Indonesian Journal of Electrical Engineering and Computer Science
Vol. 12, No. 3, December 2018, pp. 933~940
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v12.i3.pp933-940

A SoC-IP Core Test Data Compression Scheme Based on Error Correcting Hamming Codes

Sanjoy Mitra[1], Debaprasad Das[2]
[1]Department of Computer Science and Engineering, Tripura Institute of Technology, Agartala, India
[2]Department of Electronics and Communication Engineering, TSSOT, Assam University, Silchar, India

Article Info

Article history:

Received Feb 26, 2018
Revised May 21, 2018
Accepted Aug 19, 2018

Keywords:

Compression
Hamming code
Huffman coding
Non-valid slice
RLE
Test data
Valid slice

ABSTRACT

As system-on-chip (SoC) integration is growing very rapidly, increased circuit densities in SoC have lead a radical increase in test data volume and reduction of this large test data volume is one of the biggest challenges in the testing industry. This paper presents an efficient test independent compression scheme primarily based on the error correcting Hamming codes. The scheme operates on the pre-computed test data without the need of structural information of the circuit under test and thus it is applicable for IP cores in SoC. Test vectors are equally sliced into the size of '*n'* bits. Individual slices are treated as a Hamming codeword consisting of '*p'* parity bits and'd' data bits ($n = d + p$) and validity of each codeword is verified. If a valid slice is encountered '*d'* data bits prefixed by '*1'* are written to the compressed file, while for a non-valid slice all '*n'* bits preceded by '*0'* are written to the compressed file. Finally, we apply Huffman coding and RLE in order to improve the compression ratio further The efficiency of the proposed hybrid scheme is verified with the experimental outcomes and comparisons to existing compression methods suitable for testing of IP cores.

Corresponding Author:

Sanjoy Mitra,
Department of Computer Science and Engineering,
Tripura Institute of Technology,
Narsingarh Agartala, Tripura (W),
India.
Email: mail.smitra@gmail.com

1. INTRODUCTION

The prime objective of test data compression is to lessen the volume of binary bits in original ATPG generated test cube. is stored in the An automatic test equipment's (ATE) internal memory is used to store the compressed test vectors and an on-chip decompression hardware is applied to decompress this ATE stored data which is subsequently applied to circuit under test (CUT)[1], shown in Figure 1. The decompression hardware is specifically designed for any distinct compression approach and is suitable for all the original test set which using this compression approach.

1.1. Background

The test data compression methods may ordinarily be grouped in three types: code-based schemes, linear-decompression-based schemes and broadcast- scan-based schemes [2]. Code-based schemes mostly target the given test sets, in which original test data are broken into different symbols and each symbol is substituted by a code word to form the compressed test data. Here, prior knowledge of the internal structure information of the circuit under test (CUT) is not needed; besides, the fault simulation and test generation are not required. Thus, these schemes are especially handy for test data compression with SoC IP core circuits. These coding methods can be categorized into two different classes based on the differences of symbol

division. In 'fixed' category, fixed numbers of input bits are encoded by the underlying compression mechanism. Analogously in 'variable' category; variable numbers of input bits are encoded by the compression algorithm. Huffman coding is the idle instance of such "fixed" scheme. The Huffman encoding algorithm encodes frequently occurring symbols with shorter code words and on the other hand, least frequent ones are assigned relatively longer code. Other instances of such 'fixed' category are selective Huffman coding (SHC) [3] and optimal selective Huffman coding (OSHC) [4], dictionary-based coding [5] and block merging coding [6] and so on. Single run-length and double run-length encoding method fall in the category of 'variable' scheme. Runs of '0' s are encoded in case of single run-length coding techniques and examples of this include Golomb code [7], frequency-directed run-length (FDR) code [8] and variable input Huffman code (VIHC) [9]. In case of double run-length code compression techniques, both runs of 0 s and runs of 1 s are encoded. Extended FDR code (EFDR) [10] alternating run-length coding (AFDR) [11] and mixed double run-length and Huffman coding (RL-HC) [12] are the examples of double run length encoding

1.2. Problem

Faster development of Integrated Circuit fabrication process forcing an inevitable increase in the density of circuit components in a chip and this has raised test data volume a lot, which further not only enlarges the testing time but also surpasses the tester memory capacity [13]. System Integrators faces some difficulties while testing Intellectual property (IP) cores as the structure of the IPs is unknown to them and this complexity of IP cores and their size is the prime cause of larger test data volumes and obviously, longer test application time (TAT) is needed for quality post-production test. A large volume of test data is needed to be stored in the automatic test equipment (ATE) and transmitted deep into the chip as quickly as possible. Limited the size of memory and constant channel capacity of ATE triggers significant rise in the test application time and the test power. Test data compression techniques have the potential to resolute the problem of higher test data volume during SoC-IP testing.

