An Innovative Two-Stage Data Compression Scheme using Adaptive Block Merging Technique

Harpreet Vohra[1], Ashima Singh[2] and Sukhpal Singh Gill[3]

[1]Department of Electronics and Communication Engineering, Thapar Institute of Engineering and Technology, Patiala, India
[2]Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala, India
[3]School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
hvohra@thapar.edu, ashima@thapar.edu, s.s.gill@qmul.ac.uk

Abstract

Test data has increased enormously owing to the rising on-chip complexity of integrated circuits. It further increases the test data transportation time and tester memory. The non-correlated test bits increase the issue of the test power. This paper presents a two-stage block merging based test data minimization scheme which reduces the test bits, test time and test power. A test data is partitioned into blocks of fixed sizes which are compressed using two-stage encoding technique. In stage one, successive blocks are merged to retain a representative block. In stage two, the retained pattern block is further encoding based on the existence of ten different subcases between the sub-block formed by splitting the retained pattern block into two halves. Non-compatible blocks are also split into two sub-blocks and tried for encoded using lesser bits. Decompression architecture to retrieve the original test data is presented. Simulation results obtained corresponding to different ISCAS'89 benchmarks circuits reflect its effectiveness in achieving better compression.

Keywords: Test data reduction; Block merging; Code-based testing; Adaptive Block Merging; Data Compression

1. Introduction

The Integrated Circuit (IC) design and fabrication approaches have gone through tremendous advancements leading to the creation of a complex two and three-dimensional System on chip (SoC) designs. Such SoCs may also be equipped with on-chip networks (NoC) to ease out the on-chip interconnectivity. However, the tremendously growing on-chip complexity has led to new test challenges (Wang, L. T. et al 2006). One of the biggest challenges faced by test engineers is the enormously increasing test data volume which is required for fault-free delivery of the product to the market. Pre-generated test sets stored in the memory of the Automatic Test Equipment (ATE) are delivered to the Circuit Under Test (CUT) using scan-based test approach (Wu et al. 2003). The serial transportation of a large number of test bits between the ATE and CUT pins at low ATE clock increases the test application time. The test bits being uncorrelated lead to occurrence of very high switching activity at various scan cells which in turn increase the test power. Hence, a reduction of the amount of test data, test power and test application time are very crucial to reduce the overall test cost of a system. Meanwhile, it is important that the on-chip test infrastructure (decoder etc.) required should not pose an unbearable overhead.

1.1 Analysis of Existing Literature

Various techniques have been proposed in the last three decades which try to reduce the test data by appropriately reusing the unspecified bits in the test data. Such techniques can be broadly classified as *a)* Linear decompressors and broadcast scan-based techniques and *b)* Code based techniques (Wang et al. 2012). The categorization between the two majorly depends upon the fact that the former ones require the structural knowledge of the circuit under test while the code based don't. Linear decompressor-based techniques utilize on-chip hardware to compress the test data by efficiently utilizing don't care bits of the test data. Approaches like Linear feedback shift registers based reseeding (Krishna and Touba, 2002; Koenemann, 1991), scan-based concealment (Bayraktaroglu, and Orailoglu, 2001), and ring counters (Mrugalski 2004), etc. fall under the category of the linear decompressors. Broadcast scan based (Lee et. al 1998) approaches have utilized single channel of the tester to feed in other tester channels. Such approaches reduce the amount of test data which needs to be transported which in turn reduces the test time. Another scan-based approaches namely reconfigured scan forest (Xiang, et al. 2007) and reconfigurable scan architecture with weighted scan enables for determinitic BISTs reduces the test data even more. However, the use of such approaches is discouraged in scenarios like embedded core-based designs wherein the structural details of CUT are not known to the test engineers (Correa and David, 2018). Code based techniques, on the other hand, can be applied directly on the test data to reduce the number of the test bits. An off-chip compressor can compress the test data to obtain an encoded test stream which can be decoded/decompressed using on-chip decompressor hardware. Minimal on-chip test infrastructure and immunity to the underlying structural details have made the code based test data compression to be a promising solution.

Various code-based test data compression techniques (Touba 2006; Vohra, 2018) have been developed so far that attempts to compress the test data on the basis of the run lengths of the various types of test bits (Gill, 2019). Such techniques can be differentiated as statistical code based and run-length based. The statistical techniques like Huffman encoding (Jas *et. al.* 2003; Mehta, *et al*. 2010), Run-Length (RL) Huffman encoding (Nourani, Tehranipour, 2005) and optimal selective Huffman encoding (Kavousianos et al 2007) compress the data depending upon the rate of occurrence of various patterns. Run length code based techniques exploit the various combinations of test bits to shrink the overall test data length. The test data dedicated for digital on-chip testing consists of three types of data values: LOW (*0*), HIGH (*1*) and don't care (*X*). These values are present in various combinations like continuous runs of the same value (0/1/X), a mix of LOWs (*0*s), HIGHs (*1*'s) and don't cares (*X*'s) values etc. Based on the various test vector deterministic approaches for combinational designs, it can be concluded that the consecutive test patterns normally differ in a few number of bits; making 80 to 90% of the total test data of a circuit to consist of don't care bits. The appropriate filling of such bits has been used in the various techniques to compress the test data.

