

An experimental analysis of the effect of fault dropping on Huffman based test data compression techniques

Sanjoy Mitra

Dept of Computer Science and Engineering
Tripura Institute of Technology
Narsingarh, Agartala, India
mail.smitra@gmail.com

Debaprasad Das

Dept of Electronics and Communication Engineering
TSSOT, Assam University
Silchar, India
dasdebaprasad@yahoo.co.in

Abstract— The speedy rising of system-on-chip (SoC) integration has contributed significantly to the growth of SoC circuit density leading a major increase in test data volume. So, lessening of test data volume is a serious concern of the silicon testing industry and therefore, test data compression emerges as the handiest alternative for reducing test data volume and testing time. This paper presents an experimental analysis of how fault dropping effects the efficiency of Huffman based test data compression scheme. In this paper, we drop a class of faults termed here as ‘true non essential faults’ which is defined as such faults on which no other faults are dependent and whose removal has no ultimate impact on fault coverage, rather dropping of such faults may largely contribute to the significant improvement of compression ratio, test power etc. Care bits of the test cube are modified for dropping of such faults from the fault coverage and based on modified care bit position, don’t care bits are filled to optimize compression ratio. The simulation on ISCAS 89 benchmark test data yields promising improvement of compression ratio when compared with other relevant approaches.

Keywords— *True nonessential faults; fault dropping; care bits; allied vectors; coding viability; dropping feasibility*

I. INTRODUCTION

With the growing up of density of integrated circuits, test data volume has increased spectacularly in scan-based testing of system-on-chips (SoCs) since huge number of pre-computed test patterns are needed to achieve desired high fault coverage and consequently causing manifold increase of test data volume, for which not only testing time is increased but also exceeds the tester memory capacity [1]. Compressing test data is an effectual way of lessening test data volume. The prime objective of test data compression is to reduce the number of binary bits in original test set which can be produced by the automated test pattern generation (ATPG) tool after circuit design. The internal memory of automatic test equipment (ATE) is used to store the compressed test set and an on-chip decoder decompresses the test data from ATE memory and subsequently delivers to the circuit under test (CUT). The overview of this traditional framework is shown in Fig. 1. Design of the on-chip decoder hardware largely depends on the proposed compression algorithm and is suitable for the entire original test sets which are using this compression approach. The test data compression methods can commonly be categorized into three types: code-based schemes,

broadcast- scan-based schemes and linear-decompression-based scheme [3].

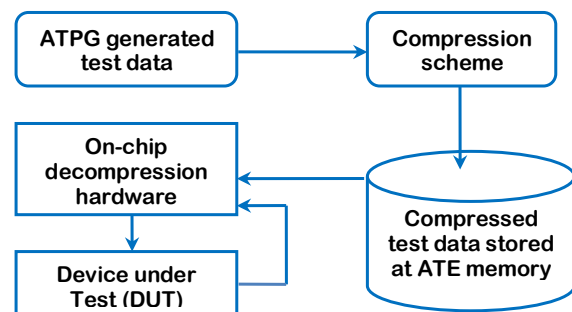


Figure.1. Conventional test data compression framework

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) [4] and optimal selective Huffman coding (OSHC) [5], dictionary-based coding [6] and block merging coding [7] 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 [8], frequency-directed run-length (FDR) code [9] and variable input Huffman code (VIHC) [10]. 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) [11] alternating run-length coding (AFDR) [12] and mixed double

run-length and Huffman coding (RL-HC) [13] are the examples of double run length encoding.

In order to augment the compression ratio, we opt to drop a class of faults termed here as ‘true nonessential fault’ which is defined as such faults on which no other faults are dependent and whose exclusion has no impact on fault coverage, rather dropping of such faults improves compression ratio, test power etc. Care bits of the test cube is modified for dropping of such faults from the fault coverage and based on modified care bit position, don’t care bits are filled to optimize compression ratio. Compression ratio is improved by the systematic flipping of test data bits based on ‘dropping feasibility’ of the applied compression algorithm while keeping fault coverage of the test set unaffected.

