PREP SYSTEM-ON-CHIP TEST DATA COMPRESSION BASED ON SPLIT- DATA VARIABLE LENGTH (SDV) CODE

First J. ROBERT THEIVADAS Second V. RANGANATHAN
Department of ECE Reserach Scholar, Anna University,Chennai, Tamil Nadu, India, roberttheivadas@gmail.com
Department of ECE, Vignan University, Guntur, Andrapradesh, India, India' drvrangan@gmail.com,

Third J. RAJA PAUL PERINBAM
Department of ECE, KCG College of Technology, Chennai, Tamil Nadu, India' rperinbam@yahoo.com.

Abstract: *System-on-a-chips with intellectual property cores need a large volume of data for testing. The large volume of test data requires a large testing time and test data memory. Therefore new techniques are needed to optimize the test data volume, decrease the testing time, and conquer the ATE memory limitation for SOC designs. This paper presents a new compression method of testing for intellectual property core-based system-on-chip. The proposed method is based on new split-data variable length (SDV) codes that are designed using the split-options along with identification bits in a string of test data. This paper analyses the reduction of test data volume, testing time, run time, size of memory required in ATE and improvement of compression ratio. Experimental results for ISCAS 85 and ISCAS 89 Benchmark circuits show that SDV codes outperform other compression methods with the best compression ratio for test data compression. The decompression architecture for SDV codes is also presented for decoding the implementations of compressed bits. The proposed scheme shows that SDV codes are accessible to any of the variations in the input test data stream.*

.

Key words: Test data compression, SDV codes, SOC, ATE, Benchmark circuits

1. Introduction

One of the important objectives of detailed testing of VLSI (very large scale integration) circuits and systems is to make sure that there are no defects in the manufactured products, and that they meet the specifications. Further, the information generated during the testing may come of help in increasing the product yield via improvising the process technology by reducing the production cost.

The fabrication process of the integrated circuit has different steps including photolithography, printing, etching and doping. During manufacturing, each of these steps has its own flaws. These flaws may lead to failure in the operation of the individual ICs. VLSI technology has made the testing of ICs complex and time consuming resulting in the increased cost of the ICs. The problems related to testing have enormously intensified in case of SOC (System on Chip) because of the large numbers of IP (Intellectual Property) cores on a single silicon chip. This huge reduction in the circuit size has increased the sensitivity to performance variations and still require complete testing before they are shipped to the customers. There is no denying of the fact that the overall quality of the final product is greatly improved through testing, though it has no bearing on the manufacturing caliber of the ICs. The testing ensures the imperfections of the product are detected only if it is implemented during the crucial stages of the development cycle. It can also be taken as an approach to validate the design and check the process involved.

The different IP cores of a SOC are not readily accessible because of the complexity of the SOC and inadequate test pins. However, one can increase the accessibility of a controllable or observable node in the circuit by applying the DFT (Design-For-Testability) strategies. These strategies the test cost, increase the product's quality, and makes easy the design characterization and test program implementation. In order to test these systems effectively, each IP core must be exercised duly with the core vendor's set of established test patterns. In case of VLSI systems, due to higher storage requirements for fault-free responses, the traditional test processes become quite expensive. Alternate approaches are pursued in order to reduce the test data volume or the amount of storage required.

A design methodology called BIST (Built-in-Self-Testing) [1] - [3] is capable of providing solutions to numerous problems faced in testing digital systems. In order to test a SOC, the test patterns are initially generated and stored in an exclusive computer. The increase in the different types of SOCs has increased the requirement for the numbers of test patterns and also the frequent downloading of these test patterns into an ATE (Automatic Test Equipment). The size of these test patterns can be quite large that it may require a lot of time for downloading an ATE. However, the data memory, input-output channel capacity, and speed of traditional ATEs are limited. These machines are also quite expensive. Because of this, newly developed compression techniques are expected to decrease testing time and storage capacity.

