Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
2326-9865

A Hybrid Test Data Compression Technique for VLSI Circuits

Anju Asokan [#1],

Department of Computer and Communication Engineering, Sri Eshwar College of Engineering, Coimbatore-641202, Tamil Nadu, India
anjuasokan@sece.ac.in

Article Info
Page Number: 7255-7266
Publication Issue:
Vol. 71 No. 4 (2022)

Article History
Article Received: 25 March 2022
Revised: 30 April 2022
Accepted: 15 June 2022
Publication: 19 August 2022

Abstract
VLSI testing ensures that the product designed is defect free and assures the better quality of the products. Testing of a circuit is made possible by the generation of test patterns. The generation of test patterns is necessary for checking the proper functionality of the circuit. As the number of inputs increases, the memory overhead associated with storing the test patterns increases and also the testing time and test cost also increases. To overcome these limitations, test patterns are compressed. In the proposed method, a hybrid test pattern compression scheme is used which uses Burrows Wheeler Transform(BWT), Move To Front Transform and their combination along with several coding schemes on test data sequences. The entire procedure is carried out on ISCAS' 85 benchmark circuits and the compression ratio for various circuits are computed. The number of transitions between adjacent patterns before and after transform is computed and switching power is also calculated. The area reduced before and after compression is also analyzed.

Keywords: - Burrows Wheeler Transform, Move-to-Front transform, Huffman coding, Frequency directed run length , Shannon Fano.

Introduction

VLSI testing is an integral part of the VLSI design cycle. Testing of a faulty circuit is done by applying the test patterns which propagate the effect of the faults to the output. But as the size of the circuit and the number of faults added to the circuit increases, the number of test patterns and the bits per test pattern increases. So it becomes difficult to store this data. To reduce this test data volume, compression is used. Compression guarantees the reduction in memory overheads which poses a greater problem for larger circuits. Also the test cost can be reduced. Various encoding schemes like Huffman, run-length, Golomb and modification of these techniques are currently used which provide improved compression ratio.

A bitmask and dictionary selection method has been proposed in [1]. Here a known dictionary of patterns is created depending on the generated patterns. Both the techniques are combined in order to obtain more matched patterns. Here the generated patterns are compared with the developed dictionary. If a match is found, encoding is done by representing using three bits. The first bit represents whether the pattern is compressed or not. The second bit indicates whether the bitmask method was used or not and the third bit indicates the index which indicates the position of the pattern in the dictionary. But this technique suffers from the

disadvantage that if a match is not found, the encoding is done such that an additional bit is added along with the pattern thereby increasing the total number of bits to be transmitted.

An alternating variable run-length code is described in [2]. This method generates a variable length code. Here the encoded code comprises of two parts: one is the prefix and the second is the tail part. The prefix is used to find the group where the calculated run lies and the tail part identifies the correct match in the group. When compared to other variations of run-length encoding, this method differs due to the fact that two prefixes were used for each group. The second prefix is an inverted version of the first prefix.

A nine coded compression scheme for data compression has been described in [3]. Here test patterns are considered as fixed sized blocks. The size can be decided depending on number of bits in the pattern. Encoding is then done on individual blocks with exactly nine code words. The block size is considered even, since it becomes easier to divide these blocks into equal groups and encode them. By creating a dictionary of all possible combinations of the blocks of data, the encoding can be done.

An advanced data compression scheme using reduced control codes is seen in [4-6]. This method is suitable mainly for patterns which are not fully specified such that unspecified bits in consecutive test patterns can be merged. Depending on the number of blocks merged, control codes are generated which indicates the number of merged blocks [7-9].

In the proposed method, the generated test patterns are divided into blocks containing equal number of bits. Then using block matching algorithm, the patterns are separated as high and low frequency. The high frequency patterns are then transformed using Burrows Wheeler Transform and the result is merged with the low frequency block. Then the merged blocks are encoded[10][11].

