A Multi-Dimensional Pattern Run-Length Method for Test Data Compression

Conference Paper *in* Proceedings of the Asian Test Symposium · January 2009

DOI: 10.1109/ATS.2009.49 · Source: DBLP

Lung-Jen Lee
Army Academy ROC
17 PUBLICATIONS **70** CITATIONS
SEE PROFILE

Wang-Dauh Tseng
Yuan Ze University
27 PUBLICATIONS **106** CITATIONS
SEE PROFILE

Rung-Bin Lin
Yuan Ze University
86 PUBLICATIONS **432** CITATIONS
SEE PROFILE

A Multi-Dimensional Pattern Run-Length Method for Test Data Compression

Lung-Jen Lee[1], Wang-Dauh Tseng, Rung-Bin Lin and Chen-Lun Lee

Department of Computer Science & Engineering, Yuan Ze University
135 Yuan-Tung Road, Chung-li, Taiwan, 32003, ROC
e-mail:[1]s959101@ mail.yzu.edu.tw

Abstract—**This paper presents a run-length-based compression method considering dimensions of pattern information. Information such as pattern length and number of pattern runs is encoded to denote the compression status. The decoder is simple and requires very low hardware overhead. Significant improvements are experimentally demonstrated on larger ISCAS'89 benchmarks.**

Keywords-test data compression; pattern run-length; code-based testing; SOC; ATE

I. INTRODUCTION

As the process technology successively scaled down within a nanometer era, more and more functions have been crammed into a smaller device. Modern SOC designs integrate several modules and intellectual property (IP) cores into a single chip, in which a rapidly growing number of transistors are used. Although increasing integration produces robust designs, much more faults are created accordingly. To detect them, immense test data volume with longer scan chains are required and have a profound effect on testing time for the stored pattern testing of SOC. Now, test data volume has been recognized as a major contributor to the cost of manufacturing testing for integrated circuits [1]-[4]. Many new techniques have been proposed to reduce test data volume so as to save memory cost and improve the transmission efficiency between Tester (ATE) and SOC. One solution to this problem is to use compression techniques to reduce the volume of test data.

Test data compression is a non-intrusive method, which condenses test data to a smaller size to save the ATE memory. During testing, the compressed data is first transferred through test channels to SOC, decompressed losslessly by the on-chip decoder and then serially scanned in the scan chain. The output response, on the other hand, can use a lossy compaction technique with little impact on fault coverage.

Techniques for test data compression can essentially be classified into three different categories: Code-based schemes, Linear-decompression-based schemes and Broadcast-scan- based schemes [5]. Among them, code-based scheme is a quite popular-used one. Code-based scheme encodes test data by a number of codewords according to specific properties embedded in corresponding bit-strings of the test data. The Huffman code [6] is proven an optimal statistical code, which provides the shortest average codeword length among all uniquely decodable variable length codes, but suffers from its exponentially grown decoder size. Many extending works such as Huffman-coding-based SHC [7], OSHC [8] and VIHC [9] are also proposed. Selective Huffman Coding [7] is an efficient test-data compression method with low hardware overhead, which merely encodes the most frequently occurring symbols. A 9C technique [10] uses exactly nine codewords aiming at pre-computed data of intellectual property cores in SOC. It is flexible in utilizing both fixed- and variable-length blocks. In addition, run-length-based compression method is also a well-known code-based scheme, which encodes runs of repeated values. Examples include GOLOMB [11], FDR [12], ALT-FDR [13], PRL [14], EFDR [15]. ALT-FDR [13] is a variable-to-variable length compression method, derived from FDR [12], both of them use the same encoding manner. PRL [14] is also an efficient approach, which compresses consecutive patterns in an innovative manner. The compression is data-independent and the program for decompression is very small and simple, thereby allowing fast and high throughput to minimize test time. Recently, BM [16], a block merging technique, was proposed by El-Maleh in which good compression effect was achieved by encoding runs of fixed-length blocks, only the merged block and number of block merged are recorded. The RL-HC [17] combines the above two well-known methods: run-length-based and Huffman coding for scan testing to reduce test data volume, test application time, and scan-in power.

