On Combining Pinpoint Test Set Relaxation and Run-Length Codes for Reducing Test Data Volume*

Seiji Kajihara Yasumi Doi
Dept. of Computer Sciences and Electronics
Kyushu Institute of Technology

Lei Li Krishnendu Chakrabarty
Dept. of Electrical and Computer Engineering
Duke University

Abstract

This paper presents a pinpoint test set relaxation method for test compression that maximally derives the capability of a run-length encoding technique such as Golomb coding or frequency-directed run-length (FDR) coding. Before encoding a given set of test patterns, we selectively relax some specified bits of the test patterns. By changing a specified bit with value 1 to a don't-care, two consecutive runs of 0s in the test sequence can be concatenated into a longer run of 0, thereby facilitating run-length coding. This procedure retains the fault coverage of the test set. Since the increase in compression depends on the lengths of the two runs that are concatenated with each bit relaxation, a lookup table, referred to as the gain table, is pre-computed and used during the test set relaxation procedure to maximize the likelihood of increasing the amount of test data compression. The gain table is used to pinpoint the bit positions with value 1, which when relaxed to don't-cares, will yield the most compression. In this way, the given test pattern set is appropriately modified as a preprocessing step before test compression. Experimental results for the ISCAS benchmark circuits show that the proposed method can be used to increase the effectiveness of run-length coding methods for test data compression.

1. Introduction

Test cost is a major component of the manufacturing cost for system-on-chip (SOC) integrated circuits. In particular, test data volume is a major contributor to SOC test cost. Large test data volumes lead to an increase in testing time and the need for multiple ATE channel reloads, either from workstations across a network or from slow hard disks. As a result, a number of techniques have recently been presented in the literature to reduce test data volume, and

thereby reduce test cost [1-15]. One such technique is test compaction, which relies on ATPG techniques to generate a small test set with maximum fault coverage [1, 2, 3]. Techniques to generate test sets with a close-to-minimum number of patterns have been developed for full-scan sequential circuits [2, 3]. For scan-based circuits however, test compaction alone does not provide sufficient reduction in test data volume if the number of scan flip-flops is very large, since test data volume is proportional not only to the number of test vectors but also to the number of scan flip-flops.

Recently, a number of methods have been proposed for reducing test data volume using compression methods. The key idea here is to compress test data using one of several techniques and use a small amount of embedded hardware for on-chip decompression. Test compression methods can be classified in terms of the type of embedded hardware employed for test delivery. For example, when the decompression hardware is based on an LFSR, reseeding is one of the efficient test compression methods [4, 5, 6]. There also exist several methods for compressing test data by encoding test sets using statistical and run-length encoding [7-11]. The decompression hardware for these methods consists of a sequential circuit implementing a finite state machine. Test compression methods that use a combinational circuit as a decompressor have also been proposed in the literature [12, 13]. Finally, a number of methods using an embedded processor for software-based decompression have been proposed, e.g. [20].

Test compression is sometimes based on test patterns generated by ATPG. It is known that unspecified values of the test patterns are very useful to derive higher compression ratio. On the other hand, test generation with leaving unspecified values cannot make the size of the test set be smaller because compaction using reverse order fault simulation [16] or double detection [2] might be useless for the unspecified test patterns. Although high compression ratio can be obtained from a larger test set in general, the resulting test data volume from the large test set may still be larger than that from a compacted test set for the same circuit. Thus if we can compress a highly compacted test set efficiently, test data volume can be reduced.

*The work of S. Kajihara was supported in part by Monbu Kagakusho Grants-in-Aid for Scientific Research 14780228. The work of L. Li and K. Chakrabarty was supported by the National Science Foundation under grants CCR-9875324 and CCR-0204077, and by a graduate fellowship from the Design Automation Conference.

When a highly compacted test set is generated in which every bit of test patterns is specified as either to 0 or 1, there exist values in the test set that can be changed to the opposite logic values without losing fault coverage. One can regards such input values as *don't-care* (X). A method, called maximal compaction, was developed for identifying an X in a test pattern for a fault [1]. A method for identifying the Xs in the entire test set was proposed in [17]. Since there are many combinations of X inputs in general, the method finds one including as many X inputs as possible. It is reported that, even for a highly compacted test set for ISCAS benchmark circuits, approximately half of values are Xs. Since we can treat the Xs of test patterns as well as unspecified values, we can utilize the Xs for test compression.