1.3. Solution

In this paper, error correcting Hamming codes are applied for test data compression. Although Hamming codes [14, 15] are mostly applied for error correction, it can also be applied in test data compression allowing a small amount of distortion [16–19]. In this paper, we introduce a hybrid compression scheme which is primarily based on the error correcting Hamming code. We claim on the basis of experimental outcomes that the scheme efficiently compresses the SoC-IP Core test data.

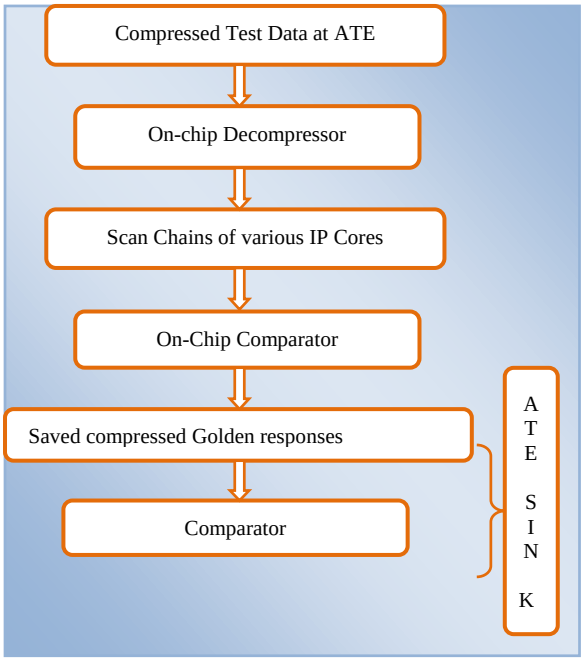

Figure 1. SoC Test Model

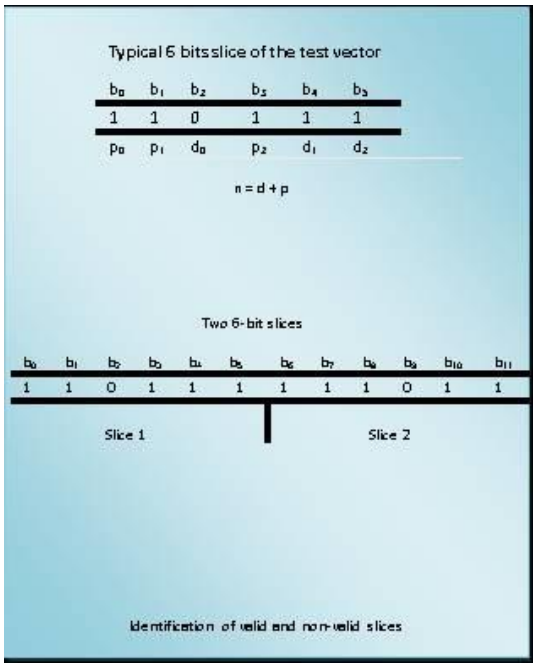

Figure 2. typical slicing of Test Data

2. THE HAMMING CODE BASED DATA COMPRESSION ALGORITHM

The error-correcting code incepted by Hamming finds more extensive application in advanced information processing and communication systems. It is well equipped for distinguishing two bits error and corrects a single bit error. Furthermore, burst errors [14, 15] can likewise be corrected with the aid of Hamming codes. Let us assume a message with d data bits and it is to be coded using Hamming codes. The prime idea of the Hamming codes lies in the utilization of additional parity bits (p) keeping in mind the end goal to recognize a single bit and an identification of two bits errors. Here, the 'n' bits of coded message is generally constituted by the relation: n = d + p. Individual parity bit drives for the parity of several groups of data bits, including itself, to be odd (or even), where every parity is calculated on different subsets of the data bits. The bits of the codeword are numbered successively, beginning with bit 1 at the left end, bit 2 to its immediate right, and so on. In Hamming codes, the parity bits and data bits are positioned at a particular place in the codeword. The parity bits occupy positions 2^0, 2^1, 2^2. . . 2^{p-1} in the sequence which has at most $2^{p-1}-1$ positions. The leftover positions are preserved for the data bits, see Figure 2. For a codeword of n bits, there are 2^n possible code words having values from 0 to $2^n - 1$, the only 2^d of them are valid code words and $2^n - 2^d$ are non-valid code words.