Methodology

ATALANTA tool is used to inject faults into the benchmark circuits and the test patterns that detect these faults are generated. With the increase in complexity of the circuit, the bits in the generated pattern increases [12][13][14]. The generated patterns may contain many transitions within them. The patterns should be such that they should contain even number of bits. These patterns are divided into different groups such that each group contains equal number of bits.

The groups with lesser bit to bit transitions are computed. These are considered the reference groups or low frequency data. Using an algorithm called the block matching algorithm, extracting the reference groups from the entire data set gives the groups with higher transitions or the high frequency groups. These are the groups to be treated further. The high frequency groups are subjected to two different transforms individually. First transform used is the Burrows Wheeler Transform. This transform is applied to lexicographic data and hence the patterns need to be converted to alphabets first before starting the transform. This mapping is done by creating a dictionary which represents this mapping. This transform involves three stages. First is the mapping of patterns to the corresponding alphabets in the dictionary and considering it as a single string. Second is the cyclic rotation of the characters in the string to

form a matrix. The final stage is the sorting of each of the rows in the matrix and extracting the last column in the sorted matrix. This column in transmitted along with the positional index representing the position of the first character in the original string just after mapping. This is required for easier decompression. Since BWT involves many stages, a large memory overhead is utilized to store the intermediate stages. So to replace BWT, Move To Front(MTF) is done. Figure 1 shows the block diagram of the proposed method.

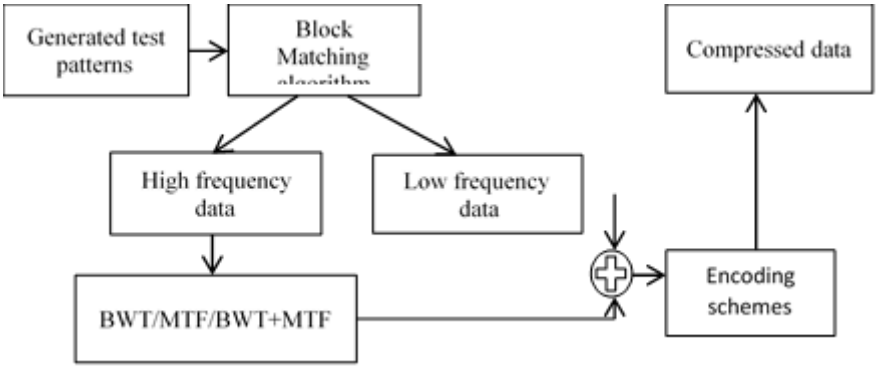

Figure 1. Block diagram of the proposed method.

Move to front Transform

BWT involves three stages and it is difficult to store the outcome of each stage separately which utilizes memory. To overcome this limitation MTF is introduced to replace BWT.

MTF is a single stage process. The main purpose of this method is to move the symbols that frequently occur to front so that they can be represented using a smaller length output.

The main steps in the algorithm are

1. Let S contain a reference list of characters such that each character is present in the string to be encoded exactly once.
2. For every (i =0 to N-1), let P[i] be the number representing position of each element in S. Tabel 1 represents the various stages in Move to Front algorithm

Let input string be “afgbd” and considering a known reference S.

S: a b c d e f g h

P[i]: 0 1 2 3 4 5 6 7

Table 1: Move to front transform stages

Iteration	**Sequence**	**List**
afgbd	0	abcdefgh
a**f**gbd	0,5	abcdefgh
af**g**bd	0,5,6	fabcdegh
afg**b**d	0,5,6,3	gfabcdeh
afgb**d**	[0,5,6,3,5]	bgfacdeh

The final encoded position is transmitted instead of the input string.

Encoding methods

Following the transform stage is the encoding stage wherein actual compression of the data set takes place. Conventionally there are various encoding schemes which are used today. In the proposed work five major encoding schemes are used.