In this paper, a multi-dimensional pattern run-length method (MD-PRC) encoding runs of variable-length patterns is proposed. We first introduce the basic concept of pattern run-length method for the methods 1D-PRC and 2D-PRC. In 2.5D-PRC, an extended bit added in the back of codeword is used to further improve the compression effect. In 3D-PRC, the concept of "times" is also introduced. Experimental results show the proposed MD-PRC is an excellent compression method. The rest of this paper is organized as follows: Section II analyses and presents the proposed method of MD-PRC through 1D, 2D, 2.5D and 3D respectively. The decompression architecture is also introduced. To evaluate its effectiveness, in Section III, experiments are implemented on six large ISCAS'89 benchmark circuits. Finally, we conclude this work in Section IV.

II. Proposed Method

MD-PRC is a run-length-based compression method which encodes runs of compatible patterns in a test set. For example, the string "AAABCC" can be encoded by the codeword "3A1B2C" in which the number "3", "2", and "1" denote the number of runs for characters "A", "B" and "C" respectively. In the proposed "Multi-dimensional pattern run-Length" method, each codeword is composed of a Control Code (C) followed by an encoded pattern. Control Code (C) is a t-bit binary code used to indicate the compression status of the encoded pattern. Since $0 \leqq C \leqq 2^t-1$, the encoded number can vary from 1 to C+1. Dimensions of information can be stored in Control Code (C), such as pattern length, number of pattern runs and so on to extend the length of the compressed bit-string. Within a certain extent, this can help improve the compression effect. For example, in one-dimensional pattern run-length compression method (1D-PRC), Control Code denotes only the number of pattern runs and the pattern length is assumed fixed. While in 2D-PRC, both number of pattern runs and pattern length are variables and can be specified by the same Control Code.

To further improve the compression effect in 2D-PRC, in the method "2.5D-PRC", an extended bit can be added in the end of codeword to indicate a doubling of the codeword to save the required codeword length and extend the length of the compressed bit-string. The method "3D-PRC" introduces the concept of "times" as the third dimension for encoding a target bit-string. 3D-PRC is especially good at compressing test data for those industrial-scale circuits. We will describe them as follows.

A. *One-Dimensional Pattern Run-Length Compression Method (1D-PRC)*

In One-Dimensional Pattern Run-Length compression method, the pattern length is set fixed and the Control Code (C) only denotes the number of pattern runs. As previously mentioned, the number of pattern runs can vary from 1 to C+1. Therefore, the length of the compressed bit-string (L), when using the compression method "1D-PRC" with the fixed pattern length P, will be equal to $P \times (C + 1)$ bits, i.e., $L = P \times (C + 1)$.

Fig.1 shows an example, where the pattern length is assumed to be 3 bits and 2 bits for Control Code (C) (i.e. t = 2), consequently, the value for C can be 00, 01, 10, and 11 and the number of pattern runs can be specified from 1 to 4 respectively. As shown, the bit-string "1XX 10X X0X 1XX"can be encoded as four runs of the pattern "10X", where "four" can be specified by the Control Code "11" and, consequently, the compressed codeword is "1110X". Although the 1D-PRC compression method is simple, it is

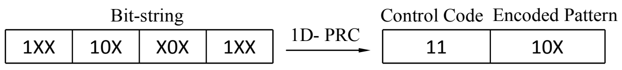

Figure 1. An example for 1D-PRC.

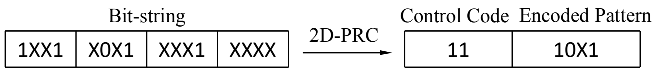

Figure 2. An example for 2D-PRC.

data 10X 10X 01X 10X as an example, the first six bits can be encoded as 01 10X, however, for the rest test data, the compression is not efficient. However, if the pattern length can be flexibly configured from 3 to 2, then the codewords for the rest data can be reduced to 10 01 and the compression effect can be effectively improved. This, therefore, gives rise to the method "2D-PRC".