The Golomb Code (Chandra and Chakrabarty, 2001) works on the compression by encoding the continuous runs of *all zeroes*. The approach initially generates a difference vector by taking the exclusive OR of the successive patterns. Owing to the fact that only a few bits differ at various bit positions among the successive test vectors, the EX-OR operation helps to increase the lengths of *zeroes* in different vectors. The codewords are produced on the basis of group prefix and tail selection. However, the compression efficiency suffered in cases of shorter run length of *zeroes.* Such cases lead to an expansion of data instead of compression. It is resolved in the Frequency directed Run-length (FDR) code (Chandra, Chakrabarty, 2003) which makes use of variable group sizes based on different run lengths. An extended FDR code (El-Maleh, A.H. 2008a) works on the achievement of even better compression by encoding runs of recurrent *zeroes/ ones*. Such techniques provide a reasonably fair amount of test data compression. Even better compression was achieved when the researchers started viewing the test data to consist of combinations of continuous *zeroes*/ *ones* or *unique* patterns. A *Unique* combination needs to be retained as it is to avoid any loss of information. The approaches which utilize encoding of block combinations fall under block encoding techniques. The nine coded compression technique (Tehranipoor *et al.* 2005) considers the unique combinations along with the runs of *zeroes* and *ones*. The blocks of fixed length on comparison can be encoded according to the existence of nine different cases: *all zeroes* (*00*), *all ones* (*11*) *all zeroes* followed by *all ones* (*01*), *all ones* followed by *all zeroes* (*10*), *all zeroes* followed by a *unique* combination (*0U*), *all ones* followed by *unique* combination (*1U*), *unique* combination followed by *all zeroes* (*U0*), unique combination followed by *all ones* (*U1*), *unique* combination followed by its compatible sub-block (*UU*). Nine Coded compression technique can be categorized as fixed-length 9C and variable length 9C depending upon the block length chosen (Tehranipoor et al. 2005). Later, it was found that the successive blocks of different lengths can be merged together to obtain a merged block on the basis of existence of compatibility (inverse compatibility). Two bits are said to be compatible (inverse compatible) if they are same (compliment) or either of them is don't care. Hence, two blocks will be compatible if all the bits at same bit positions are compatible with each other. Block merging codes like BM (El-Maleh, A.H. 2008b), Block Merging and Eight Coding (BM-8C) (Wu, 2013), Variable Prefix Run Length (VPRL) (Yuan et al. 2014), Count Compatible Pattern Run-Length (CCPRL) (Yuan et al. 2014), Optimal Selective Count Compatible Run Length (OSCCPRL) (Vohra and Singh. 2016), Hierarchical Block Merging Technique (HBMT) (Vohra and Singh 2018) increase the efficiency by compressing Compatible blocks of fixed sizes. BM-8C (Wu, 2013) integrates the compatibility-based block merging and eight codes based intra-block merging techniques.

As stated earlier, rising test power is another threat for test engineers. Due to concurrent activation of multiple circuit elements, the power dissipation during test mode turns out to be much higher in comparison to that of the normal operation. The non-correlation between the consecutive bits of the test data increases the scan-in and scan-out switching power at the boundary of the circuit under test. Various techniques have been developed to reduce the test vector peak and average values of the scan-in and scan-out powers. Optimum don't care filling and vector reordering approaches have been successfully used to reduce the test power. Alternative Statistical

Run Length (ASRL) (Haiying Y. et al. 2016) and Low Power Switched Capacitor (LPSC) (Sivanantham S. et al., 2014b), Variablelength input Huffman coding (VIHC) (Gonciari et al. 2002), Alternating frequency-directed equal-run-length (AFDER) and runlength based Huffman coding (RLHC) (Sivanantham et al. 2014) techniques are examples of approaches that help in achieving a reduction in the test power in addition to test data compression.

In this paper, a new encoding technique for test data compression is proposed which utilizes the block merging approach for achieving test data compression. The design of the associated test data decompressor is also presented. The proposed technique achieves higher compression ratio for precomputed test sets (independent of the structural details of the circuit under test) resulting in minimization of test application time and memory requirement for the test data.

The paper is structured as follows: Section 2 briefly highpoints the shortcomings of the previous test data compression techniques. Description of proposed Adaptive block merging technique and the associated decompression architecture are given in section 3 and section 4 respectively. Section 5 contains simulation results for the test data compression, Test application time and scan-in power estimates for different ISCAS’89 benchmark circuits. Finally, section 5 presents the conclusions and future directions.

2. Gaps in the Previous Schemes

Literature review reveals that 9C approach brought a kind of breakthrough in the run-length encoding technique as nine different cases were being considered for test data encoding. However, limitation of merging only two blocks at a time hampers the achievable compression efficiency. BM-8C technique emerged to be an efficient approach as it reduces the number of bits by merging subsequent blocks to form a merged block which is further compressed using eight different codes. However, it lags encoding the special cases like *all zeroes (00), all ones(11),* a sub block of *all zeroes* followed by another sub block of *all ones (01)* and one sub block of *all ones* followed by another sub block of *all ones (10)* using lesser number of bits. Another issue associated with the BM-8C technique is that it considers merging the compatible blocks till first inversely compatible block is found ignoring the scope of compatibility with successive block. A codeword is generated and a new search is started on the subsequent blocks to perform further merging. Techniques like CCPRL and VPRL provide better compression by efficiently merging both compatible and inverse compatible blocks. CCPRL technique excels VPRL in terms of test data volume reduction by removing the unnecessary end bits. However, it has a limitation of adding extra bits corresponding to a representation of unique blocks. At the same time, both CCPRL and VPRL don’t utilize the possibility of compression at the sub-block level. The OSCCPRL scheme works on the short comings of CCPRL and enhances the compression efficiency technique by utilizing the compression at the block and sub-block levels. It also reduces the redundant bits used to represent the unique blocks.

3. Adaptive Block Merging Based Test Data Compression Technique

The proposed Adaptive Block Merging based Test data Compression (ABMTC) technique works on the improvement of BM-8C compression efficiency by employing two-stage encoding approach. In the first-stage, it attempts to merge both compatible (inverse) blocks to improvise the compression at the block level. The information about the number of blocks being merged and the type of compatibility between retained pattern block and the ones being merged is preserved in the form of *count_code* and *relation bits* respectively. In second stage, the retained pattern is further compressed at the sub-blocks level. Finally, if a pattern cannot be merged with its subsequent block, it is treated as *unique (UV)*. It is also examined for compression at the sub-block level. An *inter/Intra bit* is used to signify if the block can be merged with its subsequent blocks or not. If the block can be merged, the inter/intra block bit is set to high ‘*1’* else, it is set to low ‘*0*’’ to represent uniqueness. The overall scheme can be explained with reference to the flowchart shown in fig1. The detailed description of the encoding process is as follows:

3.1 Stage One Encoding

As shown in Figure 1, a pattern block (*b* bit long) is initialized and compared with its subsequent block. Codeword used to encode the inter-block merging consists of the *retained pattern (pattern_code)*, *count_code* and the *relation code* (compatibility code). Out of these, the *retained pattern* is the representative block obtained after merging the subsequent blocks, *count_code* represents the count of blocks being merged and the *relation code* (compatibility code) consists of an array of length equal to the decimal count held in *count_code* and its values is defined by the type of compatibility between the *retained pattern* and block being merged (successor).