Run length encoding

This is a commonly used encoding scheme where the count of number of ones and zeroes is tracked and encoded. The steps followed in the algorithm are:

1. The first character is taken from the source string.
2. Count the number of times the character occurs in the source string.
3. Check if the character is repeating in the string or not.
4. If true, append the character along with the count to the destination string.
5. If false, write the character to the destination.

Huffman encoding

It is a variable length encoding scheme which is based on a binary tree like structure. The steps followed in the algorithm are:

1. Count the number of times each symbol occurs in the string.
2. Arrange the symbols in the ascending order of their frequencies and extract the two with the least frequencies.
3. Create a new node with frequency as the sum of two nodes and remove them.
4. Continue the process till all the characters are taken.
5. Place zero on the left branch and one on the right branch.

Frequency directed run length encoding

It is an encoding scheme similar to run-length encoding where 1's are tracked and preceding 0's are encoded. The steps followed in the algorithm are:

1. Track the occurrence of 1's in each block.

2. Count the number of 0's preceding the tracked 1's.

3. Represent the count of 0's using minimum number of bits.

4. Check if all the 1's are tracked

5. If true, append the line with number of 0's equal to block size

6. If false, go back to step 2.

Advanced Golomb encoding

This is a scheme based on a known dictionary. By performing mapping based on the encoding Table 2, the shorter length code is developed for each pattern.

Table 2: Encoding table for Advanced Golomb coding

INPUT BLOCK	DESCRIPTION	CODE WORD
00000 00000	All 0's	0
11111 11111	All 1's	10
00000 11111	Left Group 0's, Right Group 1's	11 000
11111 00000	Left Group 1's, Right Group 0's	11 001
11111 XXXXX	Left Group 1's, Right Group Mismatched	11 010
XXXXX 11111	Left Group Mismatched, Right Group 1's	11 011
00000 XXXXX	Left Group 0's, Right Group Mismatched	11 100
XXXXX 00000	Left Group Mismatched, Right Group 0's	11 101
XXXXX XXXXX	All Mismatched	1111

Shannon Fano encoding

This method generates a tree like structure similar to Huffman encoding but the difference in both the structures is the way in which the probability of occurrence is arranged.

The steps followed in the algorithm are:

1. Calculate the probability of occurrence of each symbol.
2. Arrange them in descending order based on their probabilities.
3. Partition the symbols such that they are as close as possible.
4. Repeat the process till all the symbols are used.
5. Place 0's on the left side of the branch and 1's on the right side of the branch.

3. Results and Discussion

The simulations were done in C compiler. Test patterns were generated with the help of ATALANTA tool. In order to compute the switching power, PRIME TIME tool was used and the memory utilization in terms of number of LUTs was synthesized using XILINX ISE.

All the simulations were carried out on ISCAS'85 benchmark circuits. The compression ratio was computed for the individual circuits. Also the switching activity in terms of the number of transitions involved was computed. Compression ratio is calculated by finding the difference

between the total number of bits before and after compression and dividing it by the total number of bits before compression and expressing it as percentage.

Table 3 shows the compression ratio obtained after performing BWT,MTF and combination of these transforms followed by run-length encoding on ISCAS'85 circuits. The number represented in brackets shows the total number of bits reduced upon compression. Consider c432 circuit which has 36 inputs. The test set totally consists of 972 bits. After BWT, the number of bits gets reduced to 798. After MTF, the bits get reduced to 453 and after performing BWT, MTF followed by run-length, the number of bits become 407.

