresses" the number of specified bits (note that this is a novel and different application of statistical codes from what has been studied before and requires new algorithms). Results indicate that the overall compression can be increased significantly using a small non-linear decoder produced with the procedure described in this paper.

1. Introduction

Test data compression provides a means to reduce test costs by reducing tester storage, test time, and test data bandwidth requirements. Compressing the output response is relatively easy because lossy compression techniques can be employed, e.g., using a multiple input signature register (MISR). However, compressing test vectors is much more difficult because lossless compression techniques must be used.

A number of coding techniques for *test cubes* (i.e., deterministic test vectors where the unassigned bit postions are left as don't cares) have been investigated. These include run-length codes [Jas 98], selective Huffman codes [Jas 99, 03], Golomb codes [Chandra 00], frequency directed codes [Chandra 01], VIHC codes [Gonciari 02], LZ77 [Wolff 02], Mutation codes [Reda 02], packet-based codes [Khoche 02], [Volkerink 02], and non-linear combinational codes [Reddy 02], [Li 03], [Würtenberger 04]. A special class of test vector compression schemes involves using a linear decompressor which uses only linear operations to decompress the test vectors. This includes techniques based on linear feedback shift register (LFSR) reseeding and combinational linear expansion circuits consisting of XOR gates. Linear compression schemes are very efficient at exploiting don't care values in the test cubes to achieve large amounts of compression. All the commercial tools for compressing test vectors that have been developed so far are linear compression schemes including TestKompress from Mentor Graphics [Rajski 02], SmartBIST from Cadence [Könemann 01], DBIST from

Proceedings of the 2005 20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'05)
0-7695-2464-8/05 $20.00 © 2005 IEEE

Synopsys [Chandramouli 03], and VirtualScan from SynTest [Wang 04]. This paper describes a new technique that can be used in conjunction with linear compression schemes to significantly improve the amount of compression. It is applicable to any linear decompressor including both combinational and sequential.

The amount of compression that can be achieved with linear compression schemes depends directly on the number of specified bits in the test cubes. While linear decompressors are very efficient at exploiting don't cares in the test set, they cannot exploit correlations in the test data, and hence they cannot compress the test data to less than the total number of specified bits in the test data. The idea in the proposed scheme is to perform statistical transformations on the test cubes using non-linear hardware to reduce the number of specified bits that need to be encoded by the linear decompressor by exploiting correlations in the test data. A diagram of the proposed scheme is shown in Fig. 1. Some transformation hardware is added at the output of the linear decompressor. Given the set of test cubes that needs to be applied in the scan chains, this paper describes a systematic procedure for designing the transformation hardware in such a way that the input stream to the transformation hardware has fewer specified bits than the output stream (i.e., the test cubes) of the transformation hardware. Since the linear decompressor is now producing the input stream for the transformation hardware instead of the test cubes, the number of specified bits that need to be encoded by the linear decompressor has been reduced thereby allowing greater compression.