In this paper, we present a test modification method for test compression that maximally derives the capability of a run-length based encoding such as Golomb encoding or FDR encoding. Before encoding given test patterns, we efficiently change values of some bits of the test patterns using a technique of don't-care identification [18]. This procedure retains the fault coverage of the test set. The bit values to be changed are selected so that two runs in the test patterns are concatenated into one longer run. Since a gain with each concatenation is different depending on the run-length of prefix and tail for the changed bit, we prepare and refer a gain table that gives the number of bits reduced by encoding after concatenation of two runs. According to the gain table, we pinpoint bit positions that produce larger gain for test modification, and apply the don't-care identification such that Xs on the bits are found maximally. The procedure of test modification is iteratively applied to the test patterns. As a result, the given test patterns are modified optimally for test compression. Experimental results for ISCAS benchmark circuits show that the proposed method can compress test sets efficiently.

This paper is organized as follows. In Section 2, we explain techniques used in the proposed method. In Section 3, we explain the proposed method. In Section 4, we present experimental results for ISCAS benchmark circuits and finally conclude this paper in Section 5.

2. Preliminaries

In this section, we explain techniques used in our work.

2.1 Run-length based encoding

As methods for encoding test patterns, FDR (Frequency Directed Run-length) encoding [8] and Golomb encoding [7] is well-known. These methods compress test data by encoding successive 0s in the test patterns with a codeword. In this paper, each set of test patterns before and after encoding are denoted by T_D and T_E respectively.

Table 1(a) and Table 1(b) show an example of code assignments with FDR coding and Golomb coding with group

Table 1 Example of codeword assignment

(a) FDR coding

Group	run-length	Prefix	Tail	Codeword
A1	0	0	0	00
	1		1	01
A2	2	10	00	1000
	3		01	1001
	4		10	1010
	5		11	1011
A3	6	110	000	110000
	7		001	110001
	8		010	110010
	9		011	110011
	10		100	110100
	11		101	110101
	12		110	110110
	13		111	110111

(b) Golomb coding

Group	run-length	Prefix	Tail	Codeword
A1	0	0	00	000
	1		01	001
	2		10	010
	3		11	011
A2	4	10	00	1000
	5		01	1001
	6		10	1010
	7		11	1011
A3	8	110	00	11000
	9		01	11001
	10		10	11010
	11		11	11011

t_1 : 00100

t_2 : 10101

t_3 : 00011

TD : 001001010100011

TE : 100010000101100100

(a) Given tests

(b) Test stream

Fig. 1 Example of FDR coding

m_1 : xffxf

m_2 : ffxxf

m_3 : xffxx

t_1' : x0100

t_2' : 101x1

t_3' : 000x1

(a) A mask set

(b) Tests with Xs

Fig. 2 Don't care identification

size 4, respectively. When a test set in Fig. 1(a) is a given, it is compressed with the FDR encoding as shown in Fig. 1(b). Note that we treat test patterns in the given test set as one stream by concatenating all the test patterns in serial.

The run-length based encoding produces shorter codes, when there are more 0s in the test patterns and the length of each run is longer. Therefore the more unspecified values the test set includes, the lesser the test data

volume is potentially.

2.2 Don't-care identification

Test patterns of a test set generated do not have unspecified values generally because they are specified by random fill or static/dynamic test compaction [19] for the detection of faults other than the target fault. However, some primary input values may be changed to opposite logic values without losing fault coverage. One can regard such input values as don't-cares (Xs). A method for identifying Xs of a generated test set has been proposed in [17]. The method in [17] determines as many Xs as possible in the test patterns of the given test set.

To Xs identified, logic values can be reassigned for the purpose of reducing test data volume. If a reassigned value is the same as the original value, the X is meaningless. Generally the test set with Xs is not uniquely determined, and there are many possible combinations of bit position of the Xs. Desired bit position of the Xs depends on the purpose of the reassignments to the Xs, while the Xs are typically used for test compaction/compression or test power reduction. If it is known that some bits do not have to be changed to Xs, Xs on the other bits can be increased.