The emergence of new test vector compression techniques [4], [5] and data compression in SOC techniques (SOC containing an embedded processor core) [6], [7] has been recently reported. Embedded processor has significantly increased Soc design flexibility, lengthened SOC product lifetime, and reduced design errors by permitting the devices to adapt to evolving standards and by allowing the addition of extra features over time [8]. Therefore, the compression technique must not only be efficient enough to decrease the test data volume, but also effective enough in its decompression. Various techniques have been proposed in the literature for compressing test data and their decompression. The Test sets are compressed using the selective Huffman coding is presented in [9] assuming full scan circuits and allowing reduction of hardware overhead of Huffman FSM. The Test data is encoded by using nine-types of codeword, Nine-Coded Compression based on fixed to variable coding scheme. The drawback of the method is the increased length of symbol. Other methods are presented based on Frequency-Directed Runlength (FDR) [10], Golomb Codes, Variable-input Huffman coding (VIHC), Lempel-Ziv-Welch (LZW) Coding [12], Associative Coder of Buynovsky (ACB) [11] , Run Length Coding (RLC), Burrow-Wheeler Transform (BWT) [14],Variable to Variable Huffman code and Tunstall coding. Compression techniques are proposed in order to reduce test data volume. In [10], the test vectors are compressed as difference vectors T_{diff} from Original Test vectors T_D and this leads to the reduction of testing time and smaller test sets.

In this paper, a new technique has been presented for efficient implementation of test data compression and decompression for system-on-a-chip designs. This paper introduces a new class of variable length compression codes that are designed using the split-options along with identification bits of string of test data.

2. Proposed Methodology

2.1. SDV Codes

The Split-data algorithm technique approach is to split-off the data from a string of test data vector in order to achieve compressed bits in the form of variable-length codes. The Simple Run-length code for remaining higher order bits contains a run of '0s' which is equal to the decimal value of the higher order bits. Digit '1' is deployed after a sequence of '0s'.For Example, Simple Run Length code is shown in Table 1 by using some of the code word. The B^{th} split-option splits the code word into LSB and MSB, whereas LSB splits-off based on the value of B. In case, if B split bit is 0 then it results in no split-bit of LSB, Incase B split bit is 1, one bit from LSB splits-off, if B split bit is 2, two bits from LSB splits-off and remaining higher order bits are encoded to a run length codes. In this paper, the B^{th} split-option is extended up to value of B=7,since each of the code word is taken as 8 bits and each of the split-option results in the reduction of the number of bits. The Table 2 shows the Split-options using simple Run-Length Codes (RLC). Test vectors considered in Table 2 are analyzed for each B value by using the concept of split-data. For example, in the code word 00001111, since there is no data split for B, B=0 so only the RLC is performed. This results in some compression. For B=1, 1 LSB splits-off and remaining higher bits are encoded into 7zeroes+1.After concatenation of LSB and RLC, the total number of bits is 9 (1LSB+RLC->1+7zeroes+1). For B=2, 2 LSB splits-off and so RLC is performed as 3zeroes+1 resulting in 6 bits (2LSB+RLC->11+3zeroes+1). In order to know highest reduction of bits from all the B split-option, the overall number of bits in each B split option has to be added. The reduction of the number of bits for each split B option is seen in Table 2. This is made up to B=7. To find highest compressed bits, in each and every split-option total number of compressed bits must be added from all the code words. In Table 2, the total number of compressed bits for B=0 is 518bits, B=1 is 201bits,B=2 is 114 bits, B=3 is 74bits,B=4 is 58 bits and finally B=5 is 55 bits. By comparing all the total number of compressed bits for all the values of B, the highest compression is achieved in B=5 with 55 bits. This result in a higher reduction of bits compared to other values of B. From Table 2, B=5 will achieve the data reduction from the original 64 bits to 55 bits. In this case, B=5 will be selected. The split-option achieving the highest compression is selected to identify the option to the decoder

Table 1.Run Length code

CODE WORD	3-BIT RUN-LENGTH CODE	CODE WORD	4-BIT RUN-LENGTH CODE
000	1	0011	0001
001	01	0100	00001
010	001	1010	00000000001
011	0001	1001	0000000001
.	.	.	.
111	00000001	1111	0000000000000001

Table 2.Split-Options Using Simple Run Length Codes (RLC)

