International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-3, Issue-5, September-2015

Enhanced 2^n PRL Code for Efficient Test Data Compression

E. Jebamalar Leavline, K. Thaneesh Kumar, C. Vimal Raj, B. Surya Raj

***Abstract*— Test data compression is needed to minimize the chip area used for storing the test data. The time taken for decompressing the test data is of major concern in the recent past. In this paper, 2^n-pattern run length coding for test data compression is addressed. The 2^n-PRL compression method iteratively encodes 2|n| runs of compatible or inversely compatible patterns either inside a single segment or across multiple segments in to a codeword. It is used to save the memory requirement in automated test equipment (ATE) and test application time (TAT). Also, an enhancement to this 2^n-PRL is proposed and discussed. Theoretical calculation shows that the proposed method further reduces the code word length and hence increases the compression ratio.**

***Index Terms*— Test data compression, Pattern run length coding, exception handling, Automated test equipment.**

I. INTRODUCTION

Data compression involves encoding or compressing the data by reducing the redundant data from the original data. Because of this reduction in the size of the data, the memory size to be used for storing the data also decreases and the transmission speed for the encoded data can be increased. The two well known types of compression available are namely lossless compression and lossy compression. Lossless compression is concerned with encoding information without any loss. In contrary, lossy compression is used to encode the data with some allowable loss of information.

Due to increased transistor count and introduction of new fault models relevant for such technologies Test complexity of integrated circuits (ICs) has increased exponentially in scaled MOS technologies. Large volume of test data are usually generated using Automatic test pattern generators (ATPGs), which is finally stored and applied through an external tester. The external testers are much expensive in terms of test time and storage; hence the low cost test methods are of greater importance. Several test data compression methods have been proposed that provide comparable fault coverage to that of conventional ATPG and reduce the amount of test data and test time [1]- [7].

Manuscript received September 25, 2015.

E.Jebamalar Leavline, Bharathidasan Institute of Technology, Anna University, Tiruchirappalli – India
(e-mail: jebi.lee@gmail.com).
K.Thaneesh Kumar, Bharathidasan Institute of Technology, Anna University, Tiruchirappalli – India
(e-mail: thanesh1kumar@gmail.com).
C. Vimal Raj, B.Surya Raj , Bharathidasan Institute of Technology, Anna University, Tiruchirappalli – India

Test data compression has attracted special attention in the recent past due to the advancements in Integrated circuit technology. The encoded test data must be stored in the equipment called automated test equipment (ATE) during circuit integration process [8]. Un-coded set of data would require larger memory requirement, and hence compressing test data is essential. During testing, the compressed test data will be decompressed and then sent to circuit under test (CUT) for testing [8,9]. In test data compression, lossless compression is used since the loss in test data will alter the fault coverage thereby affecting the performance of fault detection. There are several advantages of test data compression including increased compression ratio, reduced on-chip memory and reduced test application time. There are different run length methods available to compress the test data such as GOLOMB [10], frequency-directed run-length code (FDR) [11, 12], pattern run- length (PRL) [13], and extended frequency-directed run-length code (EFDR) [14-16]. Another most recent approach for test data compression is 2^n pattern run length coding. This approach decomposes the test data into fixed length segments and encodes the data using the run length patterns of 2^n. This method achieved better compression ratio than the state of the art methods. In this paper, an enhancement of this 2^n pattern run length coding is proposed which further improves the compression ratio.

This paper is organized as follows. Section II describes about the various run length coding methods such as frequency-directed run-length code (FDR) [11, 12], and extended frequency-directed run-length code (EFDR) [14]. The 2^n pattern run length coding and the proposed enhancement is presented in Section III and the paper is concluded in Section IV.

II. PROCEDURE FOR PAPER SUBMISSION

This section discusses about the frequency-directed run-length code (FDR), and extended frequency-directed run-length code (EFDR).

A. *Frequency Direct Run (FDR) length code*

FDR code is a coding technique which maps runs of 0's to codeword. Each code consists of a tail and a prefix [12]. For example, the data 0001 000001 011 is coded as 1001 110000 0100.

Data = 0001 000001 011

Codeword=1001 110000 0100

Data is separated into segments as shown above. Each segment ends with symbol '1' and may have many zeroes.

The sample encoding table for FDR code is given in Table – I with the corresponding group prefix and code words for

Group	Run length	Group prefix	Tail	Codeword runs of 0's	Codeword runs of 1's
A1	1	0	0	000	100
	2		1	001	101
A2	3	10	00	01000	11000
	4		01	01001	11001
	5		10	01010	11010
	6		11	01011	11011
A3	7	110	000	0110000	1110000
	8		001	0110001	1110001
	9		010	0110010	1110010
	10		011	0110011	1110011
	11		100	0110100	1110100
	12		101	0110101	1110101
	13		110	0110110	1110110
	14		111	0110111	1110111

runs of '0's and '1's.