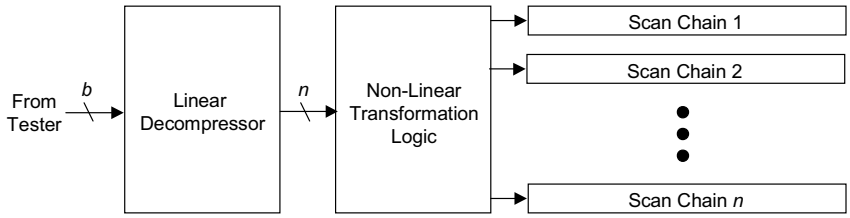

Figure 1. Diagram of Proposed Scheme

The proposed scheme is combining linear and non-linear coding together. There have been two earlier papers ([Krishna 02] and [Sun 04]) that did this as well, but in a fundamentally different way. In [Krishna 02], the inputs to the linear decompressor were encoded using a non-linear code. The objective in [Krishna 02] was to select the seeds for the LFSR in such a way that they could be effectively compressed by a non-linear code. In the proposed scheme, the inputs to the scan chains are encoded with a non-linear code. The objective here is to reduce the number of specified bits that need to be produced by the linear decompressor. Whereas the method in [Krishna 02] is only applicable for LFSR reseeding where the seed is periodically loaded, the proposed scheme is applicable for *any* linear decompressor including combinational and sequential continuous-flow decompressors (for which the method in [Krishna 02] cannot be used). In [Sun 04], dictionary coding and LFSR reseeding are combined such that either one or the other is used to load each scan bit-slice. In the proposed method, statistical coding is combined with a linear decompressor and both are used together for all scan bit-slices enabling a continuous-flow decompression with greater efficiency.

2. Proposed Scheme

In scan testing, the n scan chains are loaded with one n-bit "block" of data at a time each clock cycle (i.e., one bit-slice of the scan chain is loaded at a time). Given the test set, the set of n-bit blocks (i.e., bit-slices of the scan chain) can be obtained. The objective here then is to encode these blocks in a way that reduces the total number of specified bits. If the total number

of specified bits across all the blocks is reduced, then the linear decompressor requires less data from the tester thereby improving the compression (and hence reducing the number of bits that need to be stored on the tester).

The basic strategy for encoding the blocks in the proposed scheme is to use statistical coding. In conventional statistical coding, blocks that occur more frequently are encoded using codewords with fewer bits and blocks that occur less frequently are encoded using codewords with more bits in order to minimize the average length of a codeword. However, the problem here is not to minimize the average length of a codeword, but rather to minimize the number of specified bits. This is a novel problem that to the best of the authors' knowledge has never been studied before. It is more complicated than conventional statistical coding as will be seen.

In order to keep the decoder small, selective coding is employed similar to what was done in [Jas 99, 03]. An extra bit is added to indicate whether a block is coded or not. Only a subset of the blocks are coded while the rest are passed through unencoded. This allows the decoder to be designed only for the blocks where it can have the most significant impact. In conventional statistical coding the most significant impact would be encoding the most frequently occurring blocks because this would maximally reduce the total number of bits. However, this is not necessarily the case here where the goal is to minimize the number of specified bits and not the total number of bits. For example, if the most frequently occurring blocks have few specified bits, there may be a greater reduction in specified bits if another slightly less frequently occurring block is targeted which has more specified bits. This will be illustrated in the following example.

Consider the case where the set of blocks in the test set is shown in the first column of Table 1. A set of blocks that do not conflict in any bit position can be grouped together and represented by a group pattern that is compatible with every block in the group. Note that in Table 1 there are 6 groups and each group pattern is shown in the second column. The grouping is not unique (i.e., there are many different ways to group the blocks), however, it was shown in [Jas 03] that if the largest possible group is formed first, and then the next largest possible group, and so forth, that an optimal Huffman code can be constructed to obtain the minimum total number of bits after coding. This was done in Table 1 assuming that only 3 groups would be encoded with all other groups being sent unencoded. If the first bit of the codeword is 0, then the remaining bits in the codeword are the unencoded data itself. If the first bit of the codeword is 1, then the remaining bits are encoded and need to be decoded. So in Table 1, the three largest groups are encoded with the codewords **1**1, **1**01, and **1**00, all other groups are not encoded and have the first bit set to **0** to indicate that. As can be seen in Table 1, the total number of bits in all the blocks is equal to 108 before coding. The total number of bits after encoding is 63. Thus the data was compressed from 108 bits to 63 bits. However, since the proposed scheme involves using a linear decompressor to generate the codewords, the final compression will depend on how many specified bits the linear decompressor needs to generate and not the total number of bits. Thus, for the proposed scheme, the only thing that matters is the total number of specified bits. In Table 1, we see that the total number of specified bits in all the blocks is equal to 61 before coding. The total number of specified bits after encoding is 56. Thus the encoding used in Table 1 reduces the number of specified bits that need to be generated by the linear decompressor from 61 to 56.

