Indian Journal of Science and Technology, *Vol 9(30), DOI: 10.17485/ijst/2016/v9i30/85750, August 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Burrows Wheeler Transform based Test Vector Compression for Digital Circuits

Anju Asokan* and J. P. Anita

Amrita School of Engineering, Amrita Vishwa Vidyapeetham University, Coimbatore - 641112, Tamil Nadu, India; anju.asok@yahoo.com, jp_anita@cb.amrita.edu

Abstract

Objectives: VLSI testing plays a very crucial role in the design of a VLSI chip. The advances in technology have led to increasing density of transistors and increased circuit complexity in a chip. With the increasing number of inputs, the memory overheads associated with storing test patterns increases. Thus the test pattern volume needs to be compressed. **Methods/Statistical Analysis:** In the proposed approach, a hybrid test pattern compression technique is used along with different schemes such as Huffman and Run length encoding. These encoding schemes are applied on ISCAS'85 and ISCAS'89 benchmark circuits and the results are compared and analyzed based on their compression ratio**. Findings:** In the proposed approach, an improved compression ratio is obtained when compared to the existing techniques in the literature. **Application:** The memory requirements in Automatic Test Equipment (ATE) to store large test data is reduced.

Keywords: Burrows Wheeler Transform, Block matching, Encoding, Huffman, Run Length

1. Introduction

VLSI testing forms an integral part of VLSI design. This verifies that the product design flow is correct and hence ensures the better quality of these products. Testing of a circuit is made possible by the generation of test patterns that are applied to the circuit under test. The generation of test patterns is necessary for checking the proper functionality of the circuit. For testing a given circuit, input test patterns corresponding to the number of primary inputs is to be applied. But with the increasing number of primary inputs for a given circuit, the number of bits per test pattern also increases.

Also, by taking into account all possible faults, the total number of test patterns also increases. This limits the memory requirement on chip to store these patterns. To overcome the limitations of increased bits per pattern and the increased number of test patterns, it is required to reduce the test data volume.

In the existing approaches, the test patterns are generated and subjected to encoding schemes such as Huffman and Run length. Though these encoding schemes guarantee a reasonably good compression ratio, it is possible to use some additional transforms on the test patterns to get increased compression ratios.

Many techniques have been put forward to obtain test pattern compression. [1]proposed compression technique namely bitmask selection method which uses dictionary selection.

The compression ratio is limited since it depends upon the number of patterns which are perfectly mapped with the dictionary entries. Also, if they do not match the entries, this technique adds additional bits along with the original number of bits. To overcome this limitation, a bitmask method is used along with the dictionary selection though it fails to handle don't cares satisfactorily.

[2]has proposed a scheme using an x-ploitting strategy which is most suitable for test patterns with many don't cares. In this technique, x-filling is done and further encoded using shorter code words. This method has the disadvantage that the decoding is complex.

[3]has proposed a scheme using reduced control codes. The resultant code word which is generated is the sum of control codes and the encoded pattern. The control codes

**Author for correspondence*

are the codes which are generated depending on the number of similar blocks that can be merged. But it suffers a limitation that if none of the blocks can be merged, it adds the control bits to the original test pattern thereby increasing the bit count.

[4]has proposed a scheme wherein depending on the whether the succeeding block is a replica of the present block under consideration, the control vectors are assigned.

In the proposed work, the test patterns are generated and divided into blocks. Using the block matching algorithm, these blocks are separated into high and low frequency test blocks. The high frequency test blocks are transformed to reduce bit transitions using an algorithm called Burrows Wheeler Transform (BWT)[5].

To the result obtained after BWT, the low frequency test blocks are further merged and encoding schemes such as Huffman and Run length are applied to compress the merged data set.

2. Materials and Methods

A hybrid test pattern compression technique is used where compression is done with the aid of encoding schemes such as Huffman and Run length. The test patterns are generated by applying all possible faults to a given circuit. Once the patterns are generated, they are divided into blocks such that each block contains equal number of bits. The block matching algorithm is applied on these blocks[6-9]. To apply this algorithm, blocks with lesser number of bit transitions are selected to form the low frequency block[10]. From the original blocks of test patterns, low frequency test pattern blocks are separated to get high frequency test pattern blocks.