In this paper, we apply the fault dropped test data sets on two conventional Huffman based test data compression schemes namely Variable-length Input Huffman Coding and Selective Huffman encoding and analyze their effects on the improvement of test data compression ratio.

II. INTRODUCTION TO HUFFMAN BASED CODING

Test data compression approaches based on Huffman coding has shown promising results towards improvement of test data compression ratio, area overhead and the test application time and examples of such Huffman based coding includes variable-input Huffman coding [14], variable-to-variable Huffman coding [15], optimal selective Huffman coding [16], complementary Huffman coding [17] and modified selective Huffman coding [18] available. The concept of statistical coding relies on the calculation of the frequency of occurrence of each symbol in the test set and most frequently occurring symbols are assigned shorter codewords in terms of number of bits. Table 2 shows the codeword formation for various Huffman coding methods with the 80-bits long test set of 010001100010001001110100010001000010011100100010100011101010100100001010000000. In the category of fixed-to-variable coding, Huffman coding is proved to yield the shortest average codeword length amongst other uniquely decodable fixed-to-variable length codes. Huffman code is prefix-free (i.e., no codeword is the prefix of another one) and due to this, the decoding process is easy and simple. Here the preprocessed test vectors are parted into n -bit blocks to form symbols. These symbols are then compressed using variable-length codewords. An example test vector which is divided into 4-bit symbols is given as 0100 0110 0010 0010 0111 0100 0100 0100 0010 0111 0010 0010 1000 0111 0101 0100 1000 0101 0000 0001 which is of 80 bits. The Huffman tree for the given test vector is constructed as shown in Figure 2. Table 2 shows the frequency of occurrence of all the symbols and the corresponding codewords for each symbol which are obtained from the tree constructed. According to the codeword shown in Table 1, the compressed vector is given as 00 1101 10 10010 00 00 00 10 010 10 10 011 010 111 00 011 111 11000 11001 which is reduced 55 bits. In Huffman coding, the decoder size increases exponentially depending on the symbol size

III. PRELIMINARIES OF FAULT DROPPING

For a target vector and a target region, fault removal is performed.

A. Care Bit Identification Process

Usually, a single test vector covers many faults of the circuit and such faults are covered by setting care bits of the target test vector. Hence, identification of these care bit positions is very significant in the context of dropping of faults from any vector’s coverage. Care bits which either control the value or propagate the value of faults to be observed are identified. TetraMAX[19] is used to run fault simulation. Care-bits are set sequentially and for each setting, fault simulation is performed. After this step, care-bits corresponding to each individual fault is known. With the information, a fault table is constructed where a row and column indices are fault IDs and inputs, respectively. An entry $E_{i,j}$ in the fault table f is 1 if the setting of the care-bit in $column_j$ to 1 X-bit will result in the removal of the fault in row_i . Table.1 shows a fault table where there are three faults and five inputs. $fault_1$ is covered when $cbit_1$ and $cbit_2$ are set and $fault_2$ is covered when $cbit_3, cbit_4$ and $cbit_5$ are set. Similarly, $fault_3$ is covered when $cbit_2$ and $cbit_4$ is set.

Table.1 Fault Table

	$cbit_1$	$cbit_2$	$cbit_3$	$cbit_4$	$cbit_5$
$fault_1$	1	1	0	0	0
$fault_2$	0	0	1	1	1
$fault_3$	0	1	0	1	1

B. True Nonessential Faults

If a fault is covered by more than one test vector then it is here defined as non essential fault [20]. In case of stuck at fault model, removal of one target fault from the test vector cannot be accomplished without considering about such faults which are dependent on it and on the same sensitized path. This spectrum of non essential fault is further constricted into a more filtered form of fault which is defined here as *true non essential fault*[20] A nonessential fault is called true non essential fault if it does not contribute to other essential faults and independent of other faults and its propagation paths.

