Test Transformation to Improve Compaction by Statistical Encoding

Hideyuki Ichihara* and Kozo Kinoshita
Dept. of Applied Physics
Graduate School of Engineering
Osaka University
Suita-shi, Osaka 5650871, Japan

Irith Pomeranz* and Sudhakar M. Reddy*
Electrical and Computer Engineering Department
University of Iowa
Iowa City, IA 52242, U.S.A.

Abstract

In test compression / decompression schemes the objective is to achieve the highest compression without sacrificing fault coverage. In this paper, we propose a method to transform a test set in order to achieve higher compression without sacrificing fault coverage, while using statistical encoding techniques. The compression ratio of a test set depends on the entropy of the test set, and hence our test transformation method decreases the entropy of the test set without losing the fault coverage. Experimental results show that the proposed method transforms given test sets into highly compressible ones.

1. Introduction

As the size and complexity of VLSI circuits increase, the size of test sets for such circuits also increases. The increase in test set size requires larger storage and longer time to transport the tests from the storage device to the circuit-under-test (CUT). The longer test transportation time severely impacts test application time, especially in core based designs, since each individual core requires a separate test set. To alleviate this problem, a method using compression/decompression of test sets has been proposed [1]-[3]. Fig. 1 shows a block diagram for such a scheme. In this scheme, a given test set is compressed by a data compression technique and stored in an on-chip or off-chip memory. While testing a CUT, the compressed test set is transported to a decompressor on the chip or on the tester, and then decompressed and fed to the CUT. Due to the compressed test set, both the size of the test storage device and the time for test transportation are reduced.

* Work done was supported in part by NSF Grant No. MIP-935758 and in part by SRC Grant No. 98-TJ-645.

The method of [1] used the Burrows-Wheeler transformation and run-length code for test compression. This method can achieve high compression of test sets, so that the transportation time from the test storage device to a CUT decreases greatly. However, the decompression is complex and hence is implemented on a tester in software. This scheme may not be applicable to manufacturing testing of core based designs.

The methods described in [2] and [3] employ *statistical coding* as a compression method for a test set. Since a statistical code decides the lengths of codewords according to the probability of occurrences of each unique pattern, the compression method using statistical encoding is efficient. In addition, the decoder (decompressor) is also relatively simple. In [2], a BIST architecture for non-scan sequential circuits based on a statistical code has been presented. This method is useful for circuits in which the number of primary inputs is relatively small. In [3], a method to compress scan vectors by a statistical code has been proposed. The paper presents a new statistical code called a *selective code* based on the Huffman code. Although the average length of a codeword of a selective code is a little longer than that of the Huffman code, the decoder for the selective code needs fewer states than that for the Huffman code.

The method of [1] treats the test set as one piece of data. Therefore, the decompression cannot be performed before the complete compressed data is sent to

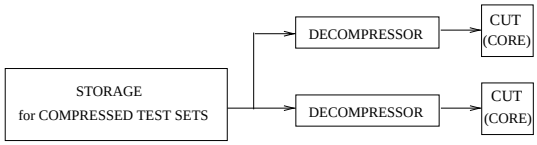

Figure 1. Scheme of compression / decompression for a test set

the CUT. In contrast, the method of [2] encodes each test vector in the test set, and the method of [3] partitions each vector into blocks and encodes each block separately. Hence, the encoded test vectors/blocks are decoded as soon as they reach the decoder.

The method we propose in this paper is useful for enhancing the compression achieved by statistical encoding. Our method can augment the methods of [2] and [3] without losing the advantage of these methods. Our method is based on the observation that the compression ratio of an encoding scheme depends on the *entropy* of the test set, which corresponds to the average number of bits needed to represent a codeword. Our idea is to decrease the entropy by transforming the given test set without losing fault coverage. The decrease in entropy causes the average length of a codeword to decrease and, in many cases, also decreases the size of the decoder as a by-product.

In this paper, we first consider an appropriate transformation of a test set to reduce the entropy. Next, we propose a procedure to transform the test set using fault simulation. Our transformation procedure greedily attempts every transformation that may decrease the entropy.

The remainder of this paper is organized as follows. Section 2 reviews the statistical encoding of a test set. Section 3 describes a transformation of a test set in order to reduce the entropy. Section 4 proposes a procedure for test transformation. Section 5 gives experimental results, and Section 6 concludes the paper.