A method for placing the Xs on a maximal number of specific bits of a test set has been proposed in [18]. The method allows a user to specify bits for the placement of Xs for each one of the given test vectors. The specific set of bits that should be maximally changed to Xs is specified using a pattern set given together with the given test set. We call the pattern set a *mask pattern set*, and in short a *mask set*. Each mask pattern is associated with a test pattern. An example of a mask set for test set T in Fig. 1(a) is shown in Fig. 2(a) where 'x' means a bit to be changed to an X, and 'f' means a fixed bit, i.e., an X on the bit is not needed. An example of a possible test set is shown in Fig. 2(b).

3. Proposed method

3.1 Motivation

It can be easily imagined that the X identification technique is suitable for test compression of the given test patterns. Especially, when we can specify bit positions to be changed to an X, the test patterns would be modified so as to derive the effect of test compression maximally. For the run-length encoding, since we need more 0s in the test set, we should prepare a mask set such that bit positions with 1 in the given test set are set to 'x', and the others are set to 'f'.

Suppose that the test set in Fig. 1(a) is compressed. The mask set for the test set would be prepared as shown in Fig. 3(a). For the original test stream TD_0 , we consider two possible streams, TD_1 and TD_2 in Fig. 3(b), which may be obtained from possible test sets with Xs. When we compress

m_1 : f f x f f	TD_0 : 001001010100011
m_2 : x f x f x	TD_1 : 00x0010101000x1
m_3 : f f f x x	TD_2 : 0010010x01000x1

(a) A mask set (b) Test stream

Fig. 3 Example of FDR coding

TE_0 : 100010000101100100
TE_1 : 101101011010
TE_2 : 1000100010011010

Fig. 4 Compressed test streams after assigning 0 to Xs

TD_1 with FDR encoding according to Table 1(a), compressed test data TE_1 as shown in Fig. 4 is obtained because by assigning value 0 on the Xs in TE_1 . The gain obtained by modifying TD_0 to TD_1 , which is calculated by $|TE_0| - |TE_1|$, is 4 bits, where $|TE_i|$ means the number of bits of TE_i . On the other hand, from TD_2 , TE_2 is obtained and no gain is obtained by the modification. Although the number of Xs in TD_1 and TD_2 is the same, the gains are different. It is observed that there is a bit in the original test set that produces larger gain by changing it to an X, while there is a bit in the original test set that does not produce large gain by changing it to an X.

3.2 Gain table

By changing a bit value, two runs in the test patterns are concatenated into one longer run. As shown in the above example, a gain with each concatenation is different depending on the run-length of 0s in prefix and tail for the changed bit. In the case of TD_1 in Fig. 3(b), a run with length 2 was concatenated with another run with length 2. As a result, a run with length 5 was created by assigning 0 to the X. From the codeword in Table 1(a), since both codes of the run with length 2 and the run with length 5 consist of 4 bits, the gain by changing the bit was 4 bits. On the other hand, in the case of TD_2 in Fig. 3(b), two runs with length 1 were concatenated. But no gain was obtained because two runs with length 1 make a 2-bit code twice and the concatenated run with length 3 makes a 4-bit code.

Based on the table for code assignments in Table 1, we can compute the gain for each bit whose value is 1 in the given test streams when the bit value is changed to 0. The gain is independent of generation of the test set and just computed from the prefix run-length and the tail run-length for the bit with value 1 statically. Hence we prepare and refer a 2-dimensional table that gives the number of bits reduced by encoding after concatenation of two runs. We call it a *gain table*. The gain table for FDR coding is given in Table. 2. The line number and the column number indicate the

