VOL. 11, NO. 9, MAY 2016 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

AN EFFICIENT TEST DATA COMPRESSION BASED ON ITERATIVE XOR MATRIX COMPUTATION

K. R. Krishnapriya and M. A. Muthiah
Department of Electronics and Communication Engineering, Sathyabama University, Tamil Nadu, India
E-Mail: krkpriya19@gmail.com

ABSTRACT

The continuous increase in complexity of system on chip (SOC) design has resulted in higher test data volume. In this paper, we have proposed a new test data compression technique using an iterative XOR Matrix. This compression is a lossless compression technique that reduces the amount of test data and therefore reduction in test time. Experimental results on ISCAS 89 benchmark circuits are obtained. This demonstrates the effectiveness of the proposed technique in obtaining high compression ratio.

Keywords: XOR matrix, system on chip (SOC), compression.

1. INTRODUCTION

The continuous and rapid advancements in VLSI technology results in increasing complexity of system on chip (SOC) design. The increasing integration density has led to higher test data volume. The time required to test and the cost of testing increases with the size of test data. Test data compression encodes the information in fewer bits than the original data. It involves in addition of on-chip hardware before and after the scan chains [1]. This approach reduces the storage and the testing time. This approach is easier to adopt as it is compatible with the design rules. Researchers have developed many techniques that have improved the design's testability through DFT changes and improved test generation [2].

The test data compression techniques are categorized as scan-based techniques, code-based and linear decompression. Scan-based techniques are based on broadcasting same values to the multiple scan chains. It includes Broadcast scan for independent scan chains, Illinois scan for dependent scan chains. In Illinois Scan Architecture (ILS), the output of scan chains were compressed to multiple input signature analyzer [8]. In code-based technique the data compression codes are used to encode test cubes. The original data is partitioned into symbols and then they are replaced with code word to form the compressed data [4]. The Dictionary based methods are classified into two: Static dictionary and dynamic dictionary. LZ77 is a well-known compression algorithm that is based on Dynamic dictionary [9]. Multiple dictionaries are used in [16] to reduce the number of bits to indicate the dictionary index. In [6], Run length based codes such as FDR and AVR are compared. In Linear decompression, linear operations are used. The linear decompression can be of two types: Combinational continuous-flow linear decompressors and Sequential continuous-flow linear decompressors [7]. LFSR reseeding decompression is used in [20] is used to eliminate the pattern lockout in linear decompression. In [19], linear compression was performed using Viterbi algorithm instead of solving linear equations. All these data compression techniques are lossless compression techniques. Application of lossless compression on test data reduces storage and test time. Most of the methods utilize Golomb code, Huffman codes, Run length coding, Arithmetic codes, and Lempel-Ziv algorithms which are lossless data compression [3]. Arithmetic coding is widely used as an efficient encoding in image compression [14]. The test vector is divided into blocks and block matching algorithm is used in [12] to find and re arrange the blocks of test vector. In [13], block merging compression technique is used. In this method, consecutive compatible blocks are merged.

Huffman coding is a most widely used compression method which is a statistical encoding method. Huffman code is generated by a binary tree named Huffman Tree [10]. The Huffman coding is modified to Selective Huffman [11], Complementary Huffman [17] and Multilevel Huffman [18].

In this paper, we propose an effective and a novel method for test data compression based on Iterative XOR Matrix. The main idea behind this method is to perform effective data compression to the requirement of the tester using the functionality of XOR gate.

2. PROPOSED COMPRESSION TECHNIQUE

The proposed test data compression technique is based on using XOR gates in an iterative fashion at consecutive levels.

The basic functionalities and characteristics of XOR gate is the base of this design. Let us consider two binary input data A and B. Let A XOR B = C, we can find C if A and B are known. If anyone input is unknown and the output is available, we can get the other input. This property is exploited in the proposed test data compression.

The input vector is spitted into blocks (either eight bit or 16 bit blocks) based on the design. The number of blocks = N. This is considered as first level. To obtain the second level, the first two blocks of level one are XORed and the output is the first block of level two. Then this block of level two is XORed with block three of level one to get second block of level two. Thus continuing this operation till reaching the last block of level one will generate block two. The number of blocks in level two =

ARPN Journal of Engineering and Applied Sciences

N-1. Continuing the same for consecutive levels, the numbers of blocks are compressed.