While the encoding in Table 1 is optimal for minimizing the *total bits* after coding, it is not optimal for minimizing the *total specified bits* after coding. This can be seen by looking at Table 2 where the exact same set of blocks is encoded differently. In this case, the forth largest group from Table 1 (corresponding to group pattern 101110) is coded while the third largest group from Table 1 (corresponding to group pattern 100100) is left unencoded. The reason why this is better for minimizing specified bits is that the total number of specified bits is more in the fourth largest group (12) compared with the third largest group (9) even though it has

fewer blocks (2 compared to 3). Also note that the block xxxxx1 is not encoded with **1**01 even though it is compatible with group pattern 010001. The reason for this is that if it is encoded with **1**01 it requires 3 specified bits, however, if it is left unencoded (i.e., **0**xxxxx1), it requires only 2 specified bits. As can be seen, the code in Table 2 requires 71 total bits (compared with 63 in Table 1), but only 48 specified bits (compared with 56 in Table 1). For the proposed scheme, the code in Table 2 is better because the resulting codewords can be more efficiently compressed with the linear decompressor since they have fewer specified bits.

Table 1. Optimal Statistical Code with 3 Groups Encoded to Minimize *Total Bits*

Blocks	Group Pattern	Codeword	Total Bits Before Coding	Total Bits After Coding	Specified Bits before Coding	Specified Bits after Coding
11x000	110000	**1**1	6	2	5	2
11x000			6	2	5	2
11x0x0			6	2	4	2
11x00x			6	2	4	2
1x00xx			6	2	3	2
1xxxxx			6	2	1	2
x1xxxx			6	2	1	2
010x01	010001	**1**01	6	3	5	3
01x0x1			6	3	4	3
x1x001			6	3	4	3
xxxxx1			6	3	1	3
10xx0x	100100	**1**00	6	3	3	3
x00xxx			6	3	2	3
xxx1x0			6	3	2	3
101110	101110	**0**101101	6	7	6	7
101110			6	7	6	7
00xxx0	00xxx0	**0**00xxx0	6	7	3	4
0xxxx1	0xxxx1	**0**0xxxx1	6	7	2	3
			108	63	61	56

Table 2. Optimal Statistical Code with 3 Groups Encoded to Minimize *Total Specified Bits*

Blocks	Group Pattern	Codeword	Total Bits Before Coding	Total Bits after Coding	Specified Bits before Coding	Specified Bits after Coding
11x000	110000	**1**1	6	2	5	2
11x000			6	2	5	2
11x0x0			6	2	4	2
11x00x			6	2	4	2
1x00xx			6	2	3	2
1xxxxx			6	2	1	2
x1xxxx			6	2	1	2
010x01	010001	**1**01	6	3	5	3
01x0x1			6	3	4	3
x1x001			6	3	4	3
xxxxx1	xxxxx1	**0**xxxxx1	6	7	1	2
10xx0x	10xx0x	**0**10xx0x	6	7	3	4
x00xxx	x00xxx	**0**x00xxx	6	7	2	3
xxx1x0	Xxx1x0	**0**xxx1x0	6	7	2	3
101110	101110	**1**00	6	3	6	3
101110			6	3	6	3
00xxx0	00xxx0	**0**00xxx0	6	7	3	4
0xxxx1	0xxxx1	**0**0xxxx1	6	7	2	3
			108	71	61	48

A systematic procedure for selecting a statistical code to minimize the total number of specified bits after coding is described in Sec. 3. Once this code is obtained, the corresponding non-linear decoder can be synthesized.

3. Selecting Encoding

An iterative procedure is described here for finding a statistical code that minimizes the total number of specified bits. A flowchart for the procedure is shown in Fig. 2. There is a cyclical dependency in that grouping the blocks depends on the codewords while selecting the codewords depends on how the blocks are grouped. Thus, an iterative procedure is used where an initial set of codewords is first assumed and then the grouping is done. Based on the grouping, a new set of codewords is selected. Using the new set of codewords, the grouping is then redone. This process repeats as long as the compression continues improving. Fairly rapidly the procedure converges to a point where the compression no longer improves and the procedure terminates at that point. Details of each step of the procedure follow. Note that it does not guarantee optimality of the result because of the dependence on the initial code and the use of a greedy procedure with limited lookahead for grouping blocks.