C. Removal of True Non Essential faults

A fault table is constructed and those care bits which are indispensable for fault dropping is identified. After that, essential care bits needed for true non essential fault dropping is sorted out.

In this step, we will remove only those *true nonessential faults* which may have an impact on the probability of transition in key bit positions of the test set while keeping the fault coverage unaltered. The selection of a candidate *true nonessential fault* for removal depends on the function ‘coding viability’ which is very much specific to any particular encoding algorithm.

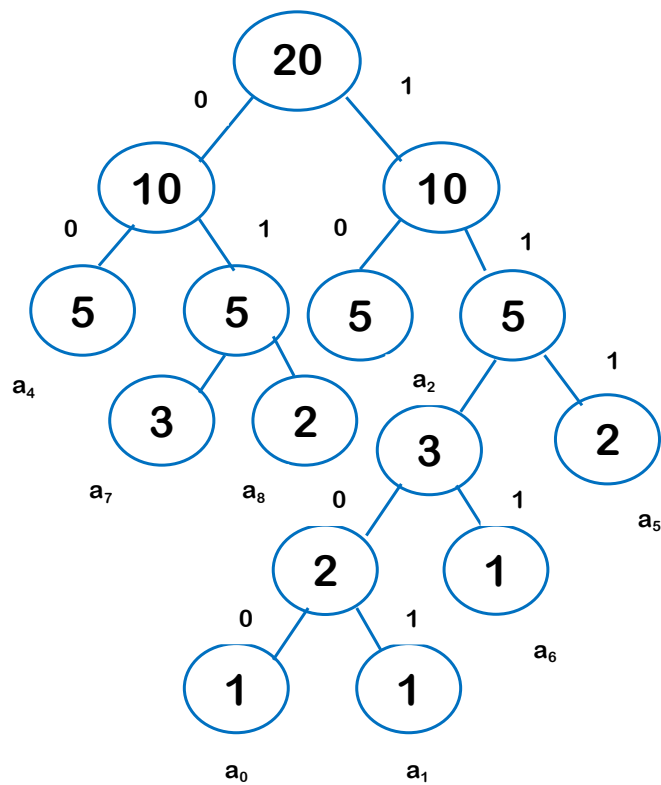


Figure 2: Typical formation of Huffman Tree

Table 2 : Symbol frequencies and the codeword for different Huffman based coding

Symbol	Frequency	Pattern	Huffman	Selective Huffman	Optimum Selective Huffman
a ₁	1	0000	11000	00000	10000
a ₂	1	0001	11001	00001	10001
a ₃	5	0010	-	110	10
a ₄	-	0011	10	00011	10011
a ₅	5	0100	00	10	0
a ₆	2	0110	111	00101	10101
a ₇	1	0111	1101	00110	10110
a ₈	3	1000	010	111	11
a ₉	2	1001	011	01000	11000
a ₁₀	-	1010	-	-	-
a ₁₁	-	1011	-	-	-
a ₁₂	-	1100	-	-	-
a ₁₃	-	1101	-	-	-
a ₁₄	-	1110	-	-	-
a ₁₅	-	1111	-	-	-

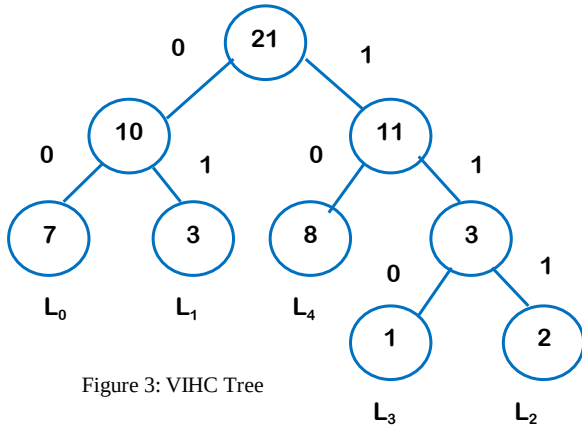


Figure 3: VIHC Tree

Table 3: VIHC codeword and pattern

Pattern	Occurrence	Codeword
$L_0=1$	7	00
$L_1=01$	3	01
$L_2=001$	1	111
$L_3=0001$	2	110
$L_4=0000$	8	10