In Figure-1, we show a simple example of the iterative XOR Matrix logic. Let B1, B2, B3, B4, B5, B6, B7, and B8 be eight blocks of input each of eight bits. The iterative XOR Matrix is computed for N-N/2 levels to reduce the blocks into half of the original size. The resultant is of four blocks. In this example, number of blocks (N) = 8. Number of levels in which iterations are performed = 4. Thus we get the blocks reduced into 50% of the original size.

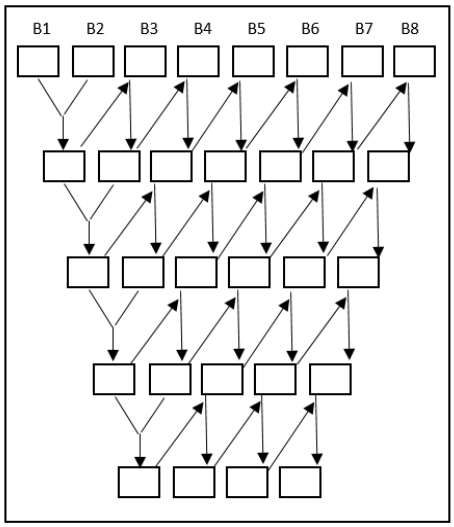

Figure-1. Iterative XOR matrix.

Let us see the encoding and decoding process using this technique in the next section.

3. ENCODING AND DECODING USING THE PROPOSED TECHNIQUE

In this section, we present the design of the Data compression Encoder and its working. The encoder is composed of eight main components. Let us see the test data flow and the functionality of each component.

The following steps provide the encoding process:

Step 1. Don't care Assignment
Step 2. Vector Length Counter
Step 3. Zero padding
Step 4. Generation of Blocks
Step 5. Iterative XOR Matrix Computation
Step 6. Block Merging

Step 1: Don't care assignment

In this step, the don't care bits of the test vector are identified and they are assigned with 0's. Assignment can be done with 1's also, but it should be uniform and make in generalized throughout the design.

Step 2: Vector length counter

In this step, the length of the test vector is determined. It is helpful in generating the data blocks.

Step 3: Zero padding

Based on the vector length obtained in step 2, the zero padding is performed in this step. The encoder is designed for 16 bits block. Therefore if the vector length is not divisible by 16, zeros are added in MSB accordingly to make it divisible by 16.

Step 4: Generation of blocks

In this step, the input test vector is divided into blocks with each 16 bits as length.

Step 5: Iterative XOR matrix computation

In this step, the iterative XOR matrix computation is performed on the input blocks for the required number of levels as per the design. As per the design we aim for 75% data compression.

Let the number of input blocks = K.

Then, the number of output blocks = K – 0.75K

The number of iterations = # input blocks - # output blocks.

Step 6: Block merging

In this step, the generated output blocks are merged together. This is the compressed test data vector obtained by the Iterative XOR Matrix Computation. In Figure-2, the flowchart illustrates the encoding process stepwise.

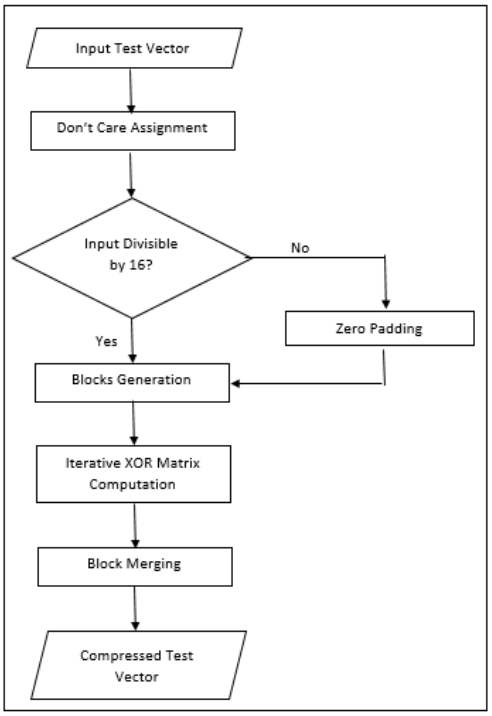

Figure-2. Schematic diagram of the proposed technique.