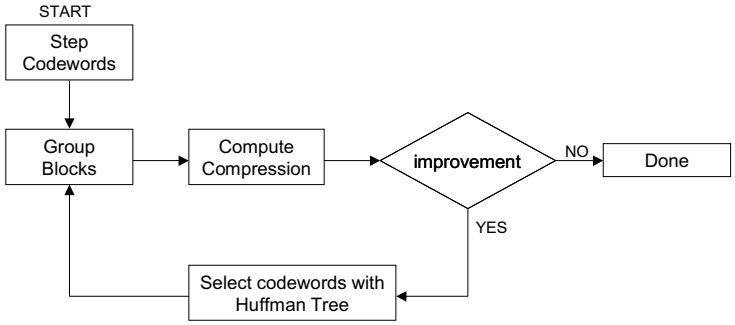

Figure 2. Flowchart for Procedure to Selecting Encoding

3.1 Step-Tree

The procedure begins with an initial set of codewords corresponding to a "step-tree". This is a coding scheme in which the codewords "step up" by one specified bit for each consecutive codeword. An example of a step-tree is shown in Fig. 3**.** Note that the codeword with only one specified bit is reserved for blocks that will not be encoded and thus the shortest codeword available for coded blocks has a length of two specified bits. Experiments indicated that the dependence on the initial code was not very significant and that good results were obtained using the step-code, hence this is used to initialize the procedure.

```
10XXXXX
110XXXX
1110XXX
11110XX
111110X
1111110
```

Figure 3. Example of Step-Tree for 6 bit blocks

3.2 Group Blocks

The next step is to group the blocks. The grouping depends on the codewords because if the number of specified bits in a block is less than the number of specified bits in the corresponding codeword, then the block should not be coded because it would be equally or more efficient to simply leave the block unencoded. Recall that this point was illustrated back in Table 2 where the block xxxxx1 was compatible with the group pattern 010001 and thus could have been coded with **1**01xxxx, however, it was more efficient to simply leave that block out of the group and let it be encoded with **0**xxxxx1 which requires fewer specified bits. So when forming the *i*-th group corresponding to the *i*-th most specified codeword, if the number of specified bits in the *i*-th most specified codeword is *s*, then no block with fewer than *s* specified bits should be added to the group.

The groups are formed one at a time. When forming a group, one block is added at a time to the group. All blocks that are compatible with the current group and have the same or more specified bits than the group are candidates to add to the group (note that initially the group is empty and thus all blocks are compatible with it). A lookahead procedure is used to decide which candidate block to add to the group by considering how many blocks would remain compatible with the group after each candidate block is added. The candidate block that would preserve the most compatibility is added to the group. Blocks are added one by one to a group until a point is reached where there are no more candidate blocks to add to the group. Then the next group is constructed in the same manner.

3.3 Compute Compression

After the groups have been formed, then the compression with respect to the number of specified bits is computed. The encoded groups are replaced by their corresponding codewords and the unencoded blocks have the extra bit added to them to indicate that there are unencoded. The total number of specified bits after coding is then computed. If the compression of specified bits is the same as the last iteration, then the procedure terminates. If not, then the procedure is repeated with a new set of codewords formed as described in the next subsection.

3.4 Select New Set of Codewords

For the next iteration of the procedure, a new set of codewords is selected. Using the groups that were formed in the last iteration, the "frequency" of each group is used to construct a Huffman tree [Huffman 52]. However, the "frequency" in this case is the frequency of specified bits, which is equal to the total number of specified bits across all the blocks contained in the group. In conventional statistical coding (e.g., what was used in [Jas 99, 03]), the goal is to minimize the average number of bits in each codeword and hence "frequency" is equal to the number of blocks in each group. However, here the goal is to minimize the average number of specified bits in each codeword and hence "frequency" is equal to the number of specified bits in all the blocks in each group. From the Huffman tree, the new set of codewords are obtained (see [Huffman 52] for more details). Since selective coding is used, only a certain number of groups are coded and not all. This is to keep the decoder size small as was described earlier.