Table 2 Gain table for FDR coding

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	0	2	2	2	0	2	2	2	2	2	2	2	0	2	2
1	0	0	2	2	0	0	2	2	2	2	2	2	0	0	2	2
2	2	2	4	2	2	2	4	4	4	4	4	2	2	2	4	4
3	2	2	2	2	2	2	4	4	4	4	2	2	2	2	4	4
4	2	0	2	2	2	2	4	4	4	2	2	2	2	2	4	4
5	0	0	2	2	2	2	4	4	2	2	2	2	2	2	4	4
6	2	2	4	4	4	4	6	4	4	4	4	4	4	4	6	6
7	2	2	4	4	4	4	4	4	4	4	4	4	4	4	6	6
8	2	2	4	4	4	2	4	4	4	4	4	4	4	4	6	6
9	2	2	4	4	2	2	4	4	4	4	4	4	4	4	6	6
10	2	2	4	2	2	2	4	4	4	4	4	4	4	4	6	6
11	2	2	2	2	2	2	4	4	4	4	4	4	4	4	6	6
12	2	0	2	2	2	2	4	4	4	4	4	4	4	4	6	6
13	0	0	2	2	2	2	4	4	4	4	4	4	4	4	6	6
14	2	2	4	4	4	4	6	6	6	6	6	6	6	6	8	6
15	2	2	4	4	4	4	6	6	6	6	6	6	6	6	6	6

length of the prefix bit and the tail bit, respectively. When creating a mask set, according to the gain table we pinpoint bit positions with large gain. Thus we can modify the given test patterns so as to be compressed maximally.

3.3 Pinpoint bit modification

In order to avoid identifying Xs on bits with small gain, we define a threshold of gain, g_{min} . We check the gain of each bit with value 1 according to the gain table and create a mask set so that if the gain of a bit is greater than or equal to g_{min} , then put 'x' on the bit, otherwise put 'f'. We initially set g_{min} to 8, and we iterate a procedure to identify Xs and to assign 1 to the X as long as Xs are found. If the test set cannot be modified for the given, we decrease g_{min} and continue the procedure.

We give an outline of the procedure below.

Step 0: Set a given test set to T , and set g_{min} to 8.

Step 1: From test set T and the gain table, create a mask set such that bits whose gain is greater than or equal to g_{min} are set to 'x', and the other bits are set to 'f'.

Step 2: identify Xs on bits in T specified by the mask set.

Step 3: For each X identified, if the gain by assigning 0 to the X is still greater than or equal to g_{min} , assign 0 to the X. Otherwise, reassign 1 to the X. Let T' be the resulting test set.

Step 4: If $T = T'$, then set g_{min} to $g_{min} - 2$. If $g_{min} < 0$, then stop.

Step 5: Set T' to T , and go to Step 1.

Although the don't-care identification is performed based on the test set and the mask set, we don't assign 0 not to all Xs found. This is because, after a value of a bit

```

TD0 : 001001010100011
gain : 004002000200022
mask set_0 : ffxfffffffffffff

TD0' : 00X001010100011
TD1 : 000001010100011
gain : 000000000200022
mask set_1 : ffffffffffxfffxx

TD1' : 000001010X000X1
TD2 : 000001010000011
gain : 000000000000002
.....
.....

```

Fig.5 An example of pinpoint test relaxation

was changed to 0, the gains on the adjacent bit with "1" may vary. The gain by two bit change is not equal to the sum of the gains by individual bit change. This procedure is based on a greedy algorithm in order to save computing time, i.e., we try to change the bits with larger gain obtained with individual bit change. Hence the optimum solution on the modified test patterns is not guaranteed.

In Fig. 5 we illustrate an example of the procedure. *Example:* We assume that a test set TD_0 is given. The gain of each bit is calculated as shown in the second line. As there is no bit whose gain is greater than 4, we start explanation from the case of $g_{min} = 4$. The mask set for TD_0 is created in $mask_set_0$. Suppose that a test set TD_0' is obtained from TD_0 and $mask_set_0$. Then, value 0 is assigned to the X in TD_0' , and resultant test set becomes TD_1 . Depending on the gain of each bit in TD_1 , $mask_set_1$ is created, where $g_{min} = 2$. We suppose that two Xs in TD_1 are identified as shown in TD_1' . While we assign 0 to the left X, we don't assign 0 to the right X because the gain is reduced to 0 that is less than g_{min} after the assignment to the left X. The procedure would still continue after obtaining TD_2 , until the test set cannot be modified. We omit the explanation here.