CODE WORD	B=0 (RLC)	B=1 (1LSB+RLC)	B=2 (2LSB+RLC)	B=3 (3LSB+RLC)	B=4 (4LSB+RLC)	B=5 (5LSB+RLC)
00001111	15zeroes+1= 16 bits	1+7zeroes+1=9 bits	11+3zeroes+1= 6 bits	111+1zero+1= 5 bits	1111+1=5bits	01111+1= 6bits
00011111	31zeroes+1= 32bits	1+15zeroes+1= 17bits	11+7zeroes+1= 10bits	111+3zeroes+1 =7bits	1111+1zero+1 =6bits	11111+1= 6bits
01011010	90zeroes+1= 91bits	0+45zeroes+1= 47bits	10+22zeroes+1 =25bits	010+11zeroes+ 1=15bits	1010+5zeroes+ 1=10bits	11010+2z eroes+1=8 bits
00111100	60zeroes+1= 61bits	0+30zeroes+1= 32bits	00+15zeroes+1 =18bits	100+7zeroes+1 =11bits	1100+3zeroes+ 1=8bits	11100+1z ero+1=7bi ts
00001111	15zeroes+1= 16 bits	1+7zeroes+1=9 bits	11+3zeroes+1= 6 bits	111+1zeroes+1 =5 bits	1111+1=5bits	01111+1= 6bits
01100100	100zeroes+1=1 01bits	0+50zeroes+1= 52bits	00+25zeroes+1 =28bits	100+12zeroes+ 1=16bits	0100+6zeroes+ 1=11bits	00100+3z eroes+1=9 bits
00110100	52zeroes+1= 53 bits	0+26zeroes+1= 28bits	00+13zeroes+1 =16bits	100+6zeroes+1 =10bits	0100+3zeroes+ 1=8bits	10100+1z ero+1=7bi ts
00001010	10zeroes+1= 11 bits	0+5zeroes+1=7 bits	10+2zeroes+1= 5bits	010+1zero+1= 5bits	1010+1=5bits	01010+1= 6bits
TOTAL BITS	518	201	114	74	58	55

BLOCK DIAGRAM AND ALGORITHM OF PROPOSED METHOD

Figure 1 shows the block diagram of the proposed technique. The Test vectors required for testing an SOC are compressed in software mode. The Execution of the decompression program allows recovering the uncompressed original test vectors and then these test vectors are applied to each and every core of the SOC to analyze the output responses. The Test vectors provided by core vendor are divided into 8 bits of equal size and the size of the block depends on the total number of bits in each vector. The Block selection is divided into several blocks of equal size based on the total number of bits. In this paper, the number of bits for each block is an 8bit code word and the total number of blocks presented here is 8 blocks in parallel i.e. B=0 to B=7.

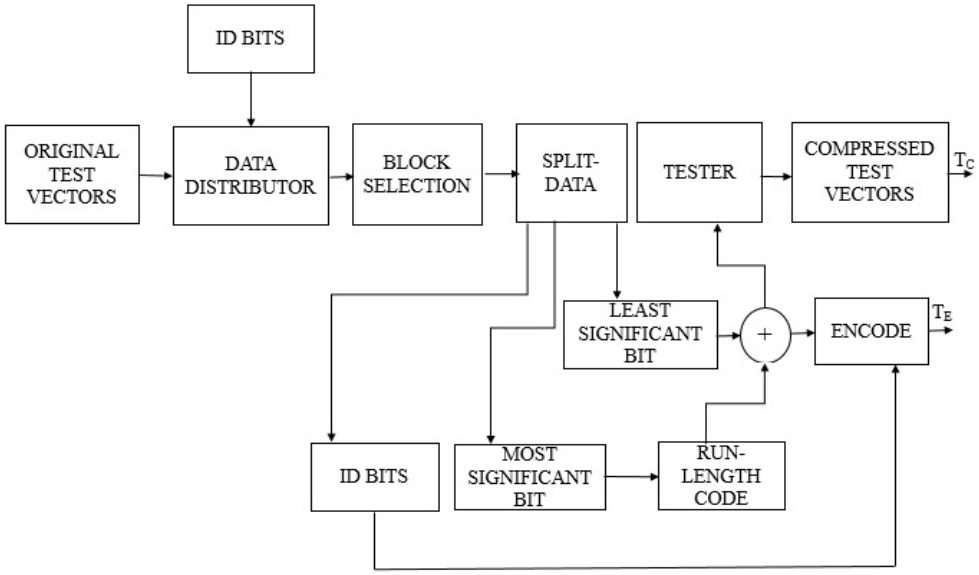

Fig 1: Block Diagram of Proposed Method

Table 3: Selection of Identification Bit pattern

BLOCK-OPTION	3-BIT LENGTH ID BITS(B≤7)
B=0	001
B=1	010
B=2	011
B=3	100
B=4	101
B=5	110

The selection of the blocks is made on the basis of giving ID bit pattern that the selected option will use to split the LSB and encodes remaining higher order bits of current block of samples. An ID bit sequence specifies the selection of option to encode the set of code words. With the same ID bits, the decoder is used for partial implementations of encoded bits. At last, ID bits and concatenation of the LSB and RLC is encoded to generate encoded test data T_E. This results in limited memory, whereas original test vectors results in high memory for storage. The Overall concept of SDV compression is shown in Algorithm 1. At the initial stage, test vectors are generated along with ID bits, this allows to select any one of the split-option and this is repeated until T_D has binary values.