After the new set of codewords are selected, the procedure repeats the step described in Sec. 3.2 where the blocks are grouped again to better optimize them for the new codewords (since the number of specified bits in each codeword may have changed from the last iteration).

4. Improving Compression

In this section, some ideas are described for further improving the compression. Note that it is possible for some individual test cubes in the test set to not achieve positive compression (i.e., to end up with more specified bits) with the proposed scheme even though significant compression is achieved on the overall test set. The reason for this is that for each unencoded block an extra specified bit is added to indicate that it is unencoded, and thus if the proportion of encoded blocks to unencoded blocks in a particular test cube is not sufficiently high, that test cube may not achieve positive compression. For those test cubes, it is better to simply bypass the non-linear decoder. This can be easily implemented by taking advantage of the fact that the order in which the test cubes are applied in the scan chains doesn't matter. Thus, the test cubes can be ordered so that all the test cubes that do not achieve positive compression come at the end. The non-linear decoder can then simply be bypassed for this last set of test cubes that do not achieve positive compression. The only hardware required for this is an AND gate to decode the pattern counter, a bypass activation flip-flop, and a bypass MUX for the non-linear decoder.

Another way to improve the compression would be to modify the decoder so that it has two (or more) modes in which it decodes the same codewords, but outputs different group patterns depending on which mode it is in. This would allow different groups of blocks to be coded for different test cubes. Essentially, the first mode would be used for the first so many test cubes, and then the mode would be changed for the next set of test cubes and so on. This would improve the compression by allowing more efficient encoding at the cost of the additional hardware required to implement the extra modes for the decoder.

5. Experimental Results

Experiments were performed on the four largest ISCAS 89 circuits using a scan architecture with 64 scan chains. A non-linear decoder for each of the circuits was designed using the procedure described in Sec. 3. The decoder was designed with 2 modes and a bypass MUX as described in Sec. 4. Table 3 shows the results. The third column shows the hardware overhead that was required in terms of gate equivalents per scan chain. The gate equivalents were computed as *0.5n* for an *n*-input NAND/NOR gate and *0.5* for an inverter. The fourth column shows the number of test vectors in the test set. The fifth column shows the original number of specified bits without the non-linear decoder. The sixth column shows the number of specified bits when the non-linear decoder is used. The last column shows the percent reduction in the number of specified bits that is achieved by using the propose scheme. As can be seen, significant reduction is achieved for all the circuits. This reduction in the specified bits is a very powerful result because it means that in most cases, an additional 20% greater compression can be achieved *on top of* the best possible compression that is currently available for any linear decompression scheme. If the test data bandwidth is held constant, this translates to a 20% reduction in test time.

Table 4 shows a comparison of different test data compression schemes in terms of their tester storage requirements. The last two schemes use the sequential decompressor shown in Fig. 4 where a 64-bit LFSR is used with a variable length shift (the shift length is loaded in the first two clock cycles of each scan vector). The sequential decompressor alone is very powerful, but when it is combined with the non-linear decompressor, the results are significantly improved. As can be seen in Table 4, the results are similar to the seed compression scheme in [Krishna 02] which is not unexpected since both schemes are essentially combining a linear decompressor with a non-linear decoder (though in very different ways). The advantage of the proposed scheme is that it can facilitate continuous-flow decompression where the tester transfers the test data as fast as it can with a constant bandwidth. The scheme in [Krishna 02] requires a more complicated non-linear decoder that

does not allow continuous-flow decompression. The scheme in [Sun 04] which is also combining linear and non-linear decompression reports similar results, but it uses a much larger decoder that is effectively storing a lot of the test set on-chip (note that the compression with the proposed scheme could also be improved by increasing the size of the decoder if that was a desired result). Note that the scheme described in [Balakrishnan 04] that uses scan inversion to improve compression could be used in conjunction with the proposed scheme to achieve even better results.