3. PREPROCESSING OF TEST DATA

Test data obtained from ATPG is subjected to preprocessing steps like don't care bit filling and splitting of test data into suitable slices.

3.1. Slicing of Input Test Data

We divide the input test data into scan chains of predetermined length. Let us assume that the test data T_D consists of n test patterns. We divide the scan elements into m scan chains in the best-balanced manner possible. This result in each vector being divided into m sub-vectors, each of length, say l. Dissimilarity in the lengths of the sub-vectors are resolved by padding don't cares at the end of the shorter sub-vectors. Thus, all the sub-vectors are of equal length. The m-bit data which is present at the same position of each sub-vector constitute an m-bit slice. If there are vectors at the beginning, we obtain a total of $n \times l$ m-bit slices, which is our uncompressed data set that needs to be compressed.

3.2. Don't' Care Bit Filling

The test cube generated by automatic test pattern generator (ATPG) tool contains a great quantity of don't care(X) bits. Such don't care bits in test cube can be manipulated for enhancing the test data compression. In statistical coding techniques, test data is split into equal size slices of m bits. Test data compression may be improved by reducing the number of distinct slices in a given test set and also by increasing the frequency of occurrence for each distinct slice. In this hybrid compression scheme, we apply an existing don't care bit filling algorithm namely MT -fill which has less computational complexity compared to other algorithms. We have chosen Minimum Transition Fill (MT-fill) over other techniques owing to the fact that it reduces the number of weighted transitions in the test vector, thereby reducing the test power.

In MT-fill, a progression of X entries in the test vector is filled with an indistinguishable value as the first non-X entry on the right side of this arrangement. This limits the quantity of transitions in the test vector when it is scanned in.

For example, consider the test vector: 100XX010X1X0. This vector, after MT-fill, would become 100000101100. If the test vector has a string of X bits that is not terminated by a non-X bit on the right side, then it should be filled by the bit value to the left of the sequence. For example 1000001011XX should be 100000101111 after MT-fill.

4. THE PROPOSED METHODOLOGY

We propose an implementation of a hybrid compression scheme for reducing the volume of test data. Our proposed scheme is primarily based on error correcting Hamming codes. The Hamming code introduces additional bits, known as parity bits, whose function is to validate the exactness of the original message sent upon receipt. This method transforms the slice of size m bits into n by adding up p parity bits, based on the size of the message m, which is encoded into a codeword of length n. Figure 4 shows the block diagram of the technique

4.1. Sequence of Compression Steps

Our proposed approach takes the ATPG generated original test vector and implements a no. of steps, as follows:

Step 1: Stacking of the test data file for encoding.
Step 2: Conversion of each m-bit slices into an n-bit length where (m>n) by applying the Hamming decoders on the input test vector. It is worth mentioning here that every slice of size m bits is not the code word; some non-code word m-bit slices may also exist. So, an extra bit is put in use to differentiate between non-code words and code words. We append a bit '1' if the bit slice is a codeword and bit '0' if it is not a codeword to an additional file called an added bit file. The test data slice that is a valid codeword is converted to an n-bit slice instead of an m-bit slice. Invalid codeword is shifted as it is, without any compression, into the compressed binary file.
Step 3: Further compression of test vector file by applying Huffman coding.
Step 4: Run Length Encoding (RLE) and the Huffman encoding algorithm are applied to added bit file. Afterward the file is added to the header of the compressed binary file. As a result, we attain a pleasing result with noteworthy improvement in compression ratio. Figure 3 signifies the flow of the proposed method.