Table 3: Run-length encoding for ISCAS'85 circuits

Ckt Name	Total no: of bits	% Compression on applying run-length alone	% Compression on applying BWT and run-length	% compression on applying MTF and run-length	% compression on applying BWT, MTF and run-length
c432	972	16.9570(809)	17.8290(798)	53.3950(453)	58.1275(407)
c880	900	14.7610(768)	26.6370(661)	51.0543(441)	58.3795(375)
c1908	3498	7.3184(3242)	17.1528(2898)	71.5555(995)	71.8124(986)
c3540	700	12.5712(612)	15.4285(592)	78.0000(154)	80.4285(137)
c6288	482	20.5394(383)	22.1991(375)	60.9958(188)	61.6182(185)

Table 4 shows the compression ratio obtained after performing BWT,MTF and combination of these transforms followed by Huffman encoding on ISCAS'85 circuits. For c432 circuit, the test set totally consists of 972 bits. After BWT, the number of bits gets reduced to 271. After MTF, the bits get reduced to 269 and after performing BWT, MTF followed by Huffman, the number of bits become 236.

Table 4: Run-length encoding for ISCAS'85 circuits

Ckt Name	Total no: of bits	% Compression on applying Huffman alone	% Compression on applying [1]	% Compression on applying BWT and Huffman	% compression on applying MTF and Huffman	% compression on applying BWT,MTF and Huffman

c432	972	69.0320(301)	46.58	72.1190(271)	72.3251(269)	75.7201(236)
c880	900	64.6660(318)	63.96	70.1110(270)	70.1442(269)	73.8068(236)
c1908	3498	70.6975(1024)	51.83	82.2750(620)	82.3607(617)	82.5042(612)
c3540	700	86.5714(94)	66.07	89.1428(76)	89.2857(75)	89.4285(74)
c6288	482	75.7261(117)	54.67	80.9128(92)	81.1203(91)	81.3278(90)

Table 5 shows the compression ratio obtained after performing BWT,MTF and combination of these transforms followed by FDR encoding on ISCAS'85 circuits. For s1196 circuit, the test set consists of 972 bits. After BWT, the number of bits gets reduced to 787. After MTF, the bits get reduced to 448 and after performing BWT, MTF followed by FDR, the number of bits becomes 368.

Table 5: FDR encoding for ISCAS'85 circuits

Ckt Name	Total no: of bits	%Compression on applying FDR alone	%Compression on applying BWT and FDR	%Compression on applying MTF and FDR	% Compression on applying BWT, MTF and FDR
c432	972	16.0490(816)	19.0470(787)	53.9094(448)	62.1399(368)
c880	900	18.0910(738)	20.1100(719)	50.9433(442)	60.5993(355)
c1908	3498	13.0930(3040)	14.0400(3006)	72.6700(956)	72.8416(950)
c3540	700	14.0000(602)	15.0690(594)	77.7142(156)	80.8571(134)
c6288	482	16.3500(403)	16.5970(402)	61.2033(187)	64.5288(171)

Table 6 shows the compression ratio obtained after performing BWT,MTF and combination of these transforms followed by Advanced Golomb encoding on ISCAS'85 circuits. For c432 circuit, the test set consists of 972 bits. After BWT, the number of bits gets reduced to 387. After MTF, the bits get reduced to 272 and after performing BWT, MTF followed by Advanced Golomb coding, the number of bits becomes 194.

Table 7 shows the compression ratio obtained after performing BWT,MTF and combination of these transforms followed by Shannon Fano encoding on ISCAS'85 circuits.

Table 6: Advanced Golomb encoding for ISCAS'85 circuits

Ckt Name	Total no: of bits	%Compression on applying Advanced Golomb alone	%Compression on applying BWT and Advanced Golomb	%Compression on applying MTF and Advanced Golomb	%Compression on applying BWT,MTF and Advanced Golomb
c432	972	57.7590(411)	60.1744(387)	72.0164(272)	80.0411(194)
c880	900	42.8412(515)	44.2640(501)	69.8113(272)	76.1376(215)
c1908	3498	42.2813(2019)	43.9020(1962)	83.5048(577)	83.5334(576)
c3540	700	52.4285(333)	57.0322(300)	87.2857(89)	90.0000(70)
c6288	482	62.6556(180)	62.6556(180)	75.5186(118)	76.7634(112)