Table 3. Results for Proposed Scheme

Circuit	Scan Chains	Overhead GE/chain	Test Vectors	Original Specified Bits	Compressed Specified Bits	Percent Reduction
s13207	64	5.81	266	9392	7499	20.2%
s15850	64	5.70	226	10869	8333	23.3%
s38417	64	6.64	105	30671	22277	27.4%
s38584	64	6.69	192	26187	23293	11.1%

Table 4. Comparison of Test Data for Different Encoding Schemes

Circuit Name	Illinois Scan Architecture [Hamzaoglu 99]		FDR Codes [Chandra 01]		Seed Overlapping [Rao 03]		Seed Compression [Krishna 02]		Sequential Decompressor Alone		Sequential Decompressor with Proposed Decoder	
	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Bits	Num. Vect.	Total Bits
s13207	273	109,772	236	30,880	272	17,970	266	11,285	266	14,301	266	10,810
s15850	178	32,758	126	26,000	174	15,774	269	12,438	269	14,391	269	12,405
s38417	337	96,269	99	93,466	288	60,684	376	34,767	376	48,612	376	32,154
s38584	239	96,056	136	77,812	215	31,061	296	29,397	296	34,012	296	31,000

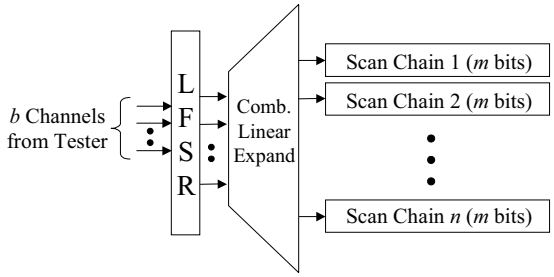

Figure 4. Sequential Linear Decompressor

6. Conclusions

The proposed scheme combines linear decompressors with a non-linear decoder to provide very high levels of compression for test data. Designing a non-linear decoder to reduce the number of specified bits is a new problem quite different from conventional encoding problems. This paper described a procedure for designing such a decompressor and showed that very good results can be obtained. The proposed scheme provides a promising way to achieve greater levels of test data compression than what conventional linear decompressors alone can achieve. Linear decompressors alone cannot exploit correlations in the test set, and thus are limited by the number of specified bits in the test set. The proposed scheme provides a way to get beyond that limitation by exploiting correlations in the test set as well.

Acknowledgements

This material is based on work supported in part by the Intel Corporation and in part by the National Science Foundation under Grant No. CCR-0306238.

References

[Balakrishnan 04] Balakrishnan, K., and N.A. Touba, "Improving Encoding Efficiency for Linear Decompressors Using Scan Inversion," Proc. of International Test Conference, pp. 936-943, 2004.

[Chandra 00] Chandra, A., and K. Chakrabarty, "Test Data Compression for System-on-a-Chip Using Golomb Codes," *Proc. of VLSI Test Symposium*, pp. 113-120, 2000.

[Chandra 01] Chandra, A., and K. Chakrabarty, "Frequency-Directed Run Length (FDR) Codes with Application to System-on-a-Chip Test Data Compression," *Proc. of VLSI Test Symposium*, pp. 42-47, 2001.

[Chandramouli 03] Chandramouli, M., "How to Implement Deterministic Logic Built-In Self-Test (BIST)," *Complier: A Monthly Magazine for Technologists Worldwide*, Synopsys, Jan. 2003.

[Gonciari 02] Gonciari, P.T., B. Al-Hashimi, and N. Nicolici, "Improving Compression Ratio, Area Overhead, and Test Application Time for System-on-a-Chip Test Data Compression/Decompression," *Proc. of Design Automation and Test in Europe (DATE)*, pp. 604-611, 2002.