4.2. Decompression Mechanism

A decompression process is carried out in order to reinstate the original test data from the compressed data file and is performed as follows:
Step 1: Read the header of the compressed file and take out the added bit file from that, after that, decode the extracted file by applying the Huffman decoder. Apply the RLE decoder in order to reinstate the added bit file to its original form.
Step 2: Decoding of the compressed data file by means of the Huffman algorithm.
Step 3*:* Encoding of the Huffman decoded test data file by applying the Hamming error correction algorithm. In this phase, every bit of this file is accessed and thus the length of every slice is dependent on the value in the *added bit* file. If the bit of *added bit* file is 1, then the slice size is *n* bits, otherwise, the size of the slice is *m* bits. The Hamming encoder returns the slice of size *n* bits to its original size of *m* bits and returns the parity bits that were removed during Hamming decoding.
Step 4*:* Lastly, the test data file is returned to its original status without any loss of data

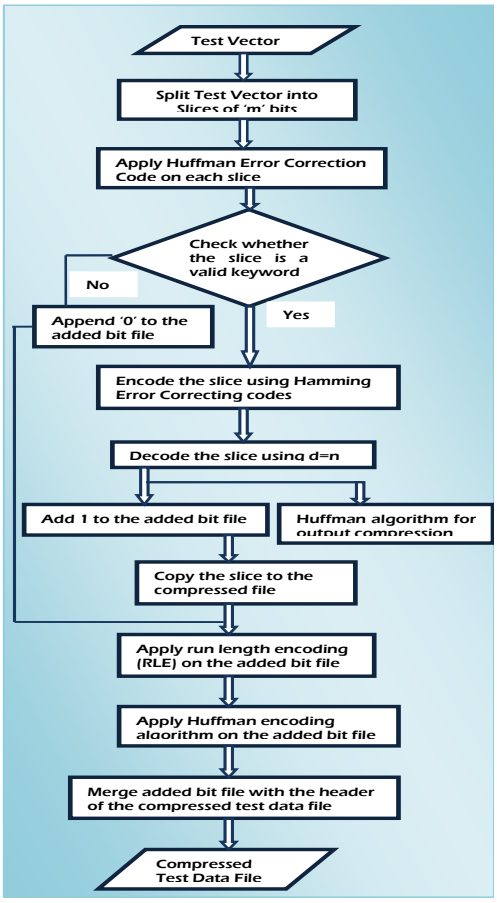

Figure 3. Flow of the compression scheme

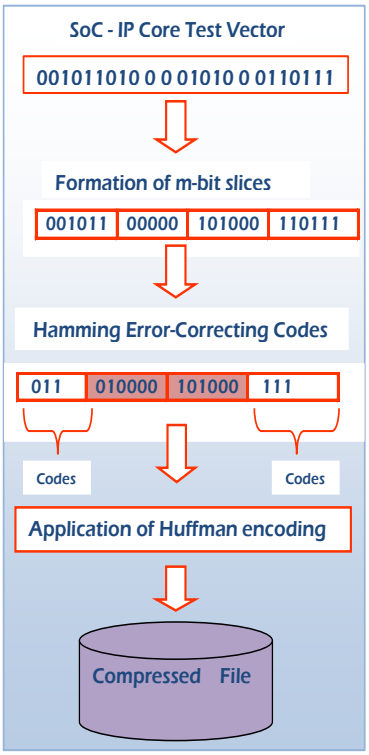

Figure 4. Scheme illustration using the block diagram

5. EXPERIMENTAL RESULTS AND ANALYSIS

In order to observe the likely improvements to be delivered by the proposed method, experiments are carried out on the seven ISCAS'89 [20] benchmark circuits. Synopsys Tetra MAX [21] ATPG tool is used to generate the test data. Synopsys Tetra MAX was functional with the dynamic compaction turned on and random-fill turned off. The proposed encoding scheme based on the Hamming code is analyzed from the several points of view of the test data compression and their effects on compression: the compression ratio, size of slice and number of parity bits etc.. The above issues are very important in the context of the proposed scheme. The compression ratio, $C_R(T_D)$, that is estimated with the following

$$C_R(T_D) = \left(\frac{|T_D| - |T_E|}{|T_D|} \right) \times 100 \quad (1)$$

Where $|T_D|$ denotes the size of the original test data and the size of the compressed test data is represented by $|T_E|$

Table 1. Some Details of Test Data in Single Scan Chain Architecture

ISCAS 89 circuit[20]	Total no of patterns	Bits per pattern	Original Test data size(T_D) in bits
s5378	111	214	23,754
s9234	159	247	39,273
s13207	236	700	1,65,200
s15850	126	611	76,986
s35932	16	1763	28,208
s38417	99	1664	1,64,736
s38584	136	1464	1,99,104

Table 2. Compression Ratio based Comparison with the Previously Proposed Techniques