Table 1: code words for runs of '0's and '1's using Frequency Direct Run (FDR) length code

According to Chandra et.al FDR coding has the following properties.

The FDR code has the following properties [11]:

- For any codeword, the prefix and tail are of equal length. For example, the prefix and the tail are each one bit long for A_1, two bits long for A_2, etc.
- The length of the prefix for group A_i equals i. For example, the prefix is 2 bits long for group A_2.
- For any codeword, the prefix is identical to the binary representation of the run-length corresponding to the first element of the group. For example, run-length 8 is mapped to group A_3, and the first element of this group is run-length 6. Hence the prefix of the codeword for run-length 8 is 110.
- The codeword size increases by two bits (one bit for the prefix and one bit for the tail) as we move from group A_i to group A_{i+1} .

B. *Extended Frequency Directed Run (EFDR) length code*

It is the advanced method of FDR. EFDR is a coding technique which maps both the runs of 0's and runs of 1's. It also consists of tail and a prefix. If the bit is 0, this indicates that the code word is encoding a run of type 0, otherwise it encodes a run of type 1[14-16]. An example is shown below.

Example:

Data = 01 11110 00001 1110 01 111111111111110

Code word = 000-11001-01001-11000-000-1110111

There are 6 segments in the above original data. Each segment ends with either '0' or '1'. If a segment starts with '0' then that segment will have one '1' at the end. If a segment starts with '1', then that will have one '0' at the end. Similar to FDR code, the length of code words increases by two bits (one for the prefix and one for the tail) when moving from group A_i to group A_{i+1}. FDR code suffers whenever we have runs of 1's, as each 1 bit will be encoded by a separate 0 run of length 0. But EFDR handles this situation by encoding both runs of 0's and 1's. EFDR outperforms FDR interms of compression ratio [15].

III. 2^N PATTERN RUN LENGTH CODING

2^n-PRL iteratively encodes $2^{|n|}$ runs of compatible or inversely compatible patterns either inside a single segment or across multiple segments into a codeword [17,18].

A. *Test data compression with 2n-PRL*

The original data to encode is first divided into equal segments. The segment length may be either 8 bit or 16 bit or 32 bit. The first segment is further divided into equal sub segments. The number of sub segments should be in the order of 2^n. (If n=2, then no of sub segments in 4).The sub segments are generated in such a way that 1st sub segment is either compatible or inversely compatible with other sub segments. This process of dividing a segment into 2^n sub segments is called as internal process. Now if the consecutive segments (2, 3, 4) are compatible or inversely compatible with the first one, then a single codeword is used to represent these consecutive segments. This process is called as external process.

Fig. 1 shows the encoding process of 2^n-PRL coding [18]. The encoding table for a 3-bit exponent PRL and a simple encoding example for segment length L=8 are shown in Table- II and Table – III respectively. Table IV details the code words for different run lengths using 2^n-PRL.

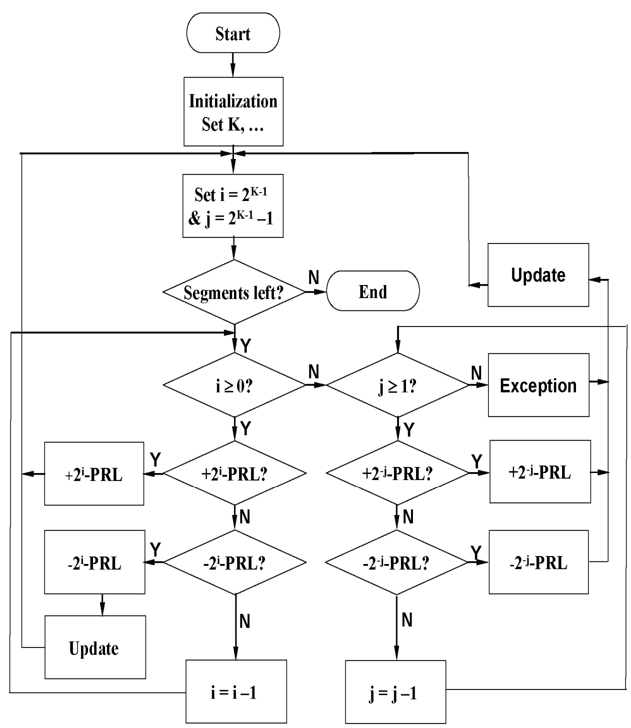

Fig. 1: Encoding process of 2n-PRL coding

The encoding process updates the reference segment at three events. (i) when a segment is encoded by internal 2^n-PRL, the underlying segment becomes the reference segment. (ii) when several segments, inversely compatible with the reference segment, are encoded by external 2^n-PRL, the inversely compatible segment becomes the reference segment. (iii) when a segment is encoded by an exception type, the underlying segment becomes the reference segment [18].