B. *Two-Dimensional Pattern Run-Length Compression Method (2D-PRC)*

In the method "2D-PRC", both the number of pattern runs and the pattern length are encoded by Control Code (C). Obviously, they will have the same value and be equal to C + 1. Therefore, the length of the bit-string compressed by the method "2D-PRC" will be equal to $(C + 1)^2$ bits, i.e., $L = (C + 1)^2$. Fig. 2 shows a simple compression example to illustrate the format for the codeword used in 2D-PRC, where the bit-string "1XX1 X0X1 XXX1 XXXX" is encoded by the Control Code"11" followed by the encoded pattern "10X1".

In the following, another example is given where a simplified test set is compressed adopting a 2-bit Control Code (t=2). Assume the simplified test set is X1XX0111X1110X1X01X11XX10XX0X. Initially, the largest C (C=11) will be firstly attempted for compression. As $L = (C + 1)^2$, the first 16-bit bit-string will be considered at first to see if it meets the constraint of containing 4 4-bit compatible patterns. If not, then C will be decreased by 1 (C=10) and repeat the above steps for the next try. In this way, compressions will be iteratively conducted without stop until all bits in the target bit-string are compressed. In this example, we first use C=11 as the Control Code, during compression, all Xs in the target sub-string may be accordingly assigned to "0" or "1" to achieve the compatibility among them. Hence, we compress the first 16-bit bit-string by the codeword "110111" in which the encoded pattern is "0111". In the same way, we compress the remaining 13 bits (01X11XX10XX0X) in the test set and obtain the codewords: 0101 and 1010X for sub-strings 01X1 and 1XX10XX0X, respectively. Consequently, the targeted test set is compressed into the codeword: "110111 0101 1010X". Notice that there still exists an "X" left in the codeword which can be filled as "0" or "1" randomly during testing to help detect the non-modeled faults.

As previously described, the control code length "t" is an important parameter for MD-PRC since a larger t may offer a bigger range of C values indicating a large range for considerations on both pattern length and the number of pattern runs. For example, t = 1 gives C a range from 0 to 1,

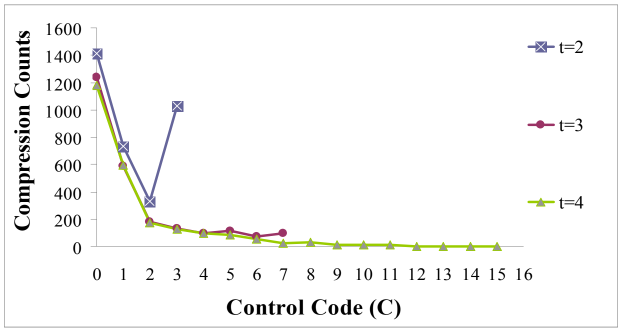

Figure 3. Relationship between Control Code and compression counts under varying t for s5378.

so the length of the encoded pattern can be at most 2 bits. For t = 2, a longest pattern length up to 4 can be considered. However, an improper increase of t may not only lead to a longer computation time but also degrade the compression effect due to the longer codewords and, consequently, offset the advantage of the compression. Therefore, a proper selection of t is important for MD-PRC since it closely relates the compression performance. To explore the relationship between Control Code (C) and the compression counts (total number of codewords) under varying t, an experiment is implemented on the circuit s5378 and the results are shown in Fig. 3. As shown in Fig. 3, there display three curves for t = 2, 3 and 4 respectively from the top to the bottom. Axial "X" shows different C values scaled from 0 to 15 and Axial "Y" shows corresponding compression counts. As can be observed from the three curves that the total compression counts decrease as C increases except an abrupt increase occurred between C = 2 and C = 3 (pattern length = 3 ~ 4) on the curve t = 2. It is because that, as the pattern length grows, a gradually fewer number of bit-strings can be found qualified for compression. While the steep increase on the curve t = 2 indicates that more than 1000 sub-strings are compressed adopting C = 3 (pattern length = 4), but such a steep increase doesn't show up on curve t = 3, only a smooth increase occurred between C = 6 and C = 7. This implies that the curve t = 2 doesn't offer enough choices of pattern lengths for compression. In contrast, when t = 4 is applied, fewer compressions are made as C increases beyond 8. However, extending Control Code from t = 3 to t = 4 may conversely negate the compression effect due to an additional bit in each codeword. Consequently, adopting t = 3 is the best choice for compressing the circuit s5378.