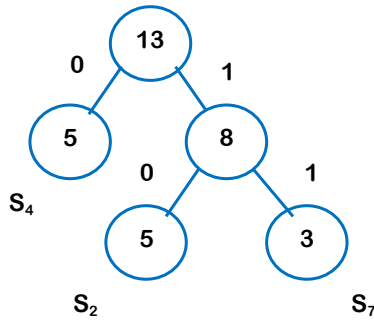


Figure 4: Selective Huffman Tree formation

We define the function ‘coding viability’ for the removal of a fault f_i which has an impact on bit transitions in the target test vector. For a fault, f_i covered by a test data t_d we define its ‘coding viability’ C_v which is the quantity of transitions acting in favour of code length optimization corresponding to a particular encoding mechanism. ‘Allied vectors’ A_v is the number of vectors that covers this target fault f_i . Here we define a term known as ‘Dropping feasibility’ D_{fs} , corresponding to particular fault f_i , which is determined by the product of coding viability and allied vectors to that particular fault.

$$D_{fs}(t_d, f_i) = C_v \times A_v \text{ -----(1)}$$

The more test vectors cover this fault, f_i , the less possible the fault coverage drops. A large ‘Dropping feasibility’ denotes that either this target fault results in a great lessening of the average code length of encoded symbols achieved by the way of care bit modification of test vector corresponding to an applied encoding mechanism or negligible effect on fault

coverage drop after fault removal. We sort these true non-essential fault candidates by the ‘dropping feasibility’ and remove fault with the maximum ‘dropping feasibility’.

It is seen that after X filling in the test cube, there exist some true non essential faults and this category of faults may be dropped from the coverage of respective test set through its care bit manipulation which in turn may act as the efficiency booster of individual encoding algorithms while keeping the fault coverage unchanged. Some care bits in the test set can be manipulated without affecting the fault coverage because of the silent existence of true nonessential faults’ coverage inside the entire coverage. Care bits are manipulated in order to promote the compression; the compression ratio will be improved. Huffman based codes are taken as the examples to illustrate the fundamental process of test set manipulation.

We start with the highest value of ‘Dropping feasibility’ D_{fs} and the removal of true non-essential fault continues until there is no any true non-essential fault is found for removal i.e. $f_{No} \neq Nil$. After that, smallest code length l_{min} is found from the set of code lengths obtained during various iterations.

Algorithm: Test Cube (T) Manipulation

```

Coverage ← Fault Coverage (T)
 $D_{fs} = MAX$ 
WHILE  $f_{No} \neq Nil$  and  $D_{fs} \geq 0$  // condition
FOR care_bit IN T // care_bit in test set T
IF care_bit = 1 // Flipping of care_bit
care_bit ← 0
//if the fault coverage is affected, restore
IF Coverage ≠ Fault Coverage (T)
bit ← 1
Compute code_length(l)
 $MAX \leftarrow MAX - 1$  // next dropping feasibility
Find  $l_{min}$  // minimum code length
END

```

The manipulation of a test set is affected by the factors like average redundancy, ability to cover exact quantity of true non essential faults and position of relevant care bits associated with the coverage of this fraternity of faults in order to raise the frequency of symbols which may result in optimal code length of certain Huffman based algorithms. The different coding method yields different optimal lengths of code on the application of fault dropping strategies. Two Huffman based test data compression algorithms guided by the dropping of true nonessential faults are discussed and analyzed here next.

IV. FAULT DROPPING IN HUFFMAN BASED ALGORITHMS

A. Variable-length Input Huffman Coding (VIHC) scheme

Huffman coding is a statistical data coding method that reduces the average code length used to represent the unique patterns of a set[21]. In this approach, variable length input patterns are used to provide an efficient exploitation of test sets which exhibit long runs of ‘0’s. In Figure 4 a 26-bit test data T_{d-init} was taken as an example to discuss compression. The test data is portioned into groups with the precondition of group size, $G_s = \text{Runs of ‘0’s of length } \leq 4$.