[Hamzaoglu 99] Hamzaoglu, I., and J.H. Patel, "Reducing Test Application Time for Full Scan Embedded Cores," *Proc. of Int. Symposium on Fault Tolerant Computing*, pp. 260-267, 1999.

[Huffman 52] Huffman, D.A., "A Method for the Construction of Minimum Redundancy Codes," *Proc. of IRE*, Vol. 40, No. 9, pp. 1098-1101, Sep. 1952.

[Jas 98] Jas, A., and N.A. Touba, "Test Vector Decompression Via Cyclical Scan Chains and Its Application to Testing Core-Based Designs", *Proc. of Int. Test Conference*, pp. 458-464, 1998.

[Jas 99] Jas, A., J. Ghosh-Dastidar, and N.A. Touba, "Scan Vector Compression/Decompression Using Statistical Coding", *Proc. of IEEE VLSI Test Symposium*, pp. 114-120, 1999.

[Jas 03] Jas, A., J. Ghosh-Dastidar, M.-E. Eng, and N.A. Touba, "An Efficient Test Vector Compression Scheme Using Selective Huffman Coding," *IEEE Trans. on CAD*, Vol. 22, No. 6, pp. 797-806, Jun. 2003.

[Khoche 02] Khoche, A., E.H. Volkerink, J. Rivoir, and S. Mitra, "Test Vector Compression Using EDA-ATE Synergies," *Proc. of VLSI Test Symposium*, pp. 97-102, 2002.

[Krishna 02] Krishna, C.V., and N.A. Touba, "Reducing Test Data Volume Using LFSR Reseeding with Seed Compression ", *Proc. of IEEE International Test Conference*, pp. 321-330, 2001.

[Könemann 01] Könemann, B., "A SmartBIST Variant with Guaranteed Encoding" *Proc. of Asian Test Symposium*, pp. 325-330, 2001.

[Li 03] Li, L., and K. Chakrabarty, "Test Data Compression Using Dictionaries with Fixed-Length Indices," *Proc. of VLSI Test Symposium*, pp. 219-224, 2003.

[Rajski 02] Rajski, J., *et al.,* "Embedded Deterministic Test for Low Cost Manufacturing Test," *Proc. of Int. Test Conf.*, pp. 301-310, 2002.

[Rao 03] Rao, W., I. Bayraktaroglu, and A. Orailoglu, "Test Application Time and Volume Compression through Seed Overlapping," *Proc. of Design Automation Conference*, pp. 732-737, 2003.

[Reda 02] Reda, S., and A. Orailoglu, "Reducing Test Application Time Through Test Data Mutation Encoding", *Proc. of Design, Automation, and Test in Europe*, pp. 387-393, 2002.

[Reddy 02] Reddy, S., K. Miyase, S. Kajihara, and I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs", *Proc. of VLSI Test Symposium*, pp. 103-108, 2002.

[Sun 04] Sun, X., L. Kinney, and B. Vinnakota, "Combining Dictionary Coding and LFSR Reseding for Test Data Compression," *Proc. of Design Automation Conference*, pp. 944-947, 2004.

[Volkerink 02] Volkerink, E.H., A. Khoche, and S. Mitra, "Packet-based Input Test Data Compression Techniques," *Proc. of International Test Conference*, pp. 154-163, 2002.

[Wang 04] Wang, L.-T., X. Wen, H. Furukawa, F.-S. Hsu, S.-H. Lin, S.-W. Tsai, K.S. Abdel-Hafez, and S. Wu, "VirtualScan: A New Compression Scan Technology for Test Cost Reduction," Proc. of International Test Conference, pp. 916-925, 2004.

[Wolff 02] Wolff, F.G., and C. Papachristou, "Multiscan-based Test Compression and Hardware Decompression Using LZ77," *Proc. of International Test Conference*, pp. 331-339, 2002.

[Würtenberger 04] Würtenberger, A., C.S. Tautermann, and S. Hellebrand, "Data Compression for Multiple Scan Chains Using Dictionaries with Corrections", *Proc. of International Test Conference*, pp. 926-935, 2004.