The relation bit is chosen to ‘*1*’ or ‘*0*’ to show the existence of compatibility or inverse compatibility between the *retained pattern* and the successor.

3.2 Stage Two Encoding

The *retained pattern* block so formed, is partitioned into two halves (of length to *b/2* each) to form two Half-Length Blocks (HLBs) which are further compared against each other. The two HLBs are compared against each other to examine for the existence of one of ten subcases, namely: *U0/U1/0U/1U/01/00/10/11/ half_comp/ half_invComp.* The categorization between various subcases is done using three bits called *unique_intra_merge_prefix bit (UIMP)* and *tail* bits. The *UIMP* bit is set to ‘*0’* and ‘*1’* corresponding to subcases *U0/U1/0U/1U/01/00/10/11 and half_comp/ half_invComp* respectively. The tail bits are further utilized to encode the various subcases as shown in Table 1.

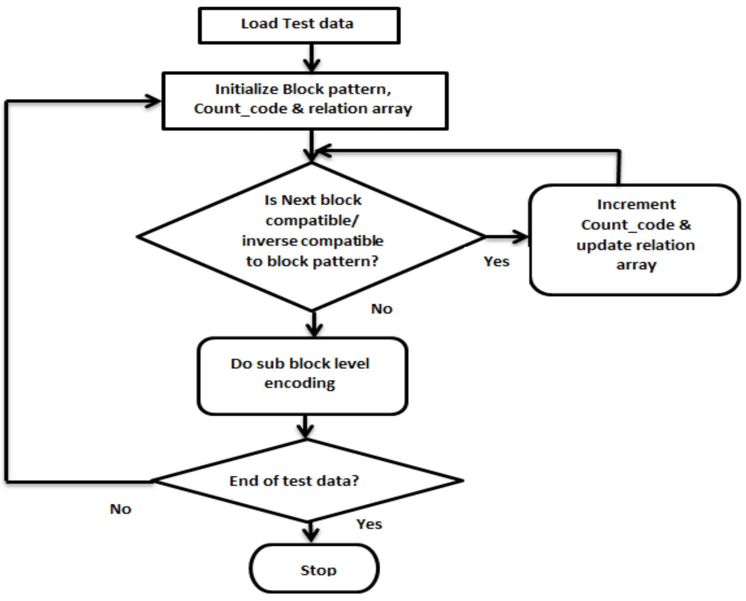
Load Test data
Initialize Block pattern, Count_code & relation array
Is Next block compatible/ inverse compatible to block pattern?
Yes
Increment Count_code & update relation array
No
Do sub block level encoding
End of test data?
No
Yes
Stop

Figure 1: Flow chart of working of ABMTC

The compression efficiency can be improved further by selecting different block lengths while merging the blocks. This is done as follows: Each time inter block merge length has to be selected, block lengths of different values are iteratively tried to find the compression efficiency. The best value is chosen and saved as the *preamble* code. It represents the length of the block being chosen for the codewords. The criteria for selecting the pattern lengths are as follows: a) it has to be below five bits as the frequency of occurrence of compatibility among the pattern blocks of length more than 32 is very less; b) pattern blocks of larger sizes need larger buffer lengths in the decoder design which increases their area and c) Block length chosen should be an even value so that it can be subdivided into two equal sub-blocks easily. To ensure the test length to be a multiple of the chosen block length, extra zeroes are appended at the end of the test data without hampering the test information. Once the blocks are encoding as per the inter/ intra block merging, the leftover don’t care bits (if any) are filled with the same value as that of their preceding bits to reduce the switching activity.

Table 1: Encoding scheme of ABMTC scheme

Inter/ Intra	UIMP	Tail	Codeword	Sub-cases	Inference
0 (Unique Block)	**0**	**100**	**0_0_100_ b/2 bits**	***U0***	**unique HLB followed by *all zeroes***
		101	**0_0_101_ b/2 bits**	***U1***	**unique HLB followed by *all ones***
		110	**0_0_110_ b/2 bits**	***0U***	***all zeroes* followed by a unique HLB**
		111	**0_0_111_ b/2 bits**	***1U***	***all ones* followed by a unique HLB**
	0	**000**	**0_0_000**	***00***	***all zeroes***
		001	**0_0_001**	***01***	***all zeroes* followed by *all ones***
		010	**0_0_010**	***10***	***all ones* followed by *all zeroes***

		011	0_0_011	*11*	*all ones*
	1	10	0_1_10_ b/2 bits	*UU*	Unique Compatible HLBs
		11	0_1_11_ b/2 bits	*UU'*	Unique Inverse Compatible HLBs
	1	0	0_1_0_ b bits	*UV*	Unique block (No compatibility at HLB level)
1 (Block level merging)	0	100	1_0_100_ b/2 bits _*count-code_relation bits*	*U0*	Block merging with *U0* subcase at HLB level
		101	1_0_101_ b/2 bits _*count-code_relation bits*	*U1*	Block merging with *U1* subcase at HLB level
		110	1_0_110_ b/2 bits _*count-code_relation bits*	*0U*	Block merging with *0U* subcase at HLB level
		111	1_0_111_ b/2 bits _ *count-code_relation bits*	*1U*	Block merging with *1U* subcase at HLB level
	0	000	1_0_000 _ *count-code_relation bits*	*00*	Block merging with *00* subcase at HLB level
		001	1_0_001_ *count-code_relation bits*	*01*	Block merging with *01* subcase at HLB level
		010	1_0_010 _ *count-code_relation bits*	*10*	Block merging with *10* subcases at HLB level
		011	1_0_011 _ *count-code_relation bits*	*11*	Block merging with *11* subcases at HLB level
1	1	10	1_1_10_ b/2 bit _ *count-code_relation bits*	*UU*	Block merging with *UU* subcase at HLB level
		11	1_1_11_ b/2 bit _ *count-code_relation bits*	*UU'*	Block merging with *UU'* subcase at HLB level
	1	0	1_1_0_ b bits_ *count-code_relation bits*	*UV*	Block merging with *unique UV* retained pattern

An example to show the implementation of ABMTC code, a random test sequence of 110 bits: '*00XX00000X1010100111X111X11X10X111X11X111011101011XXX1101XXX01XXXXX100XXX1101XXXXXX 1101XXXXXX1101XXX000XX11111*' is chosen. To ease out the explanation, the block length has been fixed to be 10 bits resulting in the formation of 10 blocks (as shown in column 2 of table 2). The codewords so developed are shown in second column of Table 2.