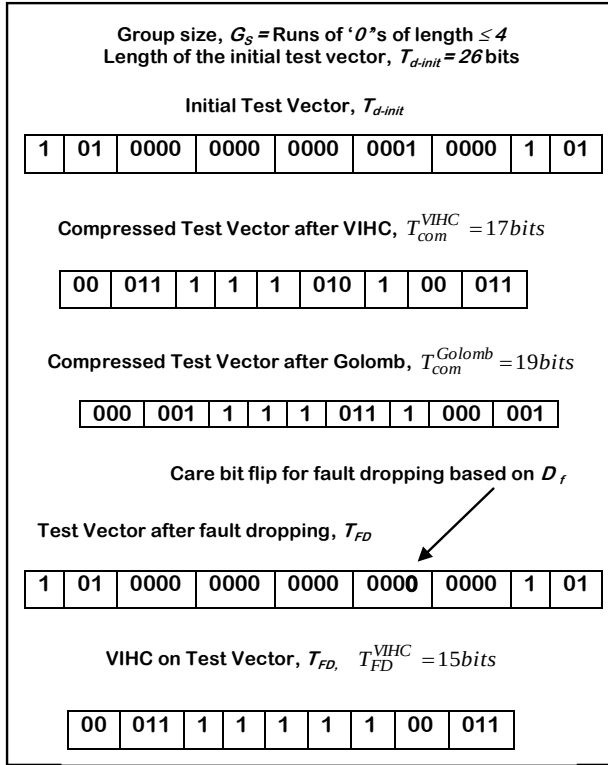


Figure 5: Compression of Test Vectors

This test data T_{d-init} was compressed into 17 bits and 19 bits by using compression methods namely VIHC and Golomb respectively and these compressed test data is denoted by T_{com}^{VIHC} and T_{com}^{Golomb} respectively in Figure 5. A true non essential fault was dropped from the test set T_{d-init} by flipping the 19th care bit on the basis of 'dropping feasibility' D_{fs} and the resulting test vector is represented by T_{FD} . After that VIHC was applied to fault dropped test data T_{FD} and resulting test data was compressed to 15 bits.

It is evident from the figure 5 that significant improvement in the compression ratio of VIHC is achieved through the dropping of nonessential faults from the coverage. VIHC has reduced 26 bits of test data into 17 bits. The care bits of the original test data was manipulated in order to drop the faults from the coverage of the test set and accordingly care bits are flipped depending on dropping feasibility. Compressed test data is reduced to 15 bits after the application of VIHC on fault dropped test data.

The effectiveness of this approach can further be observed from the Table 3 and Table 4 where it can be seen clearly that number of patterns and the average length of the VIHC code is reduced notably in Table 4 depicting the VIHC dictionary after fault dropping. A notable change in VIHC tree structure before and after fault dropping can be observed in Figure 6 and Figure 7 respectively.

Table 4: VIHC Dictionary before fault dropping

Pattern	Occurrence	Code
$P_0=1$	2	00
$P_1=01$	2	011
$P_2=0001$	1	010
$P_3=0000$	4	1