Since the high frequency blocks have many bit transitions, these blocks need to be transformed to reduce these transitions which indirectly helps in compression. This transform is the Burrows Wheeler Transform (BWT). The result obtained after BWT and the low frequency blocks are merged and encoded using Huffman and Run length.

Consider the example of s1196 which is an ISCAS'89 benchmark circuit. It has 32 primary inputs.

Let the following be the patterns which can detect all possible faults in this circuit.

T1: 00100000110101001110010011010100
T2: 10101110110101011101011001001100
T3: 10101000011110100100100110000011
T4: 10110111001000010100101101101010

The patterns are divided into blocks of equal bits as:

T1:00100000 11010100 11100100 11010100
T2:10101110 11010101 11010110 01001100
T3:10101000 01111010 01001001 10000011
T4:10110111 00100001 01001011 01101010

Here the test patterns are divided into blocks such that each block contains 8 bits. The number of bits per block varies depending on the number of inputs. To perform block matching algorithm, assume that T1 is the reference pattern, it is the low frequency block while the remaining blocks are considered to be high frequency.

2.1 Burrows Wheeler Transform

It is a transform that can be directly applied to text data rather than binary patterns. In order to perform test pattern compression it is required to treat each high frequency test pattern block as an alphabet and then sort them.

Steps followed are:

1: The high frequency patterns are each allotted with alphabets.

2: These alphabets are cyclically rotated such that the first alphabet is wrapped to the end of the string and so on to form an NxN matrix where N represents the number of alphabets in the string and stored in the form of a table.

3: Each row in the table is sorted alphabetically.

4: The last column in the table is extracted. This is the transformed string. This string is replaced back with the patterns. The transformed patterns are such that there will be reduced transitions between the adjacent patterns.

5: Along with the transformed pattern, a number which indicates the position of the first alphabet in the original string is also added to enable easier decompression.

Consider an example shown in Table 1.

Let the high frequency test patterns be mapped to alphabets as 'dblgci'

The string is rotated cyclically to form an NxN matrix as shown in Table 1 and stored in the form of a table. Here N=6.

Each row is individually treated as a string. The rotation is performed by wrapping the 1st character toward the end of the string[5].

In the Table 2 , since N=6 the rotation is performed till a 6x6 matrix is obtained i.e. till the last character in

the original string 'dblgci' becomes the 1st character in the table.

Table 1. Cyclic rotation of alphabets in the string

d	b	l	g	c	i
b	l	g	c	i	d
l	g	c	i	d	b
g	c	i	d	b	l
c	i	d	b	l	g
i	d	b	l	g	c

Once this is done each row of the matrix is to be sorted alphabetically as shown in Table 2.

This 6x6 matrix is sorted alphabetically as shown in Table 2.

Table 2. Sorting of alphabets in the string

b	l	g	c	i	d
c	i	d	b	l	g
d	b	l	g	c	i
g	c	i	d	b	l
i	d	b	l	g	c
l	g	c	i	d	b

The last column of the table is extracted and transmitted with the position of the first character in the initial string. So 'dgilcb' along with index 1 is transmitted.

The string is replaced by the original pattern and merged with the low frequency test patterns. The entire test set is encoded using Huffman and Run Length encoding individually.

2.2 Encoding Schemes

The merged high frequency and low frequency test patterns are to be encoded to complete the process of compression.

The two basic encoding schemes in the proposed work are Huffman and Run Length encoding.

Huffman encoding- It is a variable length encoding scheme which generates a variable length code word for each symbol depending on how frequently it occurs. It is developed by constructing a binary tree. All the symbols are arranged in ascending order and stored in a stack and those symbols with the lowest frequencies are considered as leaves and a new node is created which is the sum of the two frequencies. This new node is now placed in the stack depending on its frequency. The same procedure is followed till the stack becomes empty. Now the tree is created and place 0's along the leftmost branch and 1's along the rightmost branch. By back tracing the tree from its root will give the binary code word for each symbol which will be unique.