ALGORITHM 1: SDV COMPRESSION

SDV_algo()
begin
generate Test Vectors (T_D) with ID bits
select Split-option Block B
repeat
  begin
  T_C=SD(ID,LSB,MSB)
  MSB=RLC(higher order bits)
  Simulate T_C
  end
  until T_D has binary values
end
until all ID bits
while all T_D have ID bits
begin
  T_E=encode(ID,LSB,MSB)
end
until T_D have binary value
while tester have all split option block B values
begin
  T_{HC}= highest compression of T_D values
repeat
end

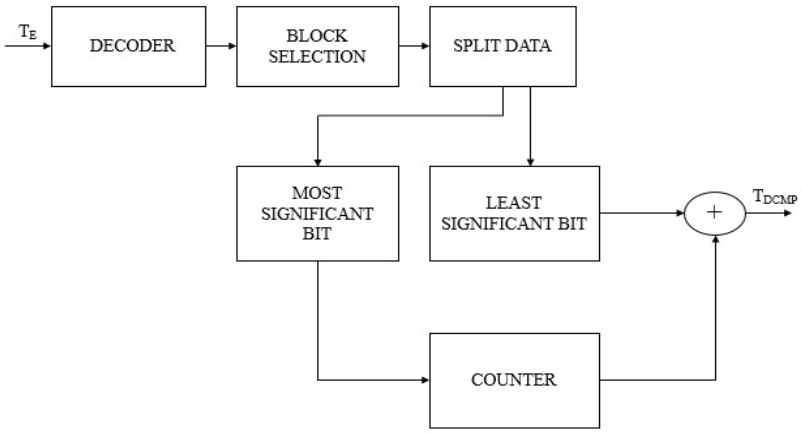

Fig 2: Conceptual Architecture of Decompression

The Figure 2 describes the overall concept of decompression to recover the original test vectors. The encoded bits are decoded with presence of ID bits, to select any one of the blocks based on ID bits. To convert to 8 bit code word, counters are used to count number of 0s until it reaches 1. These numbers of 0s are transferred to code word and finally LSB and converted code word is concatenated to result in the original uncompressed test vector. Algorithm 2 describes the overall concept of decompression to achieve the original uncompressed bits.

ALGORITHM 2: SDV DECOMPRESSION

SDV_algo( )
begin
generate encoded test vectors (T_E)
select split-option block B
B=decode (T_E)
repeat
begin
T_{DCMP}=SD(LSB,MSB)
MSB= counter (higher order bits)
simulate T_{DCMP}
end
until all T_E has binary values

3. RESULTS OF SIMULATION EXPERIMENTS

Test vector generation program was employed to obtain a set of test vectors to provide 100% fault coverage, MINTEST. The percentage of data compression is computed based on

$$\%\,Compression = \frac{Original\ Bits - compressed\ bits}{original\ bits} X\ 100$$

The Experimental results are shown in Table 4 & 5, Table 6, Figure 3 and 4, and Figure 5.

Table 4 and 5 shows compression results obtained from ISCAS 85, ISCAS 89 combinational and sequential benchmark circuits. The scan size is also considered for combinational and sequential circuits and compression achieved is 80%.The column 4 illustrates the original size of test vectors and consequently remaining column shows the compression achieved for split options with various combinational and sequential circuits, which is extended up to a value of 7.