Table 2: Example of ABMTC

BLOCKS	Bit pattern (T_D)	Codeword and length Using ABMTC (T_E)	T_{E_ABMTC} (bits)	Codeword and length Using BM-8C	T_{E_BM-8C} (bits)
1	00XX0-0000X	0_0_000 Unique block with *00* bit pattern	5	0_11_00000 Unique block with *Half_c* bit pattern	8
2	10101_00111	0_1_0_ 1010100111 Unique block with *UV* bit pattern	13	0_00_1010100111 Unique block with *UV* bit pattern	13
3	X111X_11X10	0_0_111_ 11X10 Unique block with 1U bit pattern	10	0_01101_11X10 Unique block with 1U bit pattern	11
4	X111X_11X11	0_0_011 Unique block with 11 bit pattern	5	0_11_11111 Unique block with *Half_c* bit pattern	8
5	10111-01011	1_1_0_10111-01011_101_01001 Six blocks merge with UV bit pattern	21	10_0_00_1011101011 Two block merge with unique bit pattern	15
6	XXX11-01XXX				
7	01XXX-XX100			10_1_010 Two inverse block merge	6
8	XXX11-01XXX				
9	XXX11-01XXX			10_0_11101_01XXX Two block merge with 1U bit pattern	13
10	XXX11-01XXX				
11	000XX-11111	0_0_001 Unique block with 01 bit pattern	5	0_10_11111	8
TOTAL BITS AFTER ENCODING		59		82	

For better illustration of outperformance of ABMTC over BM-8C, outcomes of both the schemes have been shown in columns 3 and 4 respectively. As evident from row number 13, the sequence of 110 bits has been reduced to 59 and 82 bits after application of ABMTC and BM-8C are 59 respectively. In the above example (Table 2), *Preamble* width used to denote the block length has been chosen to be 4 bits here, which justifies the use of any block length (even value strictly) ranging from 1 to 24-1. To get better compression, variable block lengths can be selected based on the best possible compression at each attempt to decompress the test data. Fig. 2 shows the statistics of the occurrence of eleven different subcases for the benchmark circuits for the bit length of 10 bits. It may be noted that the statistics will vary depending on the different bit lengths.

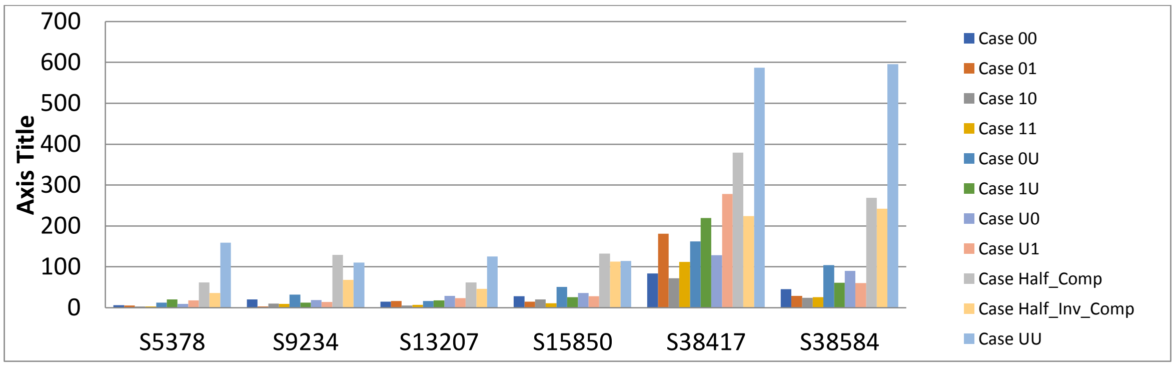

Figure 2: Frequency of occurrence of ten sub cases at intra block level for various benchmark circuit.

Also, it may be observed that the use of the bits for the *count_code* and *relation* bits become more beneficial wherein the occurrence of the successive compatible / inverse compatible blocks is more. Fig 3. shows the occurrence of the inter block merging for 3, 4 and 5 bit *count_code's*, or in other words occurrence of merging 1-7, 1-15 or 1-23 number of blocks being merge.

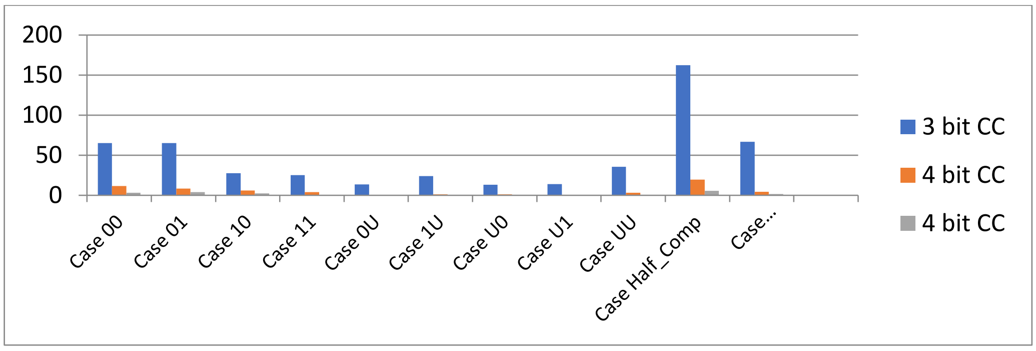

Figure 3: Statistics of various sub cases among the inter block merging for different *count_code* (CC) values.