Run length encoding- Another efficient encoding scheme which gives a high degree of compression. Instead of transmitting the entire pattern, the run along with the repeating character is transmitted rather than the original pattern. Run represents the number of times each character repeats. Figure1 shows the complete block diagram of the proposed method.

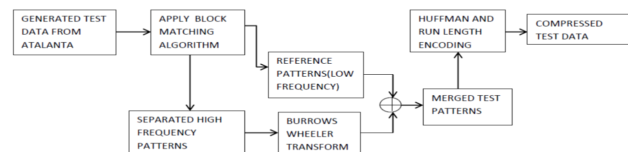

Figure 1. Block diagram of the proposed method.

3. Results and Discussion

To compare the results obtained using Run length encoding and Huffman coding , the compression ratio is calculated as

$$\text{Compression ratio} = \frac{(\text{original number of bits} - \text{number of bits after compression})}{(\text{original number of bits})}$$

The results were obtained by performing the compression techniques on ISCAS'85 and ISCAS'89 benchmark circuits and implemented in the C compiler.

Initially the test patterns were generated by using ALATANTA by injecting all possible faults to the benchmark circuits.

From Table 3, it is evident that the proposed method of using block matching algorithm along with Huffman coding offers an improved compression ratio than that obtained when Huffman is done on the entire data without block matching algorithm. So, on an average, if 1091 bits are considered 76.596 is the percentage compression obtained for the proposed method when compared to 72.21% obtained without block matching giving an increase of 4.386.

The Table 4 gives results for the same procedure followed but using Run length encoding instead of Huffman coding. Due to the larger transitions between adjacent

bits in the pattern, Run length offers reduced compression ratio compared to Huffman encoding.

Table 3. Test pattern Compression results for ISCAS circuits using Huffman encoding

Circuit name	Total no: of bits	Compression % (Huffman)	Compression % (proposed method+ Huffman)	Increase in compression
c432	972	69.032	72.119	3.087
c880	900	64.6666	70.1111	5.4445
c1908	3498	70.6975	82.275	11.5775
c3540	700	86.5714	89.1428	2.5714
s420	1505	63.5219	65.448	1.9261
s444	576	71.3541	72.916	1.5619
s510	1350	68.370	73.4814	5.1114
s1196	480	75.00	80.416	5.416
s1238	448	77.232	80.8035	3.5715
c6288	482	75.7261	79.2531	3.527

Table 4. Test pattern Compression results for ISCAS'85 and ISCAS'89 circuits using Run length encoding

Circuit Name	Total no: of bits	Compression % (Run length)	Compression % (proposed method+ Run length)	Increase in compression
c432	972	16.957	17.829	0.872
c880	900	14.67	16.111	1.441
c1908	3498	7.3184	17.1528	4.8344
c3540	700	12.5714	15.4285	2.8571
s420	1505	21.3953	22.5913	1.196
s444	576	17.0138	21.18055	4.16675
s510	1350	16.888	27.4074	5.5194
s1196	480	15.833	17.9167	2.0837
s1238	448	15.848	17.8571	2.0091
c6288	482	20.5391	22.1991	1.66

From Table 4, it is clear that the proposed scheme offers an increased compression ratio compared to the existing scheme where run length encoding is applied on the entire test set without using block matching and BWT.

So on an average, for 1091 bits, the proposed method gives a compression ratio of 19.56% when compared to 15.91% which is the compression ratio obtained when block matching algorithm is not used.

Table 5 gives the compression ratio comparing the proposed method and that obtained using the approach followed in [5].