ARPN Journal of Engineering and Applied Sciences

Let us see the functionality of each component of the Encoder design. The main eight components of the encoder design are: Don't care assigner, Vector length counter, Padding Unit, Padding count storage, Vector Divider, 16-bit stack, Matrix computation Unit, Block Merger.

The Figure-3 shows the Encoder design with its components. The architecture is for 16 bits block design. The same can be implemented for 8 bits block design also. The original input vector contains don't care bits. Compression ratio depend on how the don't care bits are assigned [15]. 0-fill, 1-fill, minimum-transition-fill and random-fill are some of the popular don't-care bit filling techniques [5]. The original input test vector is provided to the don't care assigner. In this design 0-fill is used. Thus the don't care bits are replaced with 0's. This modified data is given to the Vector length counter to obtain the count of the input.

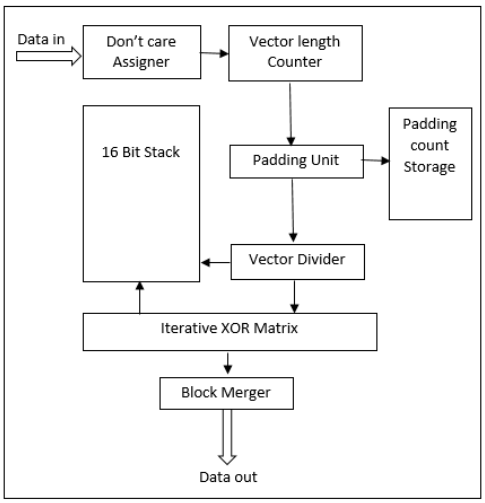

Figure-3. An overview of the proposed method.

If the count is not exactly divisible by 16, it is sent to the Padding Unit. In padding unit, zeros are added to the input such that the count of the vector is rounded to nearest value divisible by 16. The number of zeros padded are stored in the Padding count Unit. This is very important as it plays an important role in decoding. Once the vector length is exactly divisible by 16, the vector is divided into blocks as given in step 4. Therefore let the number of blocks = K. This input blocks are considered as Level 1. The value of first block of level 1 is stored in the 16 bit Stack. Then the XOR Matrix computation is performed on level 1 as given in Section II.

The Iterative XOR Matrix computation is performed till the designed level of compression is obtained. The number of iterations and the levels are determined as given in Step 5. The output blocks obtained are given to the Block Merger. The Blocks are merged and the compressed output vector is obtained in this unit. Thus the output compressed vector is with the compression ratio of 75%.

During decoding, the reverse of the encoding is performed to get the original test vector. The output vector is divided into blocks such that it contains 16 bits each. The previous level is obtained using the XOR operation again. The first block of the previous level is obtained from the 16 bit Stack. On performing the iterative XOR Matrix in the reverse direction the first level of blocks (K) are obtained. Then the padded zeros can be removed referring the padding count unit. Thus the original vector can be decoded.

4. EXPERIMENTAL RESULTS

Experiments were done on ISCAS89 benchmark circuits to measure the effectiveness of the proposed test data compression technique. Experimental results are presented in Table-1. The results are given for both block size 8 and 16. As the block size increases the number of iterations gets reduced.

Compression ratio is determined by the following formula:

Compression ratio = [(# original bits - # compressed bits)/ # original bits]* 100

Table-1. Experimental results of the proposed technique.

ISCAS 89 Benchmark circuit	Original test vector length (Number of bits)	Block size = 8		Block size = 16	
		Compressed test vector length (Number of bits)	Compression ratio	Compressed test vector length (Number of bits)	Compression ratio
S5378	214	48	77.77	48	78.57
S9234	247	56	77.41	64	75.00
S13207	700	176	75.00	176	75.00
S15850	611	152	75.32	144	76.92

5. CONCLUSIONS

In this paper, an effective technique for achieving higher compression ratio has been presented. According to the experimental results, this technique achieves higher compression than existing methods. This is effective regardless of the input data. It is dependent on the vector count.

Thus the compression ratio obtained is well predictable. As the block size decreases the computations increase. Depending on the requirement, the block size can

be selected. This flexibility in selecting the block size is an important advantage of the method.

REFERENCES

[1] N.A. Touba. 2006. Survey of Test Vector Compression Techniques. IEEE Design and Test of Computers. pp. 294-303.