Table 4: Test Vector Compression Percentage Several ISCAS'85 Benchmark Combinational Circuits Using Proposed Technique

Circuit Name	Scan Size	No. of Test Vectors	Total original Bits	Compression % B=2	Compression % B=3	Compression % B=4	Compression % B=5	Compression % B=6	Compression % B=7
C432	36	27	972	44.65	65.63	75.6	79.62	80.96	80.76
C499	41	52	2132	26.6	56.8	71.2	77.67	80.11	80.54
C880	60	15	960	30	58.7	72.5	78.64	81.04	81.45
C1355	41	84	3442	67.5	77.13	81.05	82.24	82.01	81.14
C1908	33	106	3498	42.9	64.95	75.15	79.5	80.8	80.7
C2670	233	44	10252	44.4	65.6	75.3	79.51	80.7	80.62
C3540	50	63	4200	38.72	62.8	74.11	78.9	80.5	80.52
C5315	178	136	6586	42.86	64.84	75.11	79.42	80.74	80.71
C6588	32	94	384	13.2	51.3	69.7	78.3	82.03	83.07
C7522	207	148	15111	43.8	65.2	75.17	79.32	80.6	80.5

Table 5: Test Vector Compression Percentage Several ISCAS 89 Benchmark Sequential Circuits Using Proposed Technique

Circuit Name	Scan Size	No. of Test Vectors	Total original Bits	Compression % B=2	Compression % B=3	Compression % B=4	Compression % B=5	Compression % B=6	Compression % B=7
S420	34	43	1505	49.1	68.03	76.6	80.33	81.3	81.06
S444	24	24	576	30.7	59.2	73.09	79.16	81.77	82.2
S510	25	54	1350	46.22	66.5	75.8	79.8	81.18	80.9
S526	24	49	1176	42.2	64.5	74.9	79.2	80.95	80.8
S820	23	93	2139	45.02	65.9	75.54	79.6	80.87	80.87
S838	67	75	5025	48.2	67.4	76.2	79.9	80.99	80.69
S5378	214	97	20765	37	61.89	73.48	78.5	80.34	80.43
S9234	247	105	25935	41.4	64.07	74.5	79.11	80.5	80.5
S13207	700	233	163100	39.99	63.35	74.26	78.91	80.4	80.4
S15850	611	94	57434	41.2	63.99	74.57	79.08	80.55	80.5
S38417	1664	68	113152	34.44	60.6	72.9	78.3	80.21	80.3
S38584	1464	110	161040	40.08	63.77	74.46	79.04	80.52	80.44

Finally, Figure 3 and 4 graphically shows the test vector compression results for several ISCAS 85 and ISCAS 89 benchmark circuits using SDV coding for various values of B. The graphical diagram shows that the overall average percentage for both combinational and sequential circuits is 80. Table 6 describes the overall compression percentage and time taken to compress the test data under the proposed scheme. Figure 5 shows graphically the time taken to compress the test data in seconds for combinational and sequential circuits.

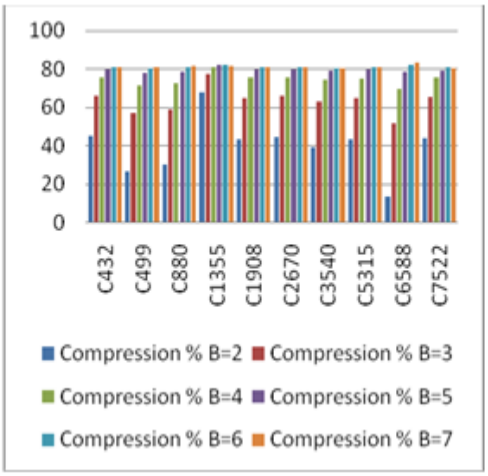

Fig 3: Test Vector Compression Results for Several ISCAS 85 Benchmark Circuits Using Proposed Technique

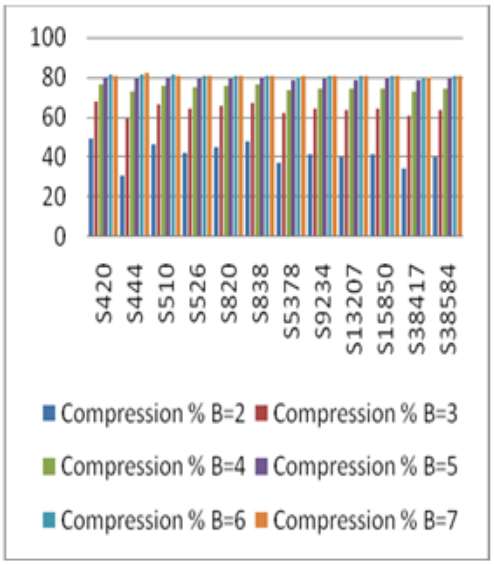

Fig 4: Test Vector Compression Results for Several ISCAS 89 Benchmark Circuits Using Proposed Technique