C. Minimal Pattern Length

As previously stated, $0 \leqq C \leqq 2^t-1$, meaning that a t-bit Control Code can, at most, compress patterns with lengths varying from 1 to C + 1. Take t = 2 as an example, the longest compressible length of pattern is 4. However, depending on the specific property of each test set, the most efficient range of compression can vary. We hereby introduce the concept of Minimal Pattern Length (MinLen)

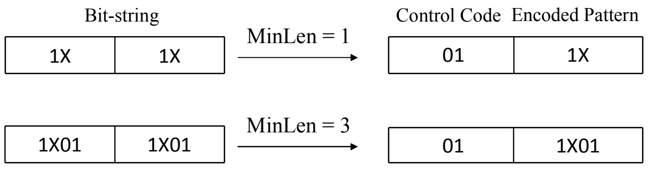

Figure 4. Examples of 2D-PRC using different MinLens.

to be the minimal pattern length, from which all other compressible pattern lengths are based. For example, if we set the minimal pattern length (MinLen) as 1, 4 different pattern lengths ranging from 1 to 4 will be considerable for compression. If we set the minimal pattern length (MinLen) as 3, the compressible range of pattern lengths will be changed to 3 ~ 6 bits. This can help improve the compression efficiency by using a shorter Control Code to cover the expected compression range rather than extending the length of Control Code. In this way, the relationship among the total length of the compressed bit-string, pattern length and the pattern runs can be expressed as L= (C + MinLen) × (C + 1), in this example, the total length of the encoded bit-string (L) becomes (C+3) × (C + 1) instead of the original $(C+1)^2$. Fig. 4 shows an example where the minimal pattern length is 3 and the length of Control Code is 2, the total length of the compressed bit-string is 8 bits instead of the original 4 bits. Fig. 5 presents the compression flow for the 2D-PRC method. As shown, appropriate ranges from 1 to i and from 1 to j are set for the length of Control Code (t) and MinLen (m) respectively. The best combination of (t, m), is then determined by exhaustively performing compressions on the target bit-string adopting different (t, m) combinations and updates for the currently best compression effect. Note that the setting for initial values of i and j will directly affect the required computation time, as experimentally evidenced, beyond a certain (t, m), the compression effect will start to decline.

D. 2.5 -Dimensional Pattern Run-Length Compression Method (2.5D-PRC)

In MD-PRC, an extended bit "0" will be added at the end of each codeword to denote an end of a codeword or a boundary between adjacent codewords. We call it "End

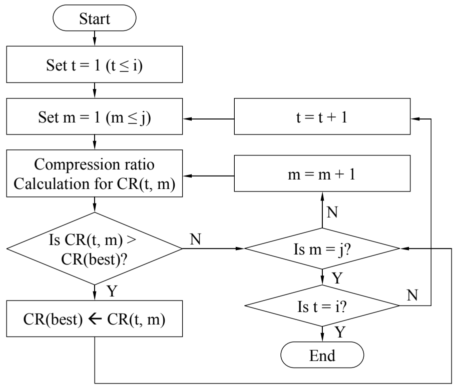

Figure 5. The compression flow for 2D-PRC.

Test Data	
1XX10XX0X	101XX1101

↓

Control Code	Encoded Pattern	Copy Code	End Code
10	101	1	0

Figure 6. An example for 2.5D-PRC.

Code". In 2.5D-PRC, an extended bit "1" could be added at the end of some codewords to declare "another repetition of the same codeword", which is efficient to save the length of the resulted codewords if the original two adjacent codewords are compatible. We call it "Copy Code". Since Copy Code is added in the end of the codeword but not another dimension indicated by Control Code, we thus called this method "2.5D-PRC ". Take the test data "1XX10XX0X 101XX1101 as an example, when t = 2, Control Code = 10 and MinLen = 1 are applied, the original codeword is 1010X 10101. We find there exist two compatible codewords connected in series, according to rule of 2.5D-PRC, the Copy Code "1" can be added at the back of the first codeword to denote "repeating codeword", so the resulting codeword becomes 1010110. Note that the original X in the originally first codeword has been specified to 1, as shown in Fig. 6.