4. Decompression Architecture

The decompression design for the retrieving the original test stream is quite straightforward. Herein, the CUT is considered to be single scan chain based testable design. Its conceptual view is shown in Figure 4. It comprises of an FSM, two counters, a 32-bit buffer register, 4-bit MUXes (one for each bit of the block length) and a shift register. A 32 bit is chosen as the maximum buffer size to limit the possibility of overhead. The whole design is fed with four incoming signals: *TestClk*, *SoCClk*, *TestData_in*, *Sync* and outgoing signal: *TestData_out*. The compressed data stream *TestData_in* is provided to the decompressor at the ATE frequency (*TestClk*), which after decompression is delivered to the CUT (SOC) at its functional clock *SoCClk* through *TestData_out*. The function of the *Sync* signal is to synchronize the ATE and SOC clocks. The circuit has two counters: counter1: used to receive the block/sub-block length and counter 2: used to extract the relation bits with its count specified by the count code (used for inter-block merging case). The working of each block is as described below:

FSM generates the necessary control signals depending upon the status of the received codewords. Quite straightforward from its working point of view, it gets disintegrated into two categories on the basis of inter/intra bit. A partial diagram of the FSM is shown in Figure 5. If the value of inter/intra is set to *high* and *low*, it reflects the existence of inter-level block merging and *unique* block respectively.

Based on the control signals generated by the FSM signifying, the scratch register is fed with *b* bit data (*UV* case), b/2 bit length (*UU/UU'/U0,U1,0U and 1U* sub cases) through the *TestData_in* signal. The *sel[1:0]* bits received from the FSM further specify whether the shift register has to be fed with *0/1/b/*compliment of *b*. It may be noted that the 4- bit MUX shown in the Fig. 4 signify group of Muxes dedicated for each bit to be filled in the shift register to complete the test block information.

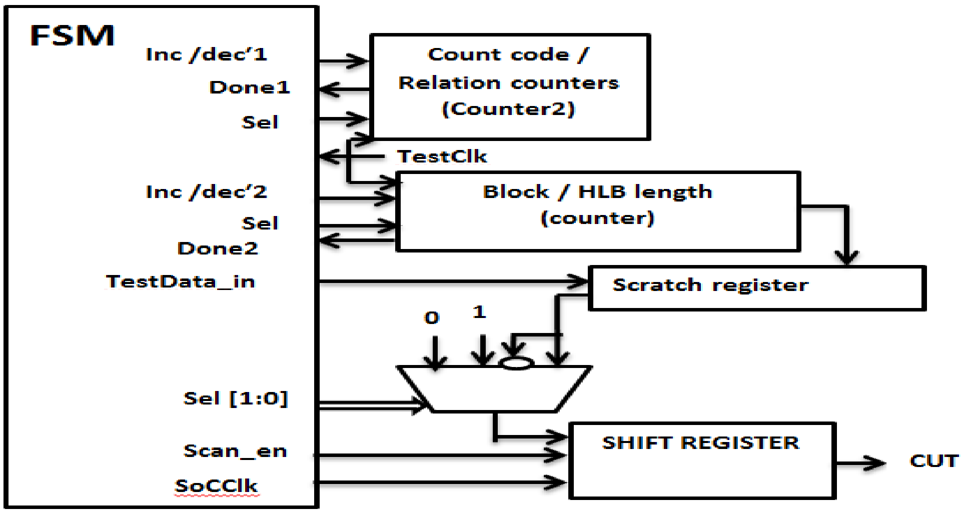

Figure 4: Decompressor architecture

In case of the intra block (HLB) merging, the status of the group code and *UIMP* bits determines the status of the *sel[1:0]* bits which in turn control the working of MUX and counters used in the decompressor design. The *sel[1:0]* bits are set to *10* levels to signify the occurrence of *inverse compatible* case.

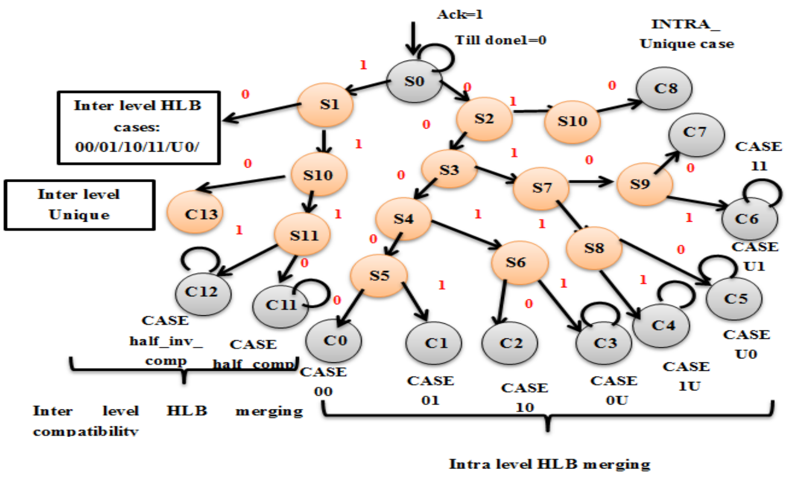

Figure 5: FSM of Decompressor

The decompression process is initiated by receiving the *preamble* (block length). The moment it is done, a value is held internally. The upcoming bits signify the occurrence of different cases and subcases (as shown in table 1). The FSM generates the respective select/control signals. For the cases of *UU/UU'/U0/U1/0U/1U,* the counter 1 is enabled to count till Half the block. The signal *done1* is set to remain in *Low* state from the start of the counting process to the end. Until a *High* on the *done1* signal of the counter1 is not received, the data on the *TestData_in* is forwarded to the scratch register. This fills the value of the *unique* bits of the HLB while the rest of the bits and their placement in the scratch register are specified based on the appropriate signals generated by the FSM. In the case of Interblock merging, counter 2 is made to count down from the *count_code* value received. Meanwhile, at each count pulse (from the start of the countdown to the occurrence of *High* on done2 signal), the status of the *relation code* bits decides whether the value of the scratch register or its complement has to be forwarded to the circuit under test. In case of the fixed block length the preamble bits once received are expected to represent its block length else, the moment the *counter2* gets decremented to zero, a reset is generated for FSM leading to a scan in of a new block length. The test data so retrieved is applied to the circuit under test using the *TestData_out* signal, *SoCclk*, and *scan_en* signal. In case of non-occurrence of any compatibility among the sub-blocks of the pattern, the counter1 helps to retrieve the original data packet held intact in the encoded stream.