2. A statistical code for a test set

In this section, we illustrate statistical encoding for a given test set, which was described in [3]. To encode a given test set, the test set is divided into n-bit blocks, i.e., each block is an n-bit pattern. Table 1 shows a test set which consists of twelve test vectors divided into 4-bit blocks. The reason for partitioning a test vector into blocks is to keep the complexity of the decompression circuit and decompression delays low. As noted in [3], the block length must be higher than a minimum value. Each block pattern is mapped into a variable-length codeword. The length of a codeword depends on the probability with which each pattern occurs in the test set. The more frequently a pattern occurs, the shorter the length of a codeword for it. Table 2 shows the probabilities p_i of occurrence of each unique pattern x_i in the test set of Table 1. In addition, the code words in two statistical codes, a Huffman code and a selective code [3], are shown based on the probability of occurrence of unique patterns.

¿From the probabilities of occurrence of unique patterns, we can calculate the entropy of the test set. The entropy of a test set is given by $H = -\sum_{i=1}^{m} p_i \cdot \log_2 p_i$, where the test set includes m unique patterns $x_1, x_2, \ldots, x_m$ with probabilities of occurrence $p_1, p_2, \ldots p_m$, respectively. The entropy of the test set of Table 1 is 3.2740.

A Huffman code is an optimal statistical code in which the average length of a codeword is the closest to the entropy. The average length of a statistical code is equal to $\sum_{i=1}^{m} p_i \cdot w_i$, where w_i is the length of the codeword of x_i. As shown in Table 2, in this example, the average length of a Huffman codeword is 3.3125, which is close to the entropy 3.2740. In addition, a Huffman code has the important property that it is prefix-free, i.e., a codeword is never a prefix of another codeword. This property makes the decoder simple. If the decoder for a Huffman code is constructed as a finite state machine (FSM), such a decoder requires m states, where m is the number of unique patterns. In Table 2, the decoder for the Huffman code requires

Table 1. A test set divided into 4-bit blocks

test 1:	1111 0101 **0011** 1111 1011 1110 1101 1011
test 2:	1111 1111 1111 1111 0000 0000 0000 0000
test 3:	1001 0010 0110 0111 1110 1101 0110 1110
test 4:	1111 1111 1110 0110 1111 1001 1111 1011
test 5:	1111 1111 0111 1010 1111 1111 0111 1111
test 6:	0010 1110 1000 0100 1111 1111 1111 1111
test 7:	0110 1011 1011 1011 1101 0111 1011 0111
test 8:	1100 1000 1010 0111 0101 1011 1111 1101
test 9:	1011 0100 1101 1101 1110 1111 1111 1111
test10:	1111 1011 0111 0101 1111 0111 1111 1101
test11:	1100 1101 1001 1110 1110 1101 1011 0110
test12:	0111 0010 1111 1111 0111 1111 1011 1111

Table 2. Probability table for a test set

i	x_i	p_i	Huffman	Selective
1	1111	0.3125	11	11
2	1011	0.1250	100	101
3	0111	0.1042	010	100
4	1101	0.0938	000	01011
5	1110	0.0833	1011	00111
6	0110	0.0521	0011	00110
7	0000	0.0417	10101	00000
8	0010	0.0313	01111	00100
9	1001	0.0313	01110	01001
10	0101	0.0313	01101	01010
11	1100	0.0208	01100	00011
12	0100	0.0208	00101	00010
13	1000	0.0208	00100	00001
14	1010	0.0208	101001	00101
15	0011	0.0104	101000	01100
Entropy: 3.2740			ave. 3.3125	ave. 3.6041
#states of FSM			15	7

fifteen states.

The selective code was designed based on the Huffman code [3]. In a selective code, among m unique patterns, $k (k < m)$ patterns with the highest probability of occurrence are selected and encoded by a Huffman code. The remaining $m - k$ patterns are not encoded. To distinguish the two kinds of patterns, the codeword of the k selected patterns is a '1' followed by the Huffman code and, on the other hand, the codeword of the $m - k$ remaining patterns is a '0' followed by the original fixed-length pattern. In the example of Table 2, the number of selected patterns is three. The average length of a selective codeword in this example is 3.6041, which is a little greater than that of the Huffman code. However, the selective code has the advantage that its decoder requires smaller number of states than the decoder of the Huffman code. The number of states of a FSM decoder for a selective code is $n+k$, where n is the number of bits of a block (4 in the example here) and k is the number of patterns selected. In the example of Table 2, the decoder for the selective code requires seven states.