E. Three-Dimensional Pattern Run-Length Compression Method (3D-PRC)

Test data used for testing the industrial-scale circuits often contains a large amount of Xs (sometimes more than 99%). Hereby, we propose a compression method purposely used for compressing test data for the industrial-scale circuits called 3D-PRC. Following the format in 2D-PRC, in 3D-PRC, another dimension "times" is added and embedded in Control Code. In other words, in 3D-PRC, concepts of pattern runs, pattern length and times of encoded bit-string are considered and specified by Control Code. In this way, except pattern length, which will vary with the assumed MinLen, both pattern runs and times are equal to (C + 1) and the relationship among the total length of the compressed bit-string, pattern runs, pattern length and times can be expressed as $L = (C + MinLen)(C + 1)^2$. As shown in Fig. 7, in this example, MinLen is assumed 1, adopting C = 10, all the three parameters: pattern runs, pattern length and times of encoded bit-string are all equal to 3 and the representing bit-string has length of $3 \times 3 \times 3 = 27$(bits). A similar concept as in the method 2.5D-PRC, an extended bit can further be added at the end of the codeword by 3D-PRC and become 3.5D-PRC.

MinLen = 1

Control Code	Encoded Pattern
10	1X0

⟶

1X0	1X0	1X0
1X0	1X0	1X0
1X0	1X0	1X0

Figure 7. An example for 3D-PRC.

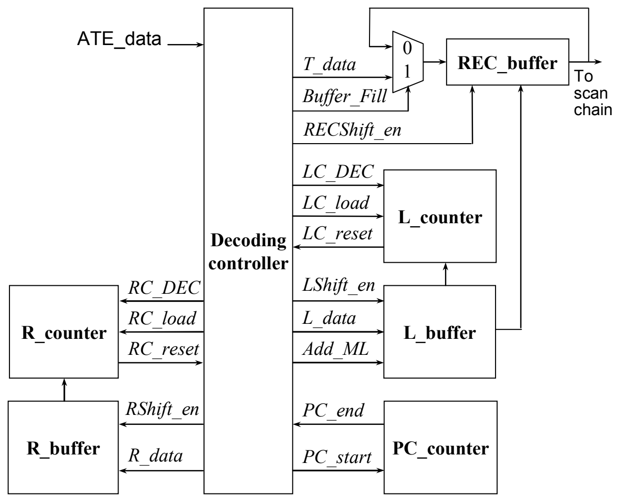

Figure 8. The decoder architecture for 2.5D-PRC.

F. The Decompression Architecture

In this section, the decoder for 2.5D-PRC will be introduced. The main components include three counters, three buffers and a decoding controller, as shown in Fig. 8. We will describe them as follows.

Decoding controller: It is a ten-state Finite State Machine (FSM) with 4 inputs and 13 outputs, by which the entire decompression flow is under control.

PC_counter: A t-bit down counter, by which Control Code is sent to L_buffer and R_buffer are monitored. Whenever PC_start initiated, the value in PC_counter will be reset to t.

L_counter: A down-counter which receives the value of the pattern length from L_buffer and monitors the data shifted from REC_buffer to scan chain.

R_counter: A down-counter which stores the number for pattern runs and monitors the pattern loading process from REC_buffer to the scan chain.

L_buffer: Store the value of the resulted pattern length derived by adding the values of the present pattern length and the Minimum Pattern Length.

R_buffer: Store the number of pattern runs received from Decoding Controller.

REC_buffer: A length-reconfigurable buffer that stores the encoded pattern data before its sending to the scan chain.

In addition, a MUX is used between the Decoding Controller and the REC_buffer which determines the input data either from Decoding Controller or repeat a run of the present data.