Table 5: VIHC Dictionary after fault dropping

Pattern	Occurrence	Code
$P_0=1$	2	00
$P_1=01$	2	011
$P_2=0000$	5	1

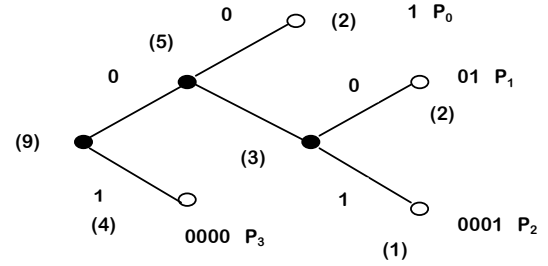


Figure 6: VIHC Tree before fault dropping

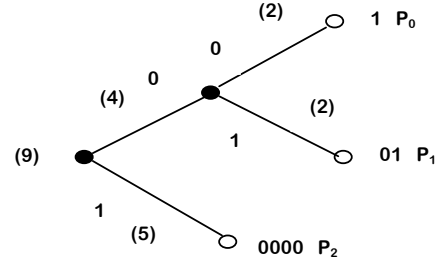


Figure 7: VIHC Tree after fault dropping

We apply fault dropping on test sets of six largest ISCAS 89 benchmark circuits and analyze their effects on the compression efficiency of VIHC and Selective Huffman encoding[23] algorithms.

V. EXPERIMENTAL RESULTS AND ANALYSIS

The already existing compression algorithms are programmed using C++ and test data of six largest ISCAS'89[24] benchmark circuits was given as input to the compression algorithms. Test data was generated using the Synopsis ATPG tool Tetra MAX [19]. We have used Synopsis Tetra MAX to run fault simulation. Care-bits of the test vector are set one after another and during each set operation fault simulation is carried out. Thus, it becomes possible to identify the care-bits corresponding to each individual fault. Afterward ATPG generated test cube is manipulated by dropping true nonessential faults depending on the 'dropping feasibility' of the compression approach. In Table 7 we can see an

improvement in compression ratio ranging from 4.5 % to 7.75% for Variable-length Input Huffman Coding [21] and Selective Huffman [23] Besides, incorporation of fault dropping concept to any test data compression approach does not require any new decompression circuitry and an existing relevant decompress circuit performs decompression efficiently. The compression approaches like VIHC, Selective Huffman coding methods do not require any newer decompression hardware even after the dropping of true non essential faults from the coverage corresponding to each test vectors. As a result, on-chip decompression is performed without any additional hardware overhead.

Table 6: Compression ratio before fault dropping

ISCAS 89 circuit	Variable length Input Huffman Coding[21]	Selective Huffman[23]
s5378	51.76	42.30
s9234	47.22	38.15
s13207	83.54	66.91
s15850	67.90	52.62
s35932	56.09	50.74
s38417	53.35	79.89
s38584	62.20	57.80

Table 7: Compression ratio after fault dropping

ISCAS 89 circuit	Variable length Input Huffman Coding[21]	Selective Huffman[23]
s5378	54.35	44.42
s9234	50.15	40.44
s13207	87.72	69.93
s15850	70.96	55.52
s35932	60.04	53.78
s38417	56.30	85.43
s38584	65.93	62.25