3. Decreasing the entropy

In order to reduce the total length of an encoded test set, two approaches can be taken. One approach is to incorporate steps into the test generation procedure aimed at producing tests that yield short encoded tests. The other approach is to modify a given test set/test sequence such that the modified tests yield shorter encoded tests. In an earlier work both these approaches were investigated [4]. In [4] the objective used was to reduce the number of unique test vectors in a test sequence for a sequential circuit. Another objective applicable to both combinational and sequential circuits was to reduce the number of unique fault-free circuit output vectors obtained for test vectors [4]. In this work, our objective is to reduce the length of the codewords used to encode the n-bit blocks of a test vector. We achieve this objective by modifying a given set of test vectors without losing fault coverage.

In the previous section we demonstrated that the entropy of a test set gives the lower bound on the average length of the encoded block, and hence the total length of the encoded test set. Thus, modification to the test set that will reduce the entropy will lead to a shorter encoded test set. We use this observation in the procedure described next.

Consider the example of Table 1 again. Assume that the pattern of the third block "0011" (which is boldfaced) can be replaced with the pattern "1111". Patterns "0011" and "1111" correspond to patterns x_{15} and x_1 in Table 2, respectively. Table 3 shows the probabilities of occurrence of each pattern and the Huffman code constructed after this replacement. Note that the difference between Tables 2 and 3 is not only the probabilities of x_1 and x_{15} but also the length of the codeword of x_{14}. As a result of these differences, the entropy decreases from 3.2740 to 3.2076 and the average length of a Huffman codeword decreases from 3.3125 to 3.2500. The total size of the encoded test set reduces from 318 bits to 312 bits.

Table 3. Probability table for a test set (after a pattern replacement)

i	x_i	f_i	Huffman code
1	1111	0.3229	11
2	1011	0.1250	100
3	0111	0.1042	010
4	1101	0.0938	001
5	1110	0.0833	1011
6	0110	0.0521	0110
7	0000	0.0417	10101
8	0010	0.0313	01110
9	1001	0.0313	01111
10	0101	0.0313	10100
11	1100	0.0208	00000
12	0100	0.0208	00001
13	1000	0.0208	00010
14	1010	0.0208	00011
15	0011	0.0000	—
Entropy: 3.2076			ave. 3.2500

In general, a test includes some don't-care bits whose values do not affect the number of faults detected by the test. Furthermore, many faults are detected by several tests in the test set. Therefore, many block patterns can be replaced with other block patterns without losing the fault coverage, and hence the replacement described above may be applicable to many block patterns.

The pattern replacements that will help reduce the entropy are characterized by Lemma 1 given next.

Lemma 1: If pattern x_a in a test set can be replaced with pattern x_b that occurs more frequently than x_a,i.e. $p_a < p_b$, then the entropy of the encoded test set decreases.

Proof: Let the entropy before a replacement be given by equation (1).

$$H = -p_1 \cdot \log_2 p_1 - \ldots - p_a \cdot \log_2 p_a - \ldots - p_b \cdot \log_2 p_b - \ldots - p_m \cdot \log_2 p_m, \quad (1)$$

where m is the number of unique patterns in the test set. After replacing x_a with x_b, the entropy changes to

the one given by equation (2).

$$H' = -p_1 \cdot \log_2 p_1 - \ldots - (p_a - p_c) \cdot \log_2(p_a - p_c) - \ldots - (p_b + p_c) \cdot \log_2(p_b + p_c) - \ldots - p_m \cdot \log_2 p_m, \quad (2)$$

where $p_c > 0$ is a decrement of p_a and an increment of p_b due to the replacement. Therefore, the difference of the entropy before and after the replacement is

$$\begin{aligned} H' - H &= -(p_a - p_c) \cdot \log_2(p_a - p_c) \\ &\quad -(p_b + p_c) \cdot \log_2(p_b + p_c) \\ &\quad +p_a \cdot \log_2 p_a + p_b \cdot \log_2 p_b \\ &= \{p_a \cdot \log_2 p_a - (p_a - p_c) \cdot \log_2(p_a - p_c)\} \\ &\quad -\{(p_b + p_c) \cdot \log_2(p_b + p_c) - p_b \cdot \log_2 p_b\}. \end{aligned} \quad (3)$$