Table 2: Encoding Table for a 3-Bit Exponent

Ext./ Int.	Type	S	E	Type	S	E	Size (Bits)
Ext. 2^n -PRL	$+2^4$	0	100	-2^4	1	100	4
	$+2^3$	0	011	-2^3	1	011	4
	$+2^2$	0	010	-2^2	1	010	4
	$+2^1$	0	001	-2^1	1	001	4
	$+2^0$	0	000	-2^0	1	000	4
Int. 2^n -PRL	1	0	111	-2^1	1	111	4+L/2
	2	0	110	-2^2	1	110	4+L/4
	$+2^{-3}$	0	101	-2^{-3}	1	101	4+L/8

Table 3: Simple Encoding Example for L = 8

Segments		Sref	Code words	Types
S1	11X11XX1	11111111	01011	$+2^{-3}$ -PRL
S2	11XXXXX1	11111111	0001	$+2^1$ -PRL
S3	X1XXXXX1			
S4	0XXXXX0X	00000000	1001	-2^1 -PRL
S5	X0XXXXXX			
S6	X01XXXXX	10101010	011010	$+2^{-2}$ -PRL
S7	X10XXXX1	01010101	1010	-2^2 -PRL
S8	0XXXXXXX			
S9	XXXXXXXXX			
S10	X1XXXXX1			
S11	X0X010XX	10101010	1001	-2^1 -PRL
S12	1XXX1XXX			
S13	100XXXX1	10010101	111010	-2^{-2} -PRL
S14	1011XX10	10111110	010010111110	$+2^4$ -PRL

Table 4: Code words for different run lengths using 2n-PRL.

Group	Run length	Group 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

From the above example it is clearly viewed that number of original bits is 112.These original data is divided into segments (S_1, S_2,… ,S_4). Each segment length is of 8 bits. Now S_1 is subjected to internal process. S_1: 11111111, the resultant codeword for internal process consist of 3 blocks

1bit	K bit	$L/2^n$ bits
Sign	Exponent	Pattern

S_1 is subdivided into 8 sub segments each segment consist only one bit '1' .Hence the first sub segment '1' is compatible with the other sub segments (Don't cares are taken as '1' here). Since S_1 is compatible, the sign bit is assigned as zero ('0'). Number of sub segments for S_1 is subjected under internal process .Hence exponent value is 2^{-3} is '101'.The pattern bit is the first sub segment bit '1'.Hence the code word for first segment is as shown below.

S	E	Pattern
0	101	1

S_2 & S_3 are compatible to S_1(Don't cares are taken as 1's). Hence S_2 & S_3 are subjected to external process .The code word for external process consists of two blocks.

Sig n	Exponent
0	001

Since S_1 & S_3 are compatible to S_1, sign is set as '0'. Two segments (S_2 & S_3) are compatible to S_1. So the exponent value is '2'.Binary representation of '2' is 001. Hence the code word for the segments S_1 & S_3 are '0001'S_4 and S_5 are inversely compatible to the previous segment. Hence the code word is 1001 [18].

Segment S_6 is neither compatible nor inversely compatible to the previous segment. So S_6 has subjected to internal process. So we can define the code words for S_1 to S_{13} by the internal & external process. The segment S_{14} cannot be compressed by both internal and external process. In this case, any of the least frequently used types such as $+2^4$ can be assigned as an exception to encode the non compressible segment. Hence the code word for S_{14} constitutes [0100+S_{14} segments] "0100" represents the binary value of $+2^4$.The total test data volume for this example is reduced from 112 bits to 45 bits with the compression ratio of 59.82% [18].

B. Modified 2^n-PRL coding

The codeword for S_{14} has 12 bits. The length of the codeword is greater than its original data length. To overcome this disadvantage, an enhancement to the 2^n-PRL coding discussed in the previous section is proposed. In this modified 2^n-PRL coding, the original data segment (8 bits) is send directly without any modification if that falls under exception case. Now the code word for S_{14} requires only 8 bits instead of 12 bits in the original 2^n-PRL coding method. All the other segments (S_1 to S_{13}) have their codeword length less than 8 bits. During decoding process, it is easy to identify the S_{14} segment because all other segments have codeword length less than 8 bits except S_{14}. The compression ratio obtained by this proposed method for the above example is 63.39%. Table – V shows the comparison of codeword length and compression ratio achieved through theoretical calculation for EFDR, 2^n Pattern run length coding and the proposed modified 2^n Pattern run length coding for the original test data size of 112bits.