Table 5. Comparison of proposed approach to Huffman encoding in [5]

Circuit Name	Total no: of bits	Compression % (Huffman) in [5]	Compression % (proposed method)
c432	972	46.58	72.119
c880	900	63.96	70.111
c1908	3498	51.83	82.275
s420	1505	60.52	65.448
s444	576	32.63	72.916
s510	1350	67.27	73.4814
s1196	480	---	80.416
c3540	700	---	89.1428
s1238	448	----	80.8035
c6288	482	----	79.2531

Figure 2 shows the bar graph representing test pattern compression ratio results using Huffman encoding. From the figure, it is evident that proposed method using block matching gives improved results in terms of compression ratio when compared to that done without block matching algorithm.

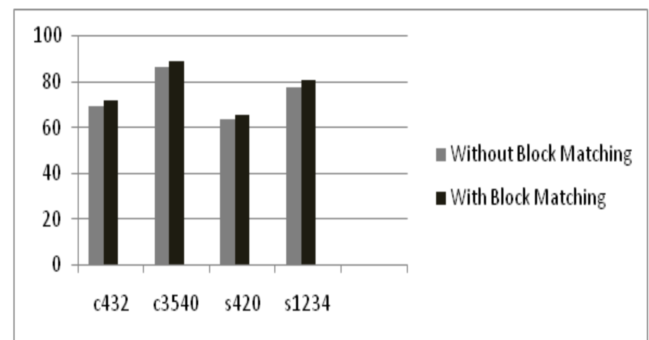

Figure 2. Bar graph representation of test pattern compression ratio using Huffman encoding.

4. Conclusion

To account for the increased memory required to store the test patterns, it is required to develop a scheme to reduce the test data volume. So an efficient method is proposed in the paper which not only reduces the memory overheads but also the testing time. Here using the

block matching algorithm, the high frequency test pattern blocks are separated considering the low frequency blocks as the reference. The high frequency test pattern blocks are transformed based on the Burrows Wheeler Transformation. The low frequency and the transformed high frequency are merged and encoded using Huffman and Run length encoding schemes. The proposed approach is done on ISCAS'85 and ISCAS'89 full scan sequential benchmark circuits. The proposed work can be extended to other encoding schemes also.

5. References

1. Basu K, Mishra P. Test data compression using efficient bitmask and dictionary selection methods. IEEE Trans Very Large Scale Integr (VLSI) Syst. 2010 Sep; 18(9):1277–86.
2. Yi M, Liang H, Zhang L, Zhan W. A novel x-ploiting strategy for improving performance of test data compression. IEEE Trans Very Large Scale Integr (VLSI) Syst. 2010 Feb; 18(2):324–9.
3. Saravanan S, Sai SR, Balasubramaniyan A, Silambamuthan R, Dinesh Babu G, Deepa E. Efficient test data compression achieved by reduced control code. Proceedings of ICMOC, 2012 Jun; 38 :680–84.
4. Tenentes V, Kavousianos X. High-quality statistical test compression with narrow ATE interface. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2013 Sep; 32(9):1369–82.
5. Biswas SN, Das SR, Petriu EM. On system-on-chip testing using hybrid test vector compression. IEEE Trans Instrumen Meas. 2014 Nov; 63(11):2611–19.
6. Saravanan S, Upadhyay HN. Adapting scan based test vector for compression method. Proceedings of International Conference on Communication Technology and System Design. 2011; 30. p. 435–40.
7. Czysz D, Mrugalski G, Mukherjee N, Rajski J, Szczerbicki P, Tyszer J. Deterministic clustering of incompatible test cubes for higher power-aware EDT compression. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2011 Aug; 30(8):1225–38.
8. Reungpeerakul T, Kay D, Mourad S. Partial-Matching technique in a mixed-mode BIST environment. IEEE Trans Instrumen Meas. 2010 Apr; 59(4):970–77.
9. Lee D, Roy K. Viterbi-Based efficient test data compression. IEEE Trans Comput-Aided Des Integr Circuits Syst. 2012 Apr; 31(4):610–19.
10. Maneesh PK, Devi NM. Power based self-referencing scheme for hardware trojan detection and diagnosis. Indian Journal of Science and Technology. 2015 Sep; 8(24):1–5.