Table 6: Test Vector Compression Ratio and Time Taken for both ISCAS 85 Combinational Circuits and ISCAS'89 Sequential Circuits

Circuit Name	Total original Bits	Compressed Bits B=6	Compression Percentage Proposed scheme	Time taken to compress the test data(s)
C432	972	185	80.96	0.2055
C499	2132	424	80.11	0.356
C880	960	182	81.04	0.196
C1355	3442	619	82.01	0.533
C1908	3498	671	80.8	0.536
C2670	10252	1972	80.7	1.399
C3540	4200	819	80.5	0.623
C5315	6586	1268	80.74	0.915
C6588	384	69	82.03	0.132
C7522	15111	2925	80.6	2.07

S420	1505	280	81.3	0.266
S444	576	105	81.77	0.162
S510	1350	254	81.18	0.281
S526	1176	224	80.95	0.2353
S820	2139	409	80.87	0.364
S838	5025	955	80.99	0.876
S5378	20765	4080	80.34	3.331
S9234	25935	5037	80.5	3.786
S13207	163100	31824	80.4	22.707
S15850	57434	11166	80.55	8.062
S38417	113152	22387	80.21	16.021
S38584	161040	31358	80.52	22.477

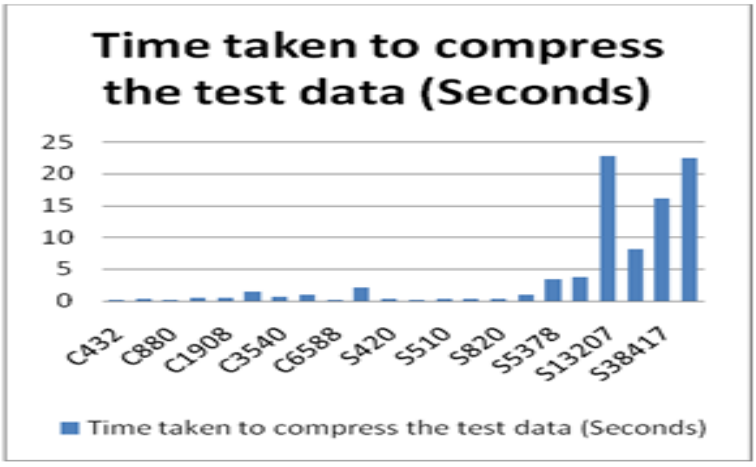

Fig 5: Time Taken to Compress Test Data for ISCAS 85 and ISCAS 89 Benchmark Circuits (Seconds)

Table 7: Test Vector Compression Results for ISCAS 85 and ISCAS 89 Benchmark Circuits Using Other Compression Methods and Proposed Technique

Circuit Name	Huffman [13]	ACB [11]	LZW [12]	BWT [14]	Proposed Scheme
C432	46.58	43.54	45.73	53.29	80.96
C499	69.47	67.12	70.38	80.01	80.11
C880	63.96	56.85	62.63	69.78	81.04
C1355	71.49	64.07	69.79	74.09	82.01
C1908	51.83	46.93	50.52	57.58	80.8
C2670	62.73	58.21	62.29	66.03	80.7
C3540	66.07	64.52	65.83	70.26	80.5
C5315	62.95	61.28	63.79	65.82	80.74
C6588	54.67	48.72	55.08	57.05	82.03
C7522	56.71	51.74	55.83	59.53	80.6
S420	60.52	51.24	59.37	63.72	81.3
S444	32.63	29.32	32.57	32.98	81.77
S510	67.27	63.74	68.08	73.48	81.18
S526	65.86	53.28	65.17	68.19	80.95
S820	64.73	43.93	65.42	67.46	80.87
S838	55.1	54.18	59.06	62.82	80.99
S5378	58.14	-	N/A	-	80.34
S9234	54.2	-	70.26	-	80.5
S13207	77	-	81.69	-	80.4
S15850	66	-	76.26	-	80.55
S38417	59	-	70.6	-	80.21
S38584	64.1	-	75.14	-	80.52

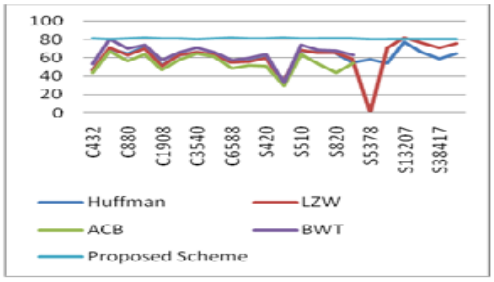

Fig 6: Test Vector Compression Results for Several ISCAS 89 Benchmark Circuits Using Proposed Technique and Other Compression Technique