Table 7: Shannon Fano encoding for ISCAS'85 circuits

Ckt Name	Total no: of bits	% Compression on applying BWT and Shannon	%Compression on applying MTF and Shannon	% Compression on applying BWT,MTF and Shannon
c432	972	55.9670(428)	57.0980(417)	79.1152(131)
c880	900	53.7800(416)	57.0470(386)	80.0000(180)
c1908	3498	64.8646(1229)	65.5517(1205)	70.1257(1045)
c3540	700	81.8570(127)	84.0000(112)	84.4220(109)
c6288	482	68.4640(152)	78.8500(126)	75.3110(119)

Switching activity is a measure to analyze the switching power. By evaluating the cumulative transitions the switching activity is found. Table 8 shows the number of transitions before and after performing BWT, MTF and combination of these transforms on ISCAS'85 circuits. The value in brackets represents the % switching activity.

Table 8: Number of transitions for ISCAS'85 circuits

Ckt Name	Switching Activity (original data)	Switching Activity (BWT transformed data)	Switching Activity (MTF transformed data)	Switching Activity (BWT+MTF transformed data)
c432	963	877(8.930%)	355(63.136%)	230(76.116%)
c880	854	831(2.693%)	373(56.323%)	241(71.779%)
c1908	1740	1622(6.781%)	831(52.241%)	737(57.643%)
c3540	630	580(7.936%)	153(75.714%)	106(83.174%)
c6288	405	379(6.419%)	181(55.308%)	154(61.975%)

Switching power is a major contributor to total power. By reducing the switching power, total power can be reduced. Table 9 shows the switching power before and after performing BWT, MTF and combination of these transforms on ISCAS'85, circuits are measured using PRIME TIME tool.

Table 9: Switching power for ISCAS'85 circuits

Ckt Name	Switching Power (initial) (in uw)	Switching Power (after BWT)(in uw)	Switching power(after MTF)(in uw)	Switching Power(after BWT+MTF)(in uw)
c432	36.4032	32.6195	20.5874	12.1546
c880	49.1352	33.1973	17.6685	10.9594
c1908	119.8585	105.7952	91.2327	65.6420
c3540	220.1592	206.6960	180.9599	148.1410
c6288	1430.4	1378.4	1102.0	923.4504

After the encoding schemes are done, the total number of bits gets reduced. As a result, the resources utilized in terms of the number of LUTs also get reduced. Table 10 shows the resource utilization before and after performing BWT,MTF and combination of these transforms followed by Huffman encoding on ISCAS'85

Table 10: Resources utilization(Number of LUTs) for ISCAS'85 circuits using Huffman

Ckt Name	Initial	After applying BWT	After applying MTF	After applying BWT +MTF
c432	63	58	28	20
c880	73	64	50	33

c1908	69	62	38	17
c3540	242	206	191	166
c6288	660	638	590	534

Table 11 shows the resource utilization before and after performing BWT,MTF and combination of these transforms followed by Frequency Directed Run-length encoding on ISCAS'85 circuits.

Table 11: Resources utilization(Number of LUTs) for ISCAS'85 circuits using FDR

Ckt Name	Initial	After applying BWT	After applying MTF	After applying BWT +MTF
c432	63	57	56	50
c880	73	69	64	63
c1908	69	62	61	58
c3540	242	238	230	224
c6288	660	655	651	639

Table 12 shows the resource utilization before and after performing BWT,MTF and combination of these transforms followed by Advanced Golomb encoding on ISCAS'85 circuits.

Table 12: Resources utilization for ISCAS'85 circuits using Advanced Golomb coding

Ckt Name	Initial	After applying BWT	After applying MTF	After applying BWT +MTF
c432	63	61	57	54
c880	73	72	69	68
c1908	69	67	66	64
c3540	242	233	230	229
c6288	660	658	654	650

Table 13 shows the resource utilization before and after performing BWT,MTF and combination of these transforms followed by Shannon Fano encoding on ISCAS'85 circuits.