5. Experimental Results

In order to validate the efficiency of the proposed technique, comparisons are done with previous techniques using benchmark circuits. The test sets generated by Mintest Automatic Test Pattern Generator (ATPG) (Hamzaoglu, 2009) for Six large ISCAS' 89 benchmark circuits are taken as input and fed to the various compression algorithms.

5.1 Compression efficiency

The compression efficiency (in %age) of an encoding scheme can be calculated by using Equation 1.

The compression efficiency (η) = $\frac{\text{uncompressed data (TD)} - \text{compressed (TE)}}{\text{uncompressed data (TD)}} * 100$ (1)

The value of η obtained for different benchmark circuits is shown in Table 3. To show the performance of the ABMTC, results of other schemes like Golomb (Chandra and Chakrabarty, 2001), FDR (Chandra and Chakrabarty, 2003), Enhanced False Discovery Rate (EFDR) (El-Maleh, 2008a), 9C (Tehranipoor et al. 2005), BM (El-Maleh, 2008b), CCPRL (Yuan et al. 2014), 2n-PRL (Pattern Run-Length) (Chang et al. 2012), BM-8C (Wu, 2013) have also been included in Table 3. The uncompressed test data as per the Mintest ATPG are shown in column1. By comparing the columns 2-8, it can be stated that the ABMTC outperforms other techniques in most of the cases. It may be noted that the block length was chosen to be of five bits which can support variable block lengths (as explained in section 3) and is best suitable for achieving higher compression.

5.2 Comparison of the decoder area

The hardware overhead of the decoder of ABMTC (modelled using Verilog HDL and synthesized using Encounter Register Transfer Language (RTL) compiler from Cadence with 1.8 V, TSMC 180 nm CMOS standard cell library) is presented in Table 4. The full-scan ISCAS'89 benchmark circuits are synthesized with a single scan chain. The decoder area overhead can be calculated using equation 2.

Area Overhead= $\frac{Area\ of\ decoder}{Area\ of\ benchmark\ circuit} * 100$ (2)

By analysing the results obtained, it is apparent that the decoder overhead of ABMTC is close to CCPRL. Though, it is larger than 9C but, the advantage of increased compression efficiency advocates its application.

5.3 Comparison of Test Application Time (TAT)

As explained in section 4, the compressed data is delivered to the SoC periphery at ATE clock frequency (*TestClk*). The decompressor helps to retrieve the original test stream which is applied to the circuit under test at *SoCClk*. To ensure the synchronization between the two clocks, *SoCClk* is chosen to be an integer multiple of *TestClk*. Let the frequency ratio be α = *SoCClk / TestClk* as given in Equation 3. Assuming the compressed test data has *M* code words C(1)−C(m) and each codeword has a length of W(*i)* (i = 1, 2, ……,m).

Let $\alpha_{\max} = \max_{2<i<M}(H(i-1)\ /W(i)\ )$ (3)

where H(i-1*)* is the length of decompressed test data for the codeword C(i-1). If α ≥ αmax, minimum TAT can be calculated as (Yuan et al. 2014, Wu, 2013).

$\text{TATmin} = \sum_{i}^{M} W(i) \quad + ([\max(H(M) - (\alpha - 2)])/\alpha$ (4)

If α < αmax, the ATE will be stalled several cycles to wait for the SoC to apply the decoded test data, which occurs when the time consumed for ATE to send the codeword C(i) to Finite State Machines (FSM) is shorter than the time consumed for the CUT to apply the decoded test data of the previous codeword C(i-1). Then, the total TAT will be calculated as:

$$TAT = TATmin + \sum_{i=2}^{M}\{\max(H(i-1) - w(i) * \alpha, 0\}/\alpha \tag{5}$$

The TAT values calculated as per equations 4 and 5 for different benchmark circuits and α values are presented in Table 5. Results are compared with the TAT obtained for other compression techniques like FDR, EFDR, BM, BM-8C, as evident from the table, proposed ABMTC offers much-reduced Test application time. The process of parallel delivery of test data from ATE and its application to the CUT helps in the reduction of the test application time.

5.4 Comparison of switching power dissipation

One common metric used to estimate the test power is the Weighted Transitions Metric (WTM). The WTM is strongly correlated to the switching activity in the internal nodes of CUT during scan-shift operation. Chandra, and Chakrabarty, 2003b showed experimentally that scan vectors with higher WTM dissipate more power in CUT. Let us say that a scan chain of length t is being dealt with and a scan vector $l_j = l_{j,1}l_{j,2}\ldots\ldots l_{j,t}$, and $l_{j,1}$ is scanned before l_{j2}. The value of the WTM can be calculated for inputs as well as their responses by the following equation

$$WTM_j = \sum_{i=1}^{t-1}(t-i)(l_{j,i} \oplus l_{j,i+1}) \tag{6}$$

With the help of the above equation, we can also calculate the peak (P_{peak}) and the average (P_{avg}) power as follows:

$$P_{avg} = \frac{\sum_{j=1}^{n}\ \sum_{i=1}^{t-1}(t-i)(l_{j,i} \oplus l_{j,i+1})}{n} \tag{7}$$

$$P_{peak} = \max_{j\in(1,2..n)}\left\{\sum_{i=1}^{t-1}(t-i)(l_{j,i} \oplus l_{j,i+1}\right\} \tag{8}$$

Equations 6-8 can be used to calculate the average and peak power for four different benchmark circuits. To results of the scan in peak and average power as obtained for different benchmark, circuits have been compared with other schemes like FDR (Chandra and Chakrabarty, 2003), EFDR (El-Maleh, 2008a), LPSC (Sivanantham S. et al., 2014b) as shown in Table 6. It may be noted that reduction in the switching power has occurred by filling don't care bits with 00/01/10 and *11* patterns thereby reducing the unnecessary transitions. Also, don't care bits left after performing the block merging are filled with same status as that of neighbouring bits to avoid extra switching. It may be noted that the peak and average power consumption, however, suffers in comparison to what has been achieved using LPSC. Better results could have been achieved if test vectors were initially re-arranged on the basis of occurrence of don't care bits such that run lengths of *all zeroes* or *all ones* could be increased. However, for that one needs to ensure that the order of test vectors dedicated for sequential circuits is not altered else, the fault coverage may suffer.