4. Experimental results

We implemented the proposed method on a PC (PentiumIII 1.80GHz, 256MB) using C programming language, and applied it for fully scanned ISCAS'89 benchmark circuits. As the initial test sets, we prepared highly compacted test sets [2], and used a 200x200 matrix as the gain tables. Table 3 shows experimental results for FDR coding and Golomb coding with $m=4$. The second column and the third column list the number of test patterns in each test set and the number of bits in each test pattern to be compressed. We refer to volume of the original test data as TD , and volume of the compressed test data as TE . The columns headed "FDR" and "Golomb" present the number of

bits compressed with FDR coding and Golomb coding, respectively. Although the proposed method is available for Golomb coding, it brings better results for FDR coding.

We analyzed the procedure and data compressed with FDR coding. Table 4 presents some data for discussion. The columns headed “#1s” and “#1to0s” gives the number of bits whose value is 1 in the original test set, and the number of bits whose value is changed from 1 to 0 by the proposed method. The following five columns present the number of iterations for each g_{min} in the procedure. The last column gives run time. The proposed method could changes approximately 75% of 1s of the given test set to 0. As the most of the changes were done in $g_{min}=2$, we would consider additional criterion for pinpointing bits in $g_{min}=2$ as a future work. While the iterations of the procedure were more

than twenty times, it was done still in practical computing time.

Table 5 gives comparisons of the proposed method with others. The column of “nonpinpoint” shows the results without using the gain table, i.e., the procedure tries to change all 1s in the test set to 0s without notice. The following columns gives results of EFDR coding [9], variable-length Huffman coding [14], RESPIN++ [15], and XOR network [12]. For many circuits, the proposed method showed the highest performance on compression. For s35932, however, our method could not work very well, because the number of bits with value 1 in the test set was greater than that with value 0. For such a circuit, we should change value 0 to 1 in the test set, and compress it based on the runs with 1.

5. Conclusion

In this paper, we presented a test modification method for test compression that maximally derives the capability of a run-length based encoding such as FDR encoding. Before encoding given test patterns, we changed values of some bits of the test patterns using the technique of don't-care identification that retains the fault coverage of the test set in any way. The bit values to be changed were selected such that two runs in the test patterns are concatenated into one longer run. In order to maximize the gain obtained by the concatenation, we prepared and referred a gain table that gives the number of bits reduced by encoding after

Table 3 Compression results with FDR coding and Golomb coding(m=4)

circuit	#tests	#inputs	TD [bits]	FDR [bits]	Golomb [bits]
s5378	100	214	21400	10490	12454
s9234	111	247	27417	15454	18096
s13207	235	700	164500	22840	52890
s15850	97	611	59267	18466	26077
s35932	12	1763	21156	18072	26609
s38417	87	1664	144768	64078	77711
s38584	114	1464	166896	59262	74991

Table 4 Detailed Results for FDR coding

circuit	#1s [bits]	#1to0s [bits]	#iterations					time [sec]
			$g_{min}=8$	$g_{min}=6$	$g_{min}=4$	$g_{min}=2$	$g_{min}=0$	
s5378	11333	8489	1	2	3	29	3	31
s9234	15154	10727	1	2	4	17	4	90
s13207	72141	67555	3	4	4	133	4	1184
s15850	28748	24212	1	2	4	49	4	412
s35932	13457	5595	2	2	3	17	7	501
s38417	94695	78147	1	2	4	77	8	2254
s38584	67733	54357	2	3	6	42	4	1216

Table 5 Comparisson of compression results

circuit	ours	nonpinpoint	EFDR [9]	VIHC [14]	RESPIN++ [15]	XORnet [12]
s5378	<u>10490</u>	10572	11419	11516	17332	N/A
s9234	<u>15454</u>	15846	21250	17736	17198	N/A
s13207	<u>22840</u>	24736	29992	27737	26004	25344
s15850	<u>18466</u>	19906	24643	30271	32226	22784
s35932	18072	17550	<u>5554</u>	9458	N/A	7128
s38417	<u>64078</u>	66020	64962	74938	89132	89356
s38584	59262	61960	73853	85674	63232	<u>38976</u>

concatenation of two runs. Then, we pinpointed bit positions, and apply the don't-care identification. The procedure of test modification was iteratively applied to the test patterns. Experimental results for ISCAS benchmark circuits showed that the proposed method could compress test sets efficiently.