Table 5 :Comparison codeword length and compression ratio

Method	Codeword length	Compression ratio
EFDR	61	45.53%
2^n Pattern run length coding	45	59.82%
Proposed method	41	63.39%

IV. Conclusion

In this paper, an enhancement to the 2^n-PRL coding is proposed. Theoretical calculations show that, the modified 2^n-PRL coding is superior to the 2^n-PRL coding in terms of the compression ratio. This will be very useful for test data compression for automatic circuit test in integrated circuits. Further, the proposed modified 2^n-PRL coding can be implemented on FPGA for its functional verification and the same can be extended to the automatic circuit test.

References

[1] B. Koenemann, “LFSR-coded test patterns for scan designs,” in Proc. Eur. Test Conf., 1991, pp. 237–242.

[2] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, “Embedded deterministic test,” IEEE Trans. Comput.-Aided Des. Integr. Circuit Syst., vol. 23, no. 5, pp. 776–792, May 2004.

[3] K. J. Balakrishnan and N. A. Touba, “Improving linear test data compression,” IEEE Trans. Very Large Scale Integr. Syst., vol. 14, no. 11, pp. 1227–1237, Nov. 2006.

[4] C. V. Krishna, A. Jas, and N. A. Touba, “Reducig test data volume using LFSR reseeding with seed compression,” in Proc. IEEE Int. Test Conf., Oct. 2002, pp. 321–330.

[5] Z. Wang, H. Fang, and K. Chakrabarty, “Deviation-based LFSR reseeding for test-data compression,” IEEE Trans. Comput.-Aided Des. Integr. Circuit Syst., vol. 28, no. 2, pp. 259–271, Feb. 2009.

[6] I. Bayraktaroglu and A. Orailoglu, “Test volume and application time reduction through scan chain concealment,” in Proc. Des. Autom. Conf., 2001, pp. 151–155.

[7] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, A. Ferko,

[8] B. Keller, D. Scott, B. Koenemann, and T. Onodera, “Extending OPMISR beyond 10X scan test efficiency,” IEEE Des. Test Comput., vol. 19, no. 5, pp. 65–72, Oct. 2002.

[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. Des. Automat. Test Eur., Mar.2002, pp. 604–611.

[10] A. Chandra and K. Chakrabarty, “System-on-a-chip data compression and decompression architecture based on Golomb codes,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 20, no. 3, pp. 355–368,Mar. 2001.

[11] Chandra, Anshuman, and Krishnendu Chakrabarty. "Frequency-directed run-length (FDR) codes with application to system-on-a-chip test data compression." In VLSI Test Symposium, 19th IEEE Proceedings on. VTS 2001, pp. 42-47. IEEE, 2001.

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

[13] L. J. Lee, W. D. Tseng, R. B. Lin, and C. L. Lee, “A multi-dimensional pattern run-length method for test data compression,” in Proc. Asian Test Symp., 2009, pp. 111–116.

[14] A. H. El-Maleh, “Test data compression for system-on-a-chip using extended frequency-directed run-length (EFDR) code,” IET Comput. Digit. Tech., vol. 2, no. 3, pp. 155–163, May 2008.

[15] El-Maleh, Aiman H., and Raslan H. Al-Abaji. "Extended frequency-directed run-length code with improved application to system-on-a-chip test data compression." In Electronics, Circuits and Systems, 2002. 9th International Conference on, vol. 2, pp. 449-452. IEEE, 2002.

[16] El-Maleh, Aiman H. "Test data compression for system-on-a-chip using extended frequency-directed run-length code." Computers & Digital Techniques, IET 2, no. 3 (2008): 155-163.

[17] Tseng, Wang-Dauh, and Lung-Jen Lee. "Test data compression using multi-dimensional pattern run-length codes." Journal of Electronic Testing 26, no. 3 (2010): 393-400.

[18] Lung-Jen Lee, Wang-Dauh Tseng, Rung-Bin Lin, Member, IEEE, and Cheng-Ho Chang, “2^n Pattern Run-Length for Test Data Compression” IEEE Transactions on Computer-Aided Design of Integrated Circuits And Systems, Vol. 31, No. 4, April 2012