Table 3: Comparison of compression efficiencies (in percentage) achieved between ABMTC compression schemes and the various other techniques

		Test data compression technique							
Benchmark circuit	**Mintest**	**Golomb**	**FDR**	**EFDR**	**9C**	**B.M**	**2n-PRL**	**B.M-8C**	**ABMTC**
S5378	**23754**	**37.11**	**48.02**	**53.67**	**51.64**	**54.98**	**54.94**	**58.56**	**59.47**
S9234	**39273**	**42.25**	**43.6**	**48.66**	**50.94**	**51.19**	**57.72**	**57.49**	**61.28**
S13207	**165200**	**79.74**	**81.3**	**82.49**	**82.31**	**84.89**	**88.1**	**87.52**	**86.49**
S15850	**76986**	**62.82**	**66.22**	**68.66**	**66.38**	**69.49**	**64.29**	**73.69**	**74.57**
S38417	**164736**	**28.37**	**43.26**	**62.02**	**60.63**	**59.39**	**58.33**	**59.92**	**62.12**
S38584	**199104**	**57.17**	**60.93**	**64.28**	**65.53**	**66.86**	**72.44**	**71.66**	**74.25**
Average		**51.24**	**57.22**	**63.29**	**62.38**	**64.46**	**65.97**	**68.14**	**69.69**

Table 4: Comparison of decompression area overhead for ABMTC with other compression techniques

Benchmark circuit	**FDR**	**BM-8C**	**9C**	**CCPRL**	**HBMTC**	**OSCCPRL**	**ABMTC**
S5378	**7.8**	**12.8**	**8.2**	**9.6**	**12.5**	**10.7**	**10.1**
S9234	**5.9**	**9.7**	**6.2**	**7.3**	**9.2**	**8.3**	**8.3**
S13207	**3.5**	**5.8**	**3.7**	**3.5**	**5.10**	**4.2**	**3.8**

S15850	**3.6**	**5.9**	**3.8**	**3.7**	**4.5**	**3.9**	**3.9**
S38417	**1.4**	**2.3**	**1.5**	**1.8**	**2.2**	**2.2**	**2.3**
S8584	**1.5**	**2.5**	**1.6**	**1.9**	**2.5**	**2.5**	**2.6**

Table 5: Comparison of Test application time achieved for ABMTC with other compression techniques

Circuits	**α**	**FDR**	**EFDR**	**BM**	**BM-8C**	**ABMTC**
S5378	**2**	**24,933**	**17,075**	**16,018**	**15,088**	**14,780**
	4	**16,803**	**13,172**	**12,239**	**11,191**	**11,334**
	6	**15,259**	**12,096**	**11,183**	**10,348**	**10,040**
	8	**14,039**	**11,652**	**10,899**	**10,089**	**9,780**
S9234	**2**	**42,039**	**26,129**	**26,336**	**24,281**	**22,266**
	4	**29,206**	**21,424**	**20,828**	**18,410**	**17,986**
	6	**26,675**	**20,557**	**19,762**	**17,278**	**16,870**
	8	**24,086**	**20,318**	**19,436**	**16,921**	**15,666**
S13207	**2**	**1,16,101**	**88,487**	**88,045**	**87,319**	**86,845**
	4	**70,361**	**52,711**	**50,784**	**49,730**	**48,670**
	6	**57,089**	**41,898**	**39,177**	**38,138**	**36,226**
	8	**48,358**	**36,946**	**33,768**	**32,326**	**31,320**
S15850	**2**	**65,020**	**46,076**	**46,076**	**44,110**	**43,910**
	4	**42,270**	**32,517**	**32,084**	**29,553**	**28,234**
	6	**36,732**	**28,798**	**28,216**	**25,522**	**24,234**
	8	**32,362**	**27,172**	**26,518**	**23,673**	**23,673**
S38417	**2**	**1,86,261**	**1,04,569**	**1,09,180**	**1,06,725**	**1,06,100**
	4	**1,23,700**	**75,614**	**80,273**	**79,074**	**78,556**
	6	**1,13,451**	**68,212**	**73,286**	**71,564**	**70,443**
	8	**1,10,521**	**65,509**	**70,202**	**69,069**	**68,566**
S38584	**2**	**1,79,530**	**1,19,849**	**1,18,844**	**1,13,821**	**1,12,661**
	4	**1,18,628**	**86,320**	**83,255**	**76,161**	**74231**
	6	**1,04,630**	**78,066**	**73,953**	**66,048**	**65336**
	8	**93,260**	**74,955**	**70,692**	**61,908**	**60,234**

Table 6. Scan-in peak-power (P_{peak}) and average-power (P_{avg}) transitions: Comparison with other compression methods.

	Mintest		**FDR**		**EFDR**		**LPSC**		**ABMTC**	
Benchmark circuit	**P_{peak}**	**P_{avg}**	**P_{peak}**	**P_{avg}**	**P_{peak}**	**P_{avg}**	**P_{peak}**	**P_{avg}**	**P_{peak}**	**P_{avg}**
S9234	**17494**	**14630**	**12994**	**5692**	**12062**	**3469**	**12102**	**3512**	**12200**	**3524**
S13207	**135607**	**122031**	**101127**	**12416**	**97613**	**8016**	**97685**	**7849**	**97780**	**7888**
S15850	**100228**	**90899**	**81832**	**20742**	**63494**	**13394**	**63586**	**13498**	**63760**	**13834**
S38417	**683765**	**601840**	**505321**	**172665**	**404654**	**117834**	**404676**	**112235**	**405447**	**112244**
S38584	**572618**	**535875**	**234233**	**136634**	**479547**	**89138**	**479748**	**89428**	**478468**	**89650**
Average	**301942**	**273055**	**187101**	**69630**	**211474**	**46370**	**211559**	**45304**	**211531**	**45428**

6. Conclusions and Future Scope

The test data compression is a very promising technique to reduce the test data volume and challenges of test application time. This paper proposed an adaptive block merging technique for test data compression. It improves the test data compression efficiency being immune to the underlying structural details of the circuit under test. As per the simulation results of the application of the ABMTC on various ISCAS'89 benchmark circuits, it can be seen that the average compression efficiency is increased by 2-20% in comparison to the previously proposed techniques. The average and peak test powers can also be reduced by employing this technique. Being very small, it seems to be feasible at the deep submicron level. As evident from the experimental results, the test application time is also reduced by 20% using this scheme. Although, ABMTC helps achieves better data compression but, it can still be worked upon to enhance its ability to reduce test power. Based on the fact that test power is dependent on the switching activity, the test vectors can be divided into two categories. First category includes the test vectors which can help in saving power by increasing run lengths of zeroes and ones while the second helps in reduction of test data. Statistical and Code based approaches can be amalgamated to enhance the power saving and data compression.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank Dr. Nur A. Touba of Texas University, USA for providing the test vectors for various benchmark circuits. Also, we would like to thank the Editor-in-Chief (Prof. S. Tan), area editor (Prof. Henry Chen) and anonymous reviewers for their valuable comments and suggestions to help and improve our research paper.