Reference

- [1] I. Pomeranz, L. N. Reddy and S. M. Reddy, "COMPACTEST: A Method to Generate Compact test Sets for Combinational Circuits," *IEEE Trans. CAD*, pp. 1040-1049, Jul. 1993.
- [2] S. Kajihara, I. Pomeranz, K. Kinoshita and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits," *IEEE Trans. CAD*, pp.1496-1504, Dec. 1995.
- [3] I. Hamzaoglu and J. H. Patel, "Test Set Compaction Algorithms for Combinational Circuits," *Proc. Int'l Test Conf.*, pp. 283-289, Oct. 1998
- [4] S. Hellebrand, J. Rajske, 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 Trans. Comp.*, pp. 223-233, Feb. 1995.
- [5] S. Hellebrand, H.-G. Liang, H. -J. Wunderlich, "A mixed-mode BIST scheme based on reseeded of folding counters," *Int'l Test Conf.*, pp.778-784, Oct. 2000.
- [6] C. V. Krishna, A. Jas, and N. A. Tuba, "Test Vector Encoding Using Partial LFSR Reseeding," *International Test Conf.*, pp. 885-893, Oct. 2001.
- [7] A. Chandra and K. Chakrabarty, "Test Data Compression for System-on-a-Chip Using Golomb Codes," *VLSI Test Symposium*, pp. 113-120, April 2000.
- [8] A. Chandra and K. Chakrabarty, "Frequency-directed Run-length (FDR) Codes with Application to System-on-a-chip Test Data Compression," *Proc. VLSI Test Symp.*, pp. 42-47, April 2001.
- [9] A. El-Maleh, R. Al-Abaji, "Extended Frequency-directed Run-length Codes with Improved Application to System-on-a-Chip Test Data Compression," *Int'l Conf. Electronics, Circuits and Systems*, pp. 449-452, 2002.
- [10] A. Jas, J. Ghosh-Dastidar, and N. A. Tuba, "Scan Vector Compression/Decompression Using Statistical Coding," *VLSI Test Symposium*, pp. 114-120, April 1999.
- [11] H. Ichihara, K. Kinoshita, I. Pomeranz, S. M. Reddy, "Test Transformation to Improve Compaction by Statistical Encoding," *International Conf. on VLSI Design*, pp.294-299, Jan. 2000.
- [12] I. Bayraktaroglu and A. Orailoglu, "Test Volume and Application Time Reduction Through Scan Chain Concealment," *Design Automation Conference*, pp.151-155, June 2001.
- [13] S. M. Reddy, K. Miyase, S. Kajihara and I. Pomeranz, "On Test Data Volume Reduction for Multiple Scan Chain Designs," *VLSI Test Symposium*, pp. 103-108, April 2002.
- [14] P. T. Gonciari, B. Al-Hashimi, N. Nicolici, "Improving compression ratio, area overhead, and test application time for system-on-a-chip test data compression/decompression," *Design Automation and Test in Europe Conf.*, pp. 604-611, 2002.
- [15] L. Schafer, R. Dorsch, H. -J. Wunderlich, "RESPIN++ - Deterministic Embedded Test," *European Test Workshop*, pp.37-44, May 2002.
- [16] M. Schulz, E. Trischler, and T. Sarfert, "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System," *IEEE Trans. on CAD.*, pp. 126-137, Jan. 1988.
- [17] S. Kajihara, and K. Miyase, "On Identifying Don't Care Inputs of Test Patterns for Combinational Circuits," *Proc. Int'l Conf. on Computer Aided Design*, pp. 364-369, Nov. 2001.
- [18] K. Miyase, S. Kajihara, I. Pomeranz and S. M. Reddy, "Don't care identification on specific bits of test patterns," 2002 International Conference on Computer Design, pp. 194-199, Sep. 2002.
- [19] P. Goel, and B. C. Rosales, "Test Generation and Dynamic Compaction of Tests," *Digest of Papers 1979 Test Conf.*, pp. 189-192, Oct. 1979.
- [20] A. Jas and N. A. Tuba, "Using an embedded processor for efficient deterministic testing of system-on-a-chip," *Int'l Conf. on Computer Design*, pp.418-423, 1999.