Let $g(x) = x \cdot \log_2 x$, then

$$H' - H = \{g(p_a) - g(p_a - p_c)\} - \{g(p_b + p_c) - g(p_b)\}. \quad (4)$$

Dividing equation (4) by $p_c > 0$,

$$\frac{H' - H}{p_c} = \frac{g(p_a) - g(p_a - p_c)}{p_c} - \frac{g(p_b + p_c) - g(p_b)}{p_c} \quad (5)$$

To prove Lemma 1, we need to show that $H' - H < 0$. If we show that $\frac{H'-H}{p_c} < 0$, then we would have shown that $H' - H < 0$ because $p_c > 0$.

Consider the second derivative of $g(x)$,

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2} g(x) = \frac{\mathrm{d}}{\mathrm{d}x}(\log_2 x + \frac{1}{ln2}) = \frac{1}{ln2} \cdot \frac{1}{x}. \quad (6)$$

¿From equation (6) it can be seen that the second derivative is strictly positive if $x > 0$, and hence the first derivative of $g(x)$, $\frac{\mathrm{d}}{\mathrm{d}x} g(x)$, is a monotonically increasing function while $x > 0$. This means that

$$\frac{g(x_1 + \Delta x) - g(x_1)}{\Delta x} < \frac{g(x_2 + \Delta x) - g(x_2)}{\Delta x}, \quad (7)$$

if $x_1 < x_2$ and $\Delta x > 0$. Let $x_1 = p_a - p_c, x_2 = p_b$ and $\Delta x = p_c$, then equation (7) is transformed into

$$\frac{g(p_a) - g(p_a - p_c)}{p_c} < \frac{g(p_b + p_c) - g(p_b)}{p_c}. \quad (8)$$

Note that the conditions $x_1 < x_2$ and $\Delta x > 0$ are maintained since $0 < p_a - p_c < p_a < p_b < p_b + p_c$ and $p_c > 0$. From equation (5) and inequality (8), we conclude that $H' - H < 0$ and hence $H' < H$. Therefore, if a pattern is replaced with another pattern which occurs more frequently, the entropy of the encoded test set becomes smaller.

4. Procedure for the transformation of a test set

When a test set with its vectors divided into fixed-length blocks is given, an outline of our procedure to modify the test set is as follows. The pattern replacements attempted are characterized by Lemma 1. We ensure that the replacements do not decrease the fault coverage by simulating the modified test sets before a replacement is accepted.

To explain our procedure, we use the following notation. N is the number of block patterns. B_i ($1 \leq i \leq N$) are the set of blocks with pattern x_i, and blocks included in B_i are numbered b_{ij}. Flag e_i shows whether every candidate replacement for blocks in B_i has been attempted.

Procedure_Replace
(1) Set every $e_i = 0$. Set $X = \{x_1, x_2 \ldots x_N\}$.
(2) Arrange X in decreasing order of the frequency of occurrence of $x_i \in X$.
(3) Select the block pattern x_p that occurs least frequently and whose flag $e_p = 0$.
(4) Repeat (4-1) for each $b_{pq} \in B_p$.
(4-1) Repeat (4-1-1) to (4-1-3) for each $k = N, N - 1, \ldots p + 1$.
(4-1-1) Replace block pattern of b_{pq} with x_k.
(4-1-2) Fault simulate the modified test set.
(4-1-3) If the fault coverage decreases, then retrieve the original test set. Otherwise go to (7)
(5) Set $e_p = 1$.
(6) If every $e_i = 1$ for $i = 2, \ldots N$, halt. Otherwise go to (3).
(7) If $B_i = \phi$, remove x_i from X and set $N = N - 1$.
(8) go to (2).

We demonstrate our procedure using Table 4. Suppose that a given test set is divided into 4-bit blocks and four unique block patterns occur. Table 4(a) shows the blocks of the initial test set. Column f_i denotes the number of occurrence of pattern x_i (this replaces the probability p_i). The patterns in the table are arranged in decreasing order of the number of their occurrences.