The decompression flow can be briefly illustrated as follows. As shown in Fig. 9, S0 launches the EN signal to ATE to read in a codeword from ATE_data. S1 reads Control Code. S2 adds current pattern length with Minimum Pattern Length to obtain the target pattern length. S3 reads in the pattern length derived in S2. S4 shifts in the encoded pattern to REC_buffer bit by bit until L_counter is decreased to 0. S5 loads in R_counter the value indicating the number of pattern runs. S6 reads in the pattern length again. S7 outputs

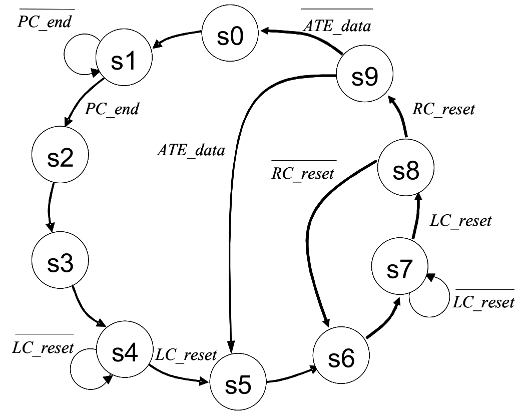

Figure 9. State diagram of the FSM during decompression.

a pattern from REC_buffer to the scan chain. S8 checks the signal RC_reset, if RC_reset is 0 then go to S6 repeating output data from REC_buffer to scan chain, else, goes to S9. S9 checks ATE_data. If it is 0 then goes to S0 for the next codeword, else, goes to S5 to repeat sending data.

III. Experimental Results

We have implemented experiments on six large ISCAS89 benchmark circuits adopting the test data generated by Mintest [13] ATPG with dynamic compaction. The compression effect is evaluated by the compression ratio which is defined as $CR\% = (|T_D| - |T_E|)/|T_D| \times 100\%$ where $|T_D|$ is the size of the test set and $|T_E|$ is the size of compressed test set.

At first, we explore the impact of varying t's (length of Control Code) on the compression effect by applying the method 2D-PRC on the circuit s5378. As shown in Fig. 10, the compression ratio increases with the increase in t. It reaches the peak at t = 3 and then decreases as t continues to grow. For the compression adopting a smaller t, the resulting low compression effect is suffering from the shortage in choices of pattern lengths, while for those adopting larger t's, compression effect is negatively affected by the long Control Code in each codeword.

Exploration on the optimal MinLen value under a certain t is also conducted on the circuit s5378 and the resulting

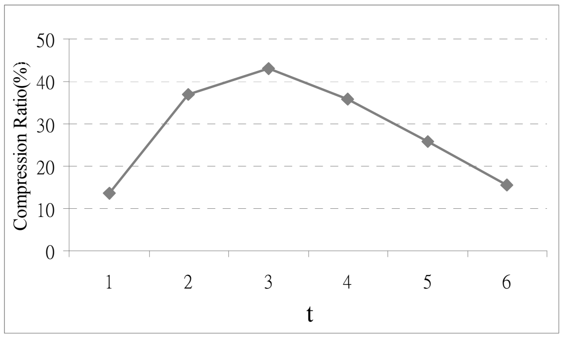

Figure 10. Compression ratios under different t for circuit s5378.

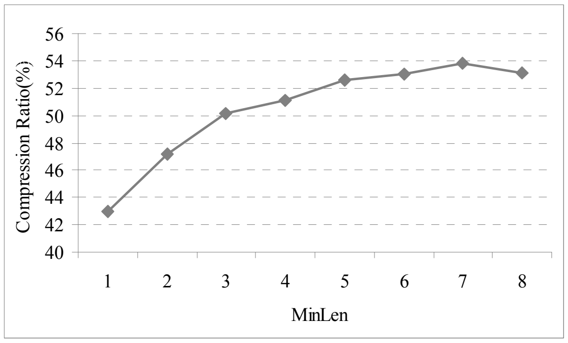

Figure 11. Comparisons on CRs with varying MinLens under t=3.

compression ratios are presented in Fig. 11, where, continue the above experiment, t is assumed 3 and the MinLen values varies from 1 to 8. As shown, the best compression ratio occurs at MinLen = 7, implying that the best compression occurs at the pattern lengths ranging from 7 to 14.