Table 7 shows the compression results obtained from ISCAS 85 and ISCAS 89 benchmark circuits with comparison of other compression techniques. The Highest compression is obtained compared to Huffman coding, Lempel-Ziv-Welch (LZW) Coding, Associative Coder of Buynovsky (ACB), Burrow-Wheeler Transformation (BWT) presented in Table 7. From the proposed technique, data compression is achieved mainly due to SDV technique in successive test vectors. Figure 6 shows the comparative results of various techniques along with proposed technique. From the comparative experimental results, the developed compression strategy provides better compression results than other various compression methods.

4. CONCLUSION

Since test Data compression is the most important technique it must be lossless and effective. The Compression method herein is based on a Split-Data Variable length (SDV) coding technique that targets to split-off the bits to achieve compression. The technique SDV reduces the test vector sequences, and results in a higher compression ratio. The compression process is done before downloading the test vectors into the on-chip memory. The method proposed helps in reducing the test data and testing time. Further this method proposed herein is completely lossless because of its higher compression ratio.

References

[1] T. Reungpeerakul and D. Kay, "Partial-matching technique in a mixedmode BIST environment,"*IEEE Trans. Instrumen. Meas*., vol. 9, no. 4,pp. 970–977, Apr. 2010.

[2] V. Groza, R. Abielmona, and M. H. Assaf, "A self-reconfigureurable platform for built-in self-test applications," *IEEE Trans. Instrumen.Meas*., vol. 56, no. 4, pp. 1307–1315, Aug. 2007.

[3] D. Kay, S. Chung, and S. Mourad, "Embedded test control schemes using iBIST for SoCs,"*IEEE Trans. Instrumen. Meas*., vol. 54, no. 3, pp. 956–964, Jun. 2005.

[4] S. Biswas, S. R. Das, and E. M. Petriu, "Space compactor design in VLSI circuits based on graph theoretic concepts," *IEEE Trans. Instrumen. Meas*., vol. 55, no. 4, pp. 1106–1118, Aug. 2006.

[5] S. Biswas and S. R. Das, "A software-based method for test vector compression in testing system-on-a-chip," *in Proc. IEEE Instrumen. Meas*. Technol. Conf., Apr. 2006, pp. 359–364.

[6] P. S. Zuchowski, C. B. Reynolds, R. J. Grupp, S. G. Davis, B. Cremen, and B. Troxel, "A hybrid ASIC and FPGA architecture," *in Proc. Int. Conf. Comput. Aided Des*., 2002, pp. 187–194.

[7] M. Abramovici, C. Stroud, and M. Emmert, "*Using embedded FPGAs for SoC yield improvement*," in Proc. Des. Autom. Conf., 2002, pp. 713–724.

[8] S. J. E. Wilton and R. Saleh, "Programmable logic IP cores in SoC design: Opportunities and challenges," *in* Proc. *IEEE Conf. Custom Integr*. Circuits, May 2001, pp. 63–66.

[9] Jas, J. Ghosh-Dastidar, M. Ng, and N. A. Touba, "An efficient test vector compression scheme using selective Huffman coding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 797–806, 2003.

[10] Chandra and K. Chakrabarty, "Frequency-directed run length (FDR) codes with application to system- on-a-chip test data compression," *in Proceedings of the 19th IEEE VLSI Test Symposium*, pp. 42– 47,MarinaDel Rey, Calif, USA, May 2001.

[11] T. Skopal, "ACB compression method and query preprocessing in text retrieval systems," *in Proc. DATESO*, 2002, pp. 1–8.

[12] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression,"*IEEE Trans. Inf. Theory*, vol.IT- 23, no. 3, pp. 337–343, May 1997.

[13] P. T. Gonciari, B. M. Al-Hashimi, and N. Nicolici, "Variable-length input Huffman coding for system- on- chip test," *IEEE Trans.Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 6, pp. 783–796, Jun. 2003.

[14] Satyendra N. Biswas, Sunil R. Das,and Emil M. Petriu, "On System-on-Chip Testing Using Hybrid Test Vector Compression" *IEEE Transactions on Instrumentation and Measurement, Vol. 63, No. 11, November 2014*