[2] P.Girard. 2002. Survey of Low-Power Testing of VLSI Circuits. IEEE Design and Test of Computers. pp. 82-92.

[3] S. Zahir, A. El-Maleh and E.Khan. 2001. An Efficient Test Vector Compression Technique Based on Geometric Shapes. IEEE. pp. 1561-1564.

[4] Usha Mehta, K.S.Dasgupta and N.M. Devashrayee. 2009. Survey of Test Data Compression Techniques Emphasizing Code Based Schemes. Proc. of 12th Euromicro Conference on Digital System Design / Architectures, Methods and Tools, IEEE Computer Society. pp. 617-620.

[5] R.Karmakar and S. Chattopadhyay. 2015. Thermal-Aware Test Data Compression Using Dictionary Based Coding. Proc. of 28th International Conference on VLSI Design and 14th International Conference on Embedded Systems, IEEE Computer Society. pp. 53-58.

[6] B.YE and M.LUO. 2010. A New Test Data Compression Method for System-on-a-Chip. pp. 129-133.

[7] K.J. Balakrishnan and N.A. Touba. 2006. Improving Linear Test Data Compression. Proc. of IEEE Transactions on Very large Scale Integration (VLSI) SYSTEMS. 14(11): 1227-1237.

[8] A.R. Pandey and J.H. Patel. 2002. Reconfiguration Technique for Reducing Test Time and Test Data Volume in Illinois Scan Architecture Based Designs. Proc. of the 20th IEEE VLSI Test Symposium (VTS02), IEEE Computer Society.

[9] L. Li and K. Chakrabarty. 2003. Test Data Compression Using Dictionaries with Fixed-Length Indices. Proceedings of the 21st IEEE VLSI Test Symposium (VTS03), IEEE Computer Society.

[10] V. Iyengar, K. Chakrabarty and B.T. Murray. 1998. Huffman Encoding of Test Sets for Sequential Circuits. IEEE Transactions on Instrumentation and Measurement. 47(1): 21-25.

[11] P. Dipu, B. Harshavardhan, E. Venkata Ramesh and M. Aarthy. 2014. A Comparitive Study of Compression Decompression Scheme Using Huffman and Selective Huffman Techniques. Proc. of International Conference on Circuit, Power and Computing Technologies [ICCPCT], pp. 1317-1320, 2014.

[12] S.N. Biswas, S.R. Das and E.M. Petriu. 2014. On System-on-Chip Testing Using Hybrid Test Vector Compression. IEEE Transactions on Instrumentation and Measurement. 63(11):2611-2619.

[13] A. El-Maleh. 2006. An Efficient Test Vector Compression Technique Based on Block Merging. ISCAS 2006. pp. 1447-1450.

[14] A. Masmoudi W. Puech. 2014. An Efficient Adaptive Arithmetic Coding for Block-based Lossless Image Compression using Mixture Models. ICIP. pp. 5646-5650.

[15] L. Jiyehi, F. Jianhua, Z. Lida, X. Wenhua and W. Xinan. 2005. A New Test Data Compression/Decompression Scheme to Reduce SOC Test Time. IEEE. pp. 653-656.

[16] V. Janfaza, P. Behnam, B. Forouzandeh and B. Alizadeh. 2014. A Low-power Enhanced Bitmask-dictionary Scheme for Test Data Compression. Proc. of IEEE Computer Society Annual Symposium on VLSI. pp. 220-225.

[17] S.K Lu, H.M. Chuang, G.Y. Lai, B. T.Lai and Y.C. Huang. 2009. Efficient Test Pattern Compression Techniques Based on Complementary Huffman Coding. IEEE.

[18] X. Kavousianos, E. Kalligeros and D. Nikolos. 2008. Multilevel-Huffman Test-Data Compression for IP Cores with Multiple Scan Chains. Proc. of IEEE Transactions on Very large Scale Integration (VLSI) SYSTEMS. 16(7): 926-931.

[19] D. Lee and K. Roy. 2012. Viterbi-Based Efficient Test Data Compression. Proc. of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 31(4): 610-619.

[20] O. Novak, J. Jenicek and M. Rozkovec. 2014. Test-Data Compression with Low Number of Channels and Short Test Time. pp. 104-109.