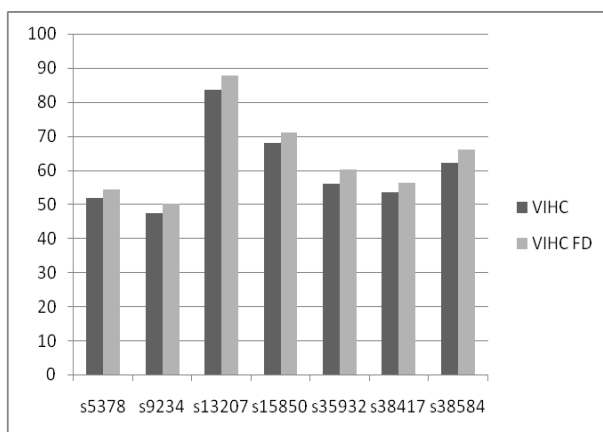


Figure 8 Graphical comparison of VIHC before and after fault dropping

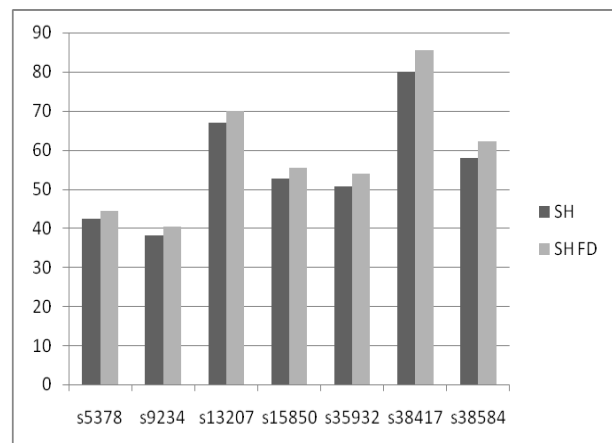


Figure 9 Graphical comparison of SH before and after fault dropping

In figure 8 compression ratio of Variable length Input Huffman Coding [21] is denoted as VIHC and compression ratio of VIHC after fault dropping is denoted as VIHC FD

In figure 9 compression ratio of Selective Huffman Coding [23] is denoted as SH and compression ratio of SH after fault dropping is denoted as SH FD

VI. CONCLUSION AND FUTURE WORK

In this paper, fault dropped test data set is fed as input to the variable length input Huffman and selective Huffman test data compression methods. A 4.5 % to 7.75% compression ratio is improved for both the Huffman based test data compression algorithms. Similarly, this concept of fault dropping may be applied to other test data compression algorithms and the compression ratio may be observed. Besides, removal of nonessential faults reduces switching activity in target regions remarkably, thereby reducing the test mode power dissipation and this at present work may be extended in this direction also.

References

- [1] Usha S. Mehta, Kankar S. Dasgupta and Niranjana 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.
- [2] 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.
- [3] 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
- [4] A. Jas, J. Ghosh-Dastidar, Mom-Eng Ng, and N.A. Toubia "An efficient test vector compression scheme using selective Huffman coding," *IEEE Transaction on. Computer-Aided Design Integration Circuits Systems*, Vol. 22(6) 2003, pp 797-806
- [5] 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
- [6] 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
- [7] 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 Electronic Testing, Vol.29 (6) December 2013, pp 849-859
 - [8] A. 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 Systems, Vol. 20 (3) March 2001, pp 355-368.
 - [9] A. 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
 - [11] A. 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
 - [12] A. 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
 - [13] 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
 - [14] Gonciari, P. T., B. M. Al-Hashimi, and N. Nicolici "Variable-length input Huffman coding for system-on-a-chip test" IEEE Trans. Computer-Aided Design Integrated Circuits Syst. vol 22(6), May 2003, pp 783-796
 - [15] Kavousianos, X., E. Kalligeros, and D. Nikolos "Test data compression based on variable-to-variable Huffman encoding with codeword reusability" IEEE Transactions on Computer-Aided Design Integrated Circuits Syst. vol. 27(7), 2008, pp 1333-1338
 - [16] Kavousianos, X., E. Kalligeros, and D. Nikolos "Optimal selective Huffman coding for test-data compression", IEEE Transactions on Computer, vol.56 (8), 2007, pp 1146-1152
 - [17] Lu, S, H Chuang, G. Lai, B. Lai, and Y. Huang "Efficient test pattern compression techniques based on complementary Huffman coding", In proc of IEEE Circuits and Systems International Conference on Testing and Diagnosis, ICTD 2009.
 - [18] Mehta, U. S., K. S. Dasgupta, and N. M. Devashrayee "Modified selective Huffman coding for optimization of test data compression, test application time and area overhead" Journal of Electronic Testing: Theory and Applications (JETTA), vol. 26(6), 2010, pp 679-688.
 - [19] Synopsys Inc.: Tetra MAX ATPG user Guide, 2006
 - [20] F W Chen, S L Chen, Y S Lin, and T T Hwang, "A Physical-Location-Aware X-Bit Redistribution for Maximum IR-Drop Reduction", IEEE Transactions on Very Large Scale Integration(VLSI) Systems, vol. 20 No. 12, December 2012, pp 2255-2264
 - [21] 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, No 6, June 2003
 - [22] T. Cover and J. Thomas, Elements of Information Theory. New York: John Wiley & Sons, 1991
 - [23] 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
 - [24] 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