ISCAS 89 circuit[20]	Selective Huffman[4]	Golomb [7]	FDR[8]	EFDR[10]	ALT-FDR[11]	RL-Huffman[12]	9C[22]	Our Proposed
s5378	42.32	37.11	48.02	53.67	45.39	46.17	51.56	**64.60**
s9234	38.14	45.25	43.59	48.66	35.32	42.0	50.91	**61.27**
s13207	66.95	79.74	69.59	**82.19**	29.11	69.51	72.31	63.65
s15850	52.61	62.82	56.82	67.82	25.90	57.83	66.37	**72.35**
s35932	50.71	43.21	44.07	39.41	34.30	55.08	57.45	**58.30**
s38417	79.87	28.37	**85.17**	62.03	22.41	89.44	60.41	68.76
s38584	57.80	57.17	60.84	61.12	23.60	61.52	65.53	**72.60**

Table 3. Evaluation of Compression Ratio based on Different Slice Sizes and Variation in the Number of Parity Bits

	Slice Size(*S*)														
ISCAS 89	*S=7*			*S=10*			*S=15*			*S=20*			*S=25*		
	No of parity bits(p)			No of parity bits(p)			No of parity bits(p)			No of parity bits(p)			No of parity bits(p)		
circuit	*p=3*	*p=4*	*p=5*	*p=3*	*p=4*	*p=5*	*p=3*	*p=4*	*p=5*	*p=3*	*p=4*	*p=5*	*p=3*	*p=4*	*p=5*
s5378	57.67	57.70	57.73	57.76	58.80	58.83	57.81	59.70	59.71	57.82	59.72	61.90	57.91	59.78	64.60
s9234	54.30	54.36	54.46	54.33	55.30	55.41	54.50	56.30	56.32	54.53	56.45	58.36	54.67	56.52	61.27
s13207	55.91	55.94	55.95	55.97	57.97	57.99	55.98	58.61	58.64	55.99	59.85	61.49	56.05	59.87	63.65
s15850	62.37	62.42	62.45	61.40	63.21	63.22	61.42	65.71	65.78	61.50	65.90	68.21	61.50	65.94	72.35
s35932	46.72	46.81	46.93	46.76	49.03	49.10	46.77	52.88	52.97	46.80	52.91	55.73	46.80	53.72	58.30
s38417	60.10	60.15	60.24	60.14	61.40	61.44	60.32	63.90	63.97	60.49	64.12	66.38	60.40	64.21	68.76
s38584	63.62	63.72	63.80	63.73	65.87	65.90	63.78	67.87	68.00	63.79	68.30	70.90	63.80	69.20	72.60

In order to verify the efficiency of the proposed method, implementation of the proposed compression method was carried out in the 'C' language on a Linux system. Test cubes were generated for the seven largest ISCAS-89[16] fully scanned test bench circuits. Tetra Max [14] ATPG tool was used to generate these test data cubes. Table 1 gives the description of the scan chain network scheme. In Table 1, we provide the details like number of patterns, bits per pattern and total bits of test data cube for the traditional single scan chain architecture etc. The the number of test patterns provided in column 2, may be further reduced by applying compaction. If the compacted test vector of the scan chain is exactly equal to other compacted test vectors, the test vector can be rejected from the test cube.

Table 2 presents a comparison of the compression ratio with selective Huffman coding [4], Golomb coding [7], FDR coding [8], EFDR coding [10], ALT-FDR Coding [11],], RL-Huffman coding [12], and 9C [22]. For each test bench circuit, we have used five different test data slice sizes (i.e. S=7, S=10, S=15, S=20, S=25) and three different lengths of parity bits (i.e. p=3, p=4, p=5). Column 9 gives the best compression ratio of the proposed compression schemes based on the number of parity bits of error correcting Hamming code and size of the test data slice. This hybrid compression proposal outperforms most of the previously published compression schemes tabulated in Table 2 for the majority of the test bench circuits. The proposed scheme compresses the test data thrice in three consecutive distinct compression methods: firstly with Hamming code based compression, then by RLE and finally with the aid of Huffman coding and thus yields impressive test data compression ratio in most of the benchmark circuits used in the experiment. In Table 2, it can be observed that this test data compression scheme has resulted in significant improvement of the compression ratio in all the listed benchmark circuits except the benchmark s13207 and s38417.