Our procedure selects a block to be replaced from the last row to the second row. Table 5 illustrates each step of our procedure. In the first step, block b_{41} is selected. Pattern x_4 of block b_{41} is replaced with pattern x_1 and fault simulation is performed. In this example, this replacement fails, i.e., the fault coverage decreases. This is indicated by F in the last column of Table 5. In the second and third steps of Table 5, pattern x_4 of block b_{41} is replaced with patterns x_2 and x_3 in this order and the fault coverage is checked. From the

Table 4. Example of the replacement procedure

(a) Initial table

i	x_i	f_i	e_i	B_i
1	1111	6	0	$b_{11}, b_{12}, b_{13}, b_{14}, b_{15}, b_{16}$
2	1011	4	0	$b_{21}, b_{22}, b_{23}, b_{24}$
3	0111	2	0	b_{31}, b_{32}
4	1101	2	0	b_{41}, b_{42}

(b) After the sixth step of table 5

i	x_i	f_i	e_i	B_i
1	1111	6	0	$b_{11}, b_{12}, b_{13}, b_{14}, b_{15}, b_{16}$
2	1011	4	0	$b_{21}, b_{22}, b_{23}, b_{24}$
3	0111	2	0	b_{31}, b_{32}
4	1101	2	1	b_{41}, b_{42}

(c) After the eighth step of table 5

i	x_i	f_i	e_i	B_i
1	1111	6	0	$b_{11}, b_{12}, b_{13}, b_{14}, b_{15}, b_{16}$
2	1011	5	0	$b_{21}, b_{22}, b_{23}, b_{24}, b_{31}$
3	0111	1	0	b_{32}
4	1101	2	1	b_{41}, b_{42}

(d) After the ninth step of table 5

i	x_i	f_i	e_i	B_i
1	1111	6	0	$b_{11}, b_{12}, b_{13}, b_{14}, b_{15}, b_{16}$
2	1011	5	0	$b_{21}, b_{22}, b_{23}, b_{24}, b_{25}$
3	1101	2	1	b_{31}, b_{32}
4	0111	1	0	b_{41}

fourth step to the sixth step, block b_{42} is selected and its pattern is replaced with patterns x_1, x_2 and x_3 in this order. Every one of these replacements fails. In the sixth step, to denote that every replacement for blocks in B_4 was attempted, flag e_4 is set to 1. Table 4(b) shows the change made to Table 4(a) after the sixth step of Table 5. In the seventh and the eighth steps of Table 5, the attempts for replacement of blocks in B_3 are demonstrated. In this example, the replacement of pattern x_3 of block b_{31} with pattern x_2 does not decrease the fault coverage. Table 4(c) shows the status of the number of occurances of blocks after step 8 of Table 5. In the ninth step of Table 5, the third row of Table 4(c) is exchanged with the fourth row and Table 4(d) is obtained. Now the subscripts corresponding to i in b_{ij} are re-numbered. Once a re-arrangement of the table occurs, the process continues from the last row of the table again. (Thus, it begins with $i = 4$.) After the tenth step, the replacement attempts for $i = 3$ are skipped because $e_3 = 1$ and the process continues until all replacements for blocks in B_2 are attempted.

5. Experimental results

We implemented our procedure in C++ and ran it on a PC (CPU: Pentium II, OS:Linux, MEMORY: 64M).

The procedure was applied to the test sets derived by MINTEST [5] for ISCAS'85 benchmark circuits. These test sets are the smallest known test sets for the ISCAS'85 benchmarks. It should also be noted that these tests did not have any unspecified or don't care values.

Table 5. Steps of the execution of the replacement procedure

step	i	b_{ij}	x_i	e_i	Fail/Success
1	4	b_{41}	x_1		F
2			x_2		F
3			x_3		F
4		b_{42}	x_1		F
5			x_2		F
6			x_3	$e_4 = 1$	F
7	3	b_{31}	x_1		F
8			x_2		S
9	Re-order the frequency table				
10	4	b_{41}	x_1	$e_4 = 1$	F
11	2	b_{21}	x_1		F

In Table 6 we report the results when Huffman encoding is used. After the circuit name, the block size used to partition the test vectors is shown. We used two different block sizes as shown. In the next column we show the total number of bits in the original test vectors, i.e., the product of the number of test vectors and their length. In the next three columns we show the number of bits in the tests encoded using Huffman encoding, followed by the ratio of the number of bits in the encoded tests and the original tests, and then the number of states in the FSM needed to decode the encoded tests. The same information for the case where Huffman encoding is employed after the transformation of the test sets as proposed here is given in the next three columns. In the last column we give the time to derive the transformed test sets.