The following experiment compares the effectiveness among the methods 2D-PRC, 2.5D-PRC and 3D-PRC. In Table I, the compression ratios of 2D-PRC, 2.5D-PRC and 3D-PRC are presented in the second, the third and the fourth columns respectively with their (t, m) values at the right. As shown, the method 2.5D-PRC can achieve a better compression effect up to 64.49 averagely for the six large ISCAS'89 benchmark circuits.

Table II presents the required computation time for the proposed 2-D, 2.5-D and 3-D methods. As shown, the proposed methods are all very efficient in computation. Finally, with the above conclusions, we compare The proposed 2.5D-PRC with other coding-based run-length compression methods such as GOLOMB [11], FDR [12], ALT-FDR [13], EFDR [15], Huffman-coding-based SHC [7], VIHC [9], RL-HC[17], 9C [10] and BM [16]. The comparison results are presented in Table III, as can be seen, the proposed MD-PRC outperforms other methods in most cases. To show the adaptability of our method to today's practical-scale circuits, we conduct 3D-PRC on two practical-scale circuits with FFs more than 10k and the care-bit rate less than 1%. As shown in Table IV, the circuit name "3M99" represents it has a test size roughly equal to 3M bits and the percentages of don't-cares around 99%. It can be observed from the results in Table IV that the method 3D-PRC outperforms others. Comparing with the results in Table I, 3D-PRC shows a better compression for the practical-scale circuits than for the ISCAS'89 benchmark circuits, which shows that the 3D-PRC method is especially good at compressing large scale circuits.

IV. Conclusion

MD-PRC is motivated by the observation that repeated compatible patterns can be compressed by recording the pattern runs and the encoded pattern. The methods including 1D-PRC, 2D-PRC, 2.5D-PRC, 3D-PRC are introduced and analyzed. The decompression architecture is simple. Results

TABLE I. EXPERIMENTAL RESULTS FOR THE PROPOSED 2D, 2.5D AND 3D METHODS

Circuit	2D			2.5D			3D		
	CRs	t	MinLen	CRs	t	MinLen	CRs	t	MinLen
s5378	54.02	3	10	54.63	1	5	43.31	1	4
s9234	52.30	3	3	53.20	2	4	39.98	1	2
s13207	85.15	5	6	86.01	3	5	82.72	3	4
s15850	66.65	4	6	69.99	2	5	58.43	2	3
s38417	52.33	3	4	55.38	1	3	42.63	1	6
s38584	65.13	4	5	67.73	2	3	58.73	2	5
Average	62.60			64.49			54.3		

TABLE II. COMPUTATION TIME FOR TH PROPOSED 2D, 2.5D AND 3D METHODS (SEC.).

Circuit	2D	2.5D	3D
s5378	8.01	7.43	3.87
s9234	15.59	14.20	8.15
s13207	65.41	64.29	25.39
s15850	32.19	29.20	14.69
s38417	107.70	100.72	45.13
s38584	119.38	112.18	50.16
Average	58.05	54.67	24.57

show, 2.5D-PRC can achieve a better compression for six large ISCAS'89 benchmark circuits and 3D is especially good at compressing the industrial-scale circuits.

TABLE IV. COMPRESSION RESULTS FOR THREE INDUSTRIAL-SCALE CIRCUITS

Circuit	2D			2.5D			3D		
	CRs	t	MinLen	CRs	t	MinLen	CRs	t	MinLen
3M99	97.55	6	1	97.52	7	1	97.84	4	1
5M99	97.48	7	1	97.44	7	1	97.78	4	1

REFERENCES

[1] S. Mitra and K. S. Kim, "X-Compact: an efficient response compaction technique," *IEEE Trans. CAD*, vol. 23, pp. 421-432, Mar 2004.

[2] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, "Embedded deterministic test," *IEEE Trans. CAD*, vol. 23, pp. 776-792, May 2004.