In Table 3, it can be seen that variation in the parameters like slice size(S) and no of parity bits during the compression with Hamming error correcting codes is influencing the overall compression ratio of this hybrid compression scheme. Here in Table 3, it is evident that in most of the cases, with the increase in test data slice size and no of parity bits, the compression ratio is also improved in contrast to the previous one. If we critically observe the compression ratio improvement pattern in Table 3, we can notice that significant improvement is recorded in case of { (S=7,p=3) to (S=10,p=4)}, {(S=10,p=4) to (S=15,p=4)},{(S=15,p=4) to (S=20,p=5)} and (S=20,p=5) to (S=25,p=5)} for most of the benchmark circuits

Table 4. Variation of Compression Ratio with the Percentage of the Valid Slice, Size of Slice and Quantity of Parity Bits

ISCAS 89 circuit	Slice Size*(S)*, No of parity bits*(p)*, Percentage of the valid slices *(V_S)*									
	S=7, p=3		*S=10, p=4*		*S=15,p=4*		*S=20,p=5*		*S=25,p=5*	
	% of valid slice*(V_S)*	*CR*	*%* of valid slice*(V_S)*	*CR*	*%* of valid slice*(V_S)*	*CR*	*%* of valid slice*(V_S)*	*CR*	*%* of valid slice*(V_S)*	*CR*
s5378	25	57.67	37	58.80	39	59.70	37	61.90	35	64.60
s9234	28	54.30	40	55.30	41	56.30	40	58.36	32	61.27
s13207	23	55.91	38	57.97	31	58.61	33	61.49	34	63.65
s15850	34	62.37	41	63.21	37	65.71	34	68.21	30	72.35
s35932	27	46.72	43	49.03	45	52.88	39	55.73	37	58.30
s38417	32	60.10	28	61.40	41	63.90	40	63.38	38	68.76
s38584	34	63.62	30	65.87	35	67.87	34	65.90	31	72.60

It can be clearly understood from Figure 3 and Figure 4 that some of the hamming errors correcting codes are valid and some others are non-valid. In Table 4, we have shown the percentage of valid slices capable of generating valid codes which in turn produces the initial level of test data compression. In Table 4, the percentage of valid slices corresponding to various test data slices are shown. Here the valid slice percentage V_S corresponding the pair (slice size, no of parity bits) against different benchmark test data and their respective compression ratio (CR) is also given. Highest compression ratio, 72.60 is achieved for the pair (S=25, p=5) having 31% of valid slices.

Apart from the tabular representation of the experimental data, we have also put the glimpse of the experimental outcomes in graphical representation with the aid of a column-bar chart. In figure 5, benchmark wise comparison of compression ratio (CR) against the pair of slice size and no of parity bits is shown. The pairs: {(S=7, p=3), (S=10, p=4), (S=15, p=4), (S=20, p=5) and (S=25, p=5)} and their corresponding benchmark circuit wise compression ratios are plotted in Figure 5 for graphical staging of the comparison

Comparison of different compression methodologies with our proposed method on the basis of the compression ratio in different benchmark circuits is plotted in figure 6. This proposed method was compared with other existing methods namely selective Huffman coding [4], Golomb coding [7], FDR coding [8], EFDR coding [10], ALT-FDR Coding [11], RL-Huffman coding [12], and 9C [22]and almost in majority of the benchmark circuits, the bar corresponding to our proposed compression scheme is standing highest with highest compression ratio among all other compression methods.

The behavior of different benchmark circuits in terms of compression ratio(CR) corresponding to the pairs of slice size and number of parity bits is graphically depicted in Figure 7. Here, the compression ratio corresponding to the pairs: {(S=7, p=3), (S=10, p=4), (S=15, p=4), (S=20, p=5) and (S=25, p=5)} against the benchmark circuits used in this experiment are plotted in Figure 7. From this graphical presentation, we may correlate how the compression ratio is varying across different benchmark circuits

depending on the combination of the size of the slice and number of parity bits used in the Hamming error correcting codes.