Table 13: Resources utilization for ISCAS'85 circuits using Shannon Fano

Ckt Name	Initial	After applying BWT	After applying MTF	After applying BWT +MTF
c432	63	61	56	49
c880	73	72	71	69
c1908	69	68	47	38
c3540	242	239	237	229
c6288	660	659	655	651

4. Conclusion

Compression of test data helps to reduce the memory overhead incurred due to the increased complexity of the circuit. After getting the compressed data, the switching activity between the patterns also get reduced and as a result the switching power is also reduced. The results obtained after MTF guarantees an optimized design in terms of both compression ratio and reduced switching power.

Block matching algorithm was used to separate high and low frequency data and then BWT,MTF and combination of both the transforms were individually applied on the high frequency data and encoding done. The number of transitions was computed and switching power was calculated.

References

[1] K. Basu, P. Mishra, P, "A novel test-data compression technique using application-aware bitmask and dictionary selection method", Proceedings of the 18th ACM Great Lakes symposium on VLSI, pp.83-88.

[2] B. Ye, Q.Zhao ,D.Zhao , Z.Wang, M. Luo, " Test data compression using alternating variable length code", Integration, **44(2)**, 103-110, 2011.

[3] M. Habibi M , A. Bafandeh, M.A Montazerolghaem, "A digital array based bit serial processor for arbitrary window size kernel convolution in vision sensors", Integration-The VLSI Journal, 47(4), pp.417-430 ,2014.

[4] S. Saravanana and V.Sai , "Efficient Test Data Compression Achieved by Reduced Control Code", International Conference on Modeling Optimization and Computing, pp. 680-684 ,2012.

[5] S. Asvini, and C. Nirmala, "Design for testability in timely testing VLSI circuits", International Journal of engineering Research and Applications ,5(3), pp.10-13 ,2015.

[6] K. Basu , P. Krishnamurthy, F. Khorrami and R. Karri, "A theoretical study of hardware performance counters-based malware detection", IEEE Transactions on Information Forensics and Security, 31, 2020.

[7]A. Alagarsamy, S. Mahilmaran, L.Gopalakrishnan and S.B. Ko, "FRDS: An efficient unique on-chip interconnection network architecture",Integration, 87, pp.90-103, 2022.

[8] F. Silva, R. Oliveira, A.L. Zimpeck ,C. Meinhardt, R.Reis, "Exploring XOR-based full adders and decoupling cells to variability mitigation at FinFET technology", Integration, 87, pp.137-146 ,2022.

[9] J.Wei, T.Zhao, Z.Zhang, J.Wan, "Modeling of CMOS transistors from 0.18 um process by artificial neural network", Integration, 87, pp.11-15 ,2022.

[10] A. Razzaq, S.R.Sani, A.G. Ye, "The effect of gate voltage boosting on the power efficiency of multi-context FPGAs", Integration, 86, pp. 30-43, 2022.

[11] L.Raja ,P.S. Periasamy, "A trusted distributed routing scheme for wireless sensor networks using Block Chain and Jelly fish search optimizer based deep generative adversarial neural network (Deep-GANN) technique", Wireless Personal Communications ,31, pp.1-25, 2022.

[12]M.Z. Hussain, M. Ashraf, D.K. Singh, A. Haldorai, D.K. Mishra, T.N. Shanavas, "Intelligent data post and read data system like to feed for IoT sensors", International Journal of System Assurance Engineering and Management, 33, pp.1-22, 2022.

[13]S.Ayub, R. Boddu, H. Verma, S.Revathi , B.K. Saraswat and A.Haldorai, "Health index estimation of wind power plant using Neurofuzzy modelling", Computational and Mathematical Methods in Medicine, 2022, pp.1-8

[14]C.K.Raja, A.Kannimuthu, "Conditional generative adversarial network approach for autism prediction", Sci.Eng, 44,pp.741-755,2023.