For c1908 and c5315, the test sets encoded before the transformation need more bits than the original test sets. This is due to the fact that the block size does not evenly divide the test vector length and the last few bits are randomly filled for the last block.

¿From Table 6 it can be seen that the proposed transformation reduces the size of the encoded test sets significantly. Additionally, the number of states of the FSM needed to decode the encoded tests is also reduced.

In Table 7 we report the results when a selective code [3] is used to encode the tests. The number of the selected block patterns is decided to minimize the total number of bits of the encoded test set. In several cases, the test sets encoded before the transformation need more bits than the original test sets. This is because the entropy of the original test set is too large to disregard the influence of an additional distinguishing

Table 6. Results for Huffman code

circuit	block size	total #bits	before transformation			after transformation			time(sec.)
			bits	ratio	#states	bits	ratio	#states	
c432	4	972	950	0.977	15	480	0.493	9	0.72
	10		633	0.651	87	317	0.326	23	1.59
c499	4	2132	1843	0.864	15	1825	0.856	15	2.88
	10		1425	0.668	126	1188	0.557	74	16.66
c880	4	960	930	0.968	15	701	0.730	12	1.99
	10		614	0.639	83	473	0.492	42	8.99
c1355	4	3444	2524	0.732	15	2413	0.700	15	9.40
	10		2106	0.611	152	1904	0.552	108	121.74
c1908	4	3498	3500	1.000	15	3036	0.867	15	38.50
	10		2832	0.809	277	2036	0.582	99	251.08
c2670	4	10252	10252	1.000	15	4874	0.475	15	52.38
	10		9379	0.914	631	3238	0.315	104	484.27
c3540	4	4200	4200	1.000	15	2589	0.616	15	54.69
	10		3524	0.839	343	1907	0.454	77	432.86
c5315	4	6586	6588	1.000	15	4201	0.637	15	131.52
	10		5837	0.886	480	2830	0.429	97	1150.35
c6288	4	384	318	0.828	14	293	0.763	13	20.32
	10		198	0.515	33	179	0.466	27	68.56
c7552	4	15111	14997	0.992	15	7790	0.515	15	353.72
	10		14103	0.933	743	6149	0.406	148	7207.38

Table 7. Results for selective code

circuit	block size	before		after	
		ratio	#states	ratio	#states
c432	4	1.054	8	0.676	7
	10	0.752	98	0.426	34
c499	4	0.947	8	0.939	8
	10	0.768	137	0.657	85
c880	4	1.041	8	0.830	7
	10	0.739	94	0.592	53
c1355	4	0.828	7	0.803	7
	10	0.711	107	0.653	94
c1908	4	1.099	8	0.918	7
	10	0.909	202	0.682	80
c2670	4	1.103	8	0.636	6
	10	0.974	246	0.410	52
c3540	4	1.105	8	0.732	7
	10	0.939	202	0.554	50
c5315	4	1.113	8	0.747	7
	10	0.964	156	0.522	58
c6288	4	0.901	7	0.872	7
	10	0.617	44	0.567	38
c7552	4	1.081	8	0.660	6
	10	0.979	194	0.493	50

bit to each code. As can be seen from this table the proposed transformation helps reduce the size of the encoded test sets in this case also.

6. Conclusion

For the situation where a given test set is encoded by a statistical code to reduce its size, we proposed a method to transform a given test set so as to decrease the average length of a codeword. The experimental results presented show that the proposed transformation method can significantly decrease not only the average length of a codeword but also the number of states of the decoder needed to decode the encoded tests.

References

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

[2] V. Iyengar, 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.

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

[4] I. Pomeranz and S. M. Reddy, "On Test Compaction Objectives for Combinational and Sequential Circuits," Proc. VLSI Design Conference., pp. 279-294, Jan. 1998.

[5] I. Hamzaoglu and J. H. Patel, "Test Set Compaction Algorithm for Combinational Circuits," Proc. of International Conference on Computer-Aided Design, pp. 283-289, 1998.