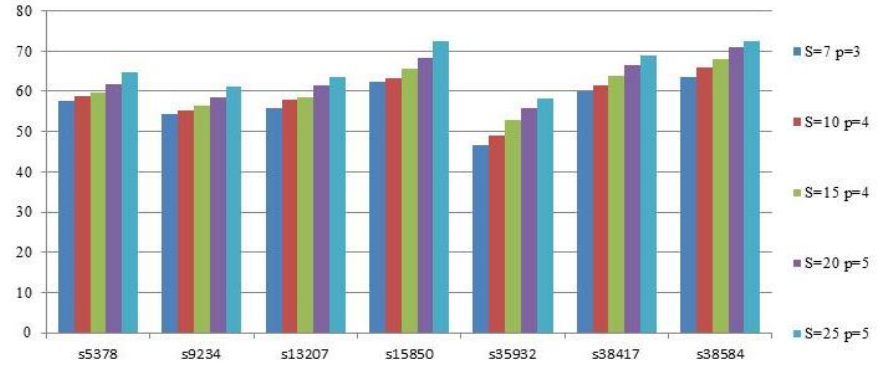

Figure 5. Benchmark wise comparison of CR based on the combination of slice size(S) and number of parity bits (p)

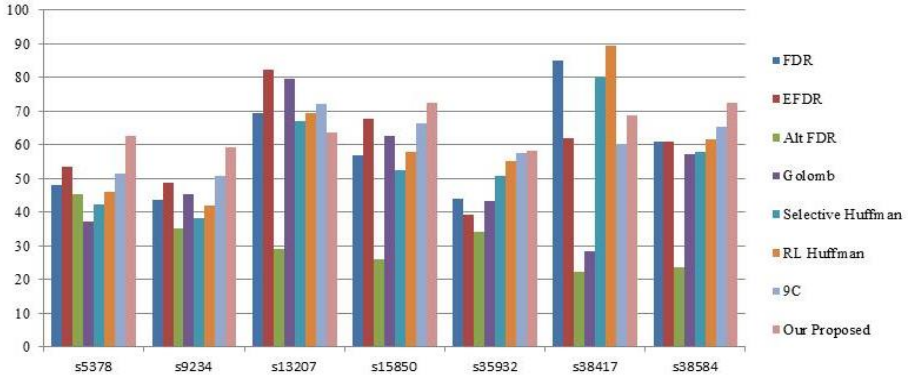

Figure 6. CR based comparison of different compression methods with the proposed method on different benchmark circuits

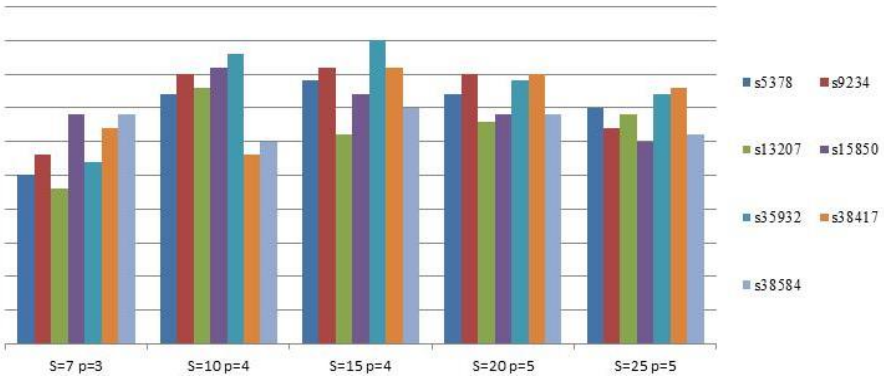

Figure 7. CR based behavior analysis of different benchmark circuits depending on the combination of slice size(S) and number of parity bits (p)

6. CONCLUSION

This paper presented a test compression algorithm that combines the advantages of the Hamming error-correcting codes, RLE and Huffman encoding. This paper developed the efficient utilization of Hamming error-correcting codes in combination with RLE and Huffman encoding algorithm for test data compression in order to improve compression ratio of SoC-IP core test data. We have applied our algorithm on various benchmarks and compared our results with existing test compression techniques. Our hybrid

compression scheme outperforms other existing test data compression in a significant manner, giving a best possible compression of 72.60%. Significant improvement in compression efficiency is observed at the cost of probably little increase in on-chip decoder area overhead. Further improvement of compression ratio and on-chip decoder area minimization for such hybrid test data compression schemes may be the future prospects of research in this particular sub-problem. Significant features from the compression approaches [23-25] may also be incorporated in ordr to frame up more efficient hybrid compression mechanism.

REFERENCES

[1] P.T. Gonciari, B.M. Al-Hashimi and N. Nicolici, "*Improving compression ratio, area overhead and test application time for system-on-a-chip test data compression/decompression*," Proc. Design, Automation and Test in Europe Conf., 2002.