[3] S. Mitra and K. S. Kim, "XMAX: X-tolerant architecture for maximal test compression," in Proc. *IEEE Int Conf. Computer Design*, 2003, pp. 326-330.

[4] B. Koenemann, C. Banhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, "A SmartBIST variant with guaranteed encoding," in Proc. *Asia Test Symposium*, 2001, pp. 325-330.

[5] N.A. Touba, "Survey of Test Vector Compression Techniques," *IEEE Des. Test Comput*, vol. 23, no. 4, pp. 294-303, April 2006.

[6] D. A. Huffman, "A Method for the construction of minimum redundancy codes," in Proc. *IRE*, vol. 40, 1952, pp. 1098-1101.

[7] A. Jas, J. Ghosh-Dastidar, Ng Mom-Eng, and N.A. Touba, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 797-806, June 2003.

[8] X. Kavousianos, E. Kalligeros, and D. Nikolos, "Optimal Selective Huffman Coding for Test-Data Compression," *IEEE Trans. Computers*, vol. 56, no. 8, pp. 1146-1152, August 2007.

[9] P. T. Gonciari, B. Al-Hashimi, and N. Nicolici, "Improving compression ratio, area overhead, and test application time for system-on-a-chip test data compression/decompression," in Proc. *Design Automation Test* in Europe, Paris, France, March 2002, pp. 604–611.

[10] M. Tehranipoor, M. Nourani, and K. Chakrabarty, "Nine-Coded Compression Technique for Testing Embedded Cores in SoCs," *IEEE Trans. VLSI Systems*, vol.13, no.6, pp.719-731, June 2005.

[11] A. Chandra and K. Chakrabarty, "System-on-a-chip data compression and decompression architecture based on Golomb codes," *IEEE Trans. Comput. Aided Des.*, vol. 20, no.3, pp. 355–368, 2001.

[12] A. Chandra and K. Chakrabarty, "Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes," *IEEE Trans. Computers*, vol. 52, no. 8, pp. 1076–1088, Aug. 2003.

[13] A. Chandra and K. Chakrabarty, "A unified approach to reduce SoC test data volume, scan power and testing time," *IEEE Trans. Computer-Aided Design Integrated Circuits Syst.*, vol. 22, no. 3, pp. 352-363, Mar 2003.

[14] X. Ruan and R. Katti, "An efficient data-independent technique for compressing test vectors in systems-on-a-chip," *ISVLSI*, 2006, pp. 153-158.

[15] A.H. El-Maleh and R.H. Al-Abaji, "Extended frequency-directed run length code with improved application to system-on-a-chip test data compression," in Proc. *9th IEEE Int. Conf. Electron., Circuits Syst.*, Sept. 2002, pp. 449–452.

[16] A.H. El-Maleh, "Efficient test compression technique based on block merging", *IET Comput. Digit. Tech.*, 2, (5), pp. 327–335, 2008.

[17] M. Nourani and M. Tehranipour, "RL-Huffman encoding for test compression and power reduction in scan application," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 10, no. 1, pp. 91–115, 2005.

TABLE III. RESULT COMPARISONS WITH OTHER PREVIOUS WORKS

Circuit	GOLOMB[11]	FDR[12]	ALT-FDR[13]	EFDR[15]	SHC[7]	VIHC[9]	RL-HC[17]	9C[10]	BM[16]	MD-PRC
s5378	37.11	47.98	50.77	53.67	55.10	51.52	53.75	51.64	54.98	54.63
s9234	45.25	43.61	44.96	48.66	54.20	54.84	47.59	50.91	51.19	53.20
s13207	79.74	81.3	80.23	82.49	77.00	83.21	82.51	82.31	84.89	86.01
s15850	62.82	66.21	65.83	68.66	66.00	60.68	67.34	66.38	69.49	69.99
s38417	28.37	43.37	60.55	62.02	59.00	54.51	64.17	60.63	59.39	55.38
s38584	57.17	60.93	61.13	64.28	64.10	56.97	62.40	65.53	66.86	67.73
Average	51.74	57.23	60.58	63.30	62.57	60.29	62.96	62.90	64.47	64.49