[2] Z You, W Wang, Z Dou, P Liu and J Kuang "A scan disabling-based BAST scheme for test cost reduction," *IEICE Electron. Express*, vol. 8 (2011) pp 1367-1373

[3] Jas ,J. Ghosh-Dastidar ,Mom-Eng Ng and N.A. Touba "An efficient test vector compression scheme using selective Huffman coding*,"IEEE Transaction on. Computer-Aided Design Integration Circuits System*s, vol. 22(6) 2003, pp 797-806

[4] X Kavousianos, E Kalligeros and D Nikolos "Optimal selective Huffman coding for test-data compression," *IEEE Transactions on Computers*, vol. 56 (8) July 2007, pp 1146-1152

[5] P. Sismanoglou and D. Nikolos "Input test data compression based on the reuse of parts of dictionary entries: Static and dynamic approaches," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32 (11) October 2013, pp 1762-1775

[6] T. B. Wu, H Z Liu and P X Liu "Efficient test compression technique for SOC based on block merging and eight coding," *Journal of* ElectronicTesting, Vol.29 (6) December 2013, pp 849-859

[7] Chandra and K. Chakrabarty "System-on-a-chip test-data compression and decompression architectures based on Golomb codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*s, vol. 20 (3) March 2001, pp 355-368.

[8] Chandra and K. Chakrabarty "Test data compression and test resource partitioning for system-on-a-chip using frequency directed run-length (FDR) codes," *IEEE Transactions on Computers*, vol.52 (8) 2003, pp 1076-1088

[9] P.T. Gonciari, B.M. Al-Hashimi , and N. Nicolici, "Variable-length input Huffman coding for system-on-a-chip test" IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.22(6), pp 783 – 796, June 2003

[10] H. El-Maleh "Test data compression for system-on-a-chip using extended frequency-directed run-length code," *IET Computers and Digital Techniques*, vol.2 (3) April 2008, pp 155-163

[11] Chandra and K. Chakrabarty "A unified approach to reduce SOC test data volume, scan power and testing time," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22 (3) March 2003, pp 352-363

[12] M. Nourani and M. H. Tehranipour "RL-Huffman encoding for test compression and power reduction in scan application," *ACM Transactions on Design Automation of Electronic Systems*, Vol.10 (1) 2005, pp 91-115

[13] Usha S. Mehta, Kankar S. Dasgupta and Niranjan M. Devashrayee "Run-length-based test data compression techniques: How far from entropy and power bounds?—A survey," *VLSI Design* Feb 2010 pp 1-9

[14] R.W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29(2) April 1950, pp 147–160.

[15] Tanenbaum, Computer Networks, Prentice Hall, 2003.

[16] G. Caire, S. Shamai and S. Verdu "*Lossless data compression with error correction codes,"* Proc. IEEE International Symposium on Information Theory, 2003, pp 22

[17] G. Caire, S. Shamai, S. Verdu, "A *new data compression algorithm for sources with memory based on error correcting codes*," Proc. IEEE Workshop on Information Theory, 2003, pp 291–295

[18] A.A. Sharieh, "An enhancement of Huffman coding for the compression of multimedia files," *Transactions on Engineering, Computing and Technology*, Vol. 3 2004, pp 303–305.

[19] T.C. Bell, I.H. Witten, J. G, *Cleary Text Compression*, Prentice Hall, 1990.

[20] F. Brglez, D. Bryan and K. Kozminski "*Combinational profiles of sequential benchmark circuits*," In IEEE International Symposium on Circuits and Systems, Vol. 3 May 1989, pp 1929-1934

[21] Synopsys Inc.: Tetra MAX ATPG user Guide, 2006.

[22] M Tehranipoor, M Nourani and K Chakrabarty, " Nine-coded compression technique for testing embedded cores in SoCs" . IEEE Transactions on VLSI *Systems*, Vol. 13(6) 2005 , pp 719–731

[23] S J. Sarkar, N K Sarkar, T Dutta, P Dey, and A Mukherjee5, "Arithmetic Coding Based Approach for Power System Parameter Data Compression" , *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 2 (2),pp. 268-274, May 2016

[24] T S Gunawan, M Kartiwi, "Performance Evaluation of Multichannel Audio Compression", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 10(1), pp. 146-153, April 2018

[25] W Song, "Strategies and Techniques for Data Compression in Wireless Sensor Networks" *TELKOMNIKA*, vol.11 (11), pp. 6624-6630, November 2013.