References

Bayraktaroglu, I. and Orailoglu, A. (2001) 'Test volume and application time reduction through scan chain concealment', *Proceedings of design automation conference*, pp 151-155.
Chandra, A. and Chakrabarty, K. (2001) 'System-on-a-chip data compression and decompression architecture based on Golomb codes', *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems,* Vol. 20, No. 3, pp. 355–368.
Chandra, A. and Chakrabarty, K. (2003a) 'Test data compression and test resource partitioning for system-on-a-chip using frequency-directed run-length (FDR) codes', *IEEE Transactions of Computers*, Vol 52, No. 8, pp. 1076–1088.
Chandra, A. and Chakrabarty, K. (2003b), '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, No. 3, pp. 352–363.
Chang, C.H., Lee, L.J., Tseng, W.D, Lin, R.B.(2012), '2n pattern run-length for test data compression', *IEEE Trans Comput Aided Des Integr Circuits Syst*, Vol. 31,No. 4, pp.644–648.
Correa, Roberto Sanchez, and Jean Pierre David. "Ultra-low latency communication channels for FPGA-based HPC cluster." Integration 63 (2018): 41-55.
El-Maleh, A.H. (2008a), 'Test data compression for system-on-a-chip using extended frequency-directed run-length code' *IET Computers and Digital Techniques*, Vol. 2, No. 3, pp.155–163.
El-Maleh, A.H. (2008b),' Efficient test compression technique based on block merging', *IET Computers and Digital Technology*, Vol. 2, No. 5, pp. 327–335.
Gill, S. S., Tuli, S., Xu, M., et al. (2019)."Transformative effects of IoT, Blockchain and Artificial Intelligence on cloud computing: Evolution, vision, trends and open challenges." Internet of Things (2019): Vol. 8. 100118.
Gonciari, P.T., Al-Hashimi, B.M. and Nicolici, N. (2002) 'Improving compression ratio, area overhead, and test application time for system on-a-chip test data compression/decompression. *Proceedings of IEEE Design automation and test in Europe conference and exhibition*, pp. 604–611.
Hamzaoglu, I. and Patel, J.H., (2009), 'Test set compaction algorithms for combinational circuits', *IEEE Trans. Comp.-Aided Design Integr. Circuits Syst.*,Vol. 19, No.8, pp. 957–963.
Jas, A., Ghosh, J. D., Ng, M.-E. and Touba, N.A.(2003) 'An Efficient Test Vector Compression Scheme Using Selective Huffman Coding' *IEEE Transactions on Computer-Aided Design*, Vol. 22,No. 6, pp. 797-806.
Koenemann, B.(1991) 'LFSR-Coded Test Patterns for Scan Designs'. *Proceedings of European Test Conf. VDE Verlag*, pp. 237-242.
Krishna, C. and Touba, N.A. (2002) 'Reducing test data volume using LFSR reseeding with seed compression' *Proceedings of IEEE international test conference (ITC),* pp 321–330.
Kavousianos, X., Kalligeros, E. and Nikolos, D. (2007) 'Optimal selective Huffman coding for test-data compression', *IEEE Transaction of Computers*, Vol 56, No. 8, pp. 1146–115.
Lee, K.-J., Chen, J.J. and Huang, C.H. (1998) 'Using a Single Input to Support Multiple Scan Chains', *Proceedings of Int. Conf. Computer-Aided Design*, pp. 74-78.
Mrugalski, G., Rajski, J. and Tyszer, J. (2004) 'Ring Generators—New Devices for Embedded Test Applications', *IEEE Trans. Computer-Aided Design*, Vol 23, No 9, pp. 1306-1320.
Mehta, U., Dasgupta, K. S. and Devashrayee, N. M. (2010) 'Modified selective Huffman coding for optimization of test data compression, test application time and area overhead', *Journal of Electronic Testing,* Vol. 26, No. 6, pp. 679-688.
Nourani, M. and Tehranipour, M.H. (2005), 'RL-Huffman Encoding for Test Compression and Power Reduction in Scan Applications', *ACM Transactions on Design Automation of Electronic Systems,* Vol.10, No. 1, pp. 91-115.
Sivanantham S., Padmavathy, M., Gopakumar, G., Mallick, P.S. and Perinbam, J.R.P. (2014a), 'Enhancement of test data compression with multistage encoding, *Integration, the VLSI Journal,* Vol. 47, pp. 499–509.

Sivanantham S., Mallick, P.S. and Perinbam J.R.P.(2014b), 'Low-power selective pattern compression for scan-based test applications' Computer and Electrical Engineering, Vol. 40, No. 4, pp. 1053-1063.
Tehranipoor, M., Nourani, M. and Chakrabarty, K. (2005), ''Nine-Coded Compression Technique for Testing Embedded Cores in SoCs', *IEEE Transactions On Very Large Scale Integration (VLSI) Systems*, Vol.13, No. 6, pp. 719-731.
Touba, N. A. (2006) 'Survey of test vector compression techniques', *IEEE Design and Test of Computers*, Vol 23, No. 4, pp 294–303.
Vohra, Harpreet, and Amardeep Singh. "Test data compression using hierarchical block merging technique." IET Computers & Digital Techniques 12, no. 4 (2018): 176-185.
Wang, L. T., Wu, C. W., and Wen, X. (2006). VLSI test principles and architectures: design for testability. Elsevier.
Wu, T., Liu, H. and Liu, P. J. (2013) 'Efficient Test Compression Technique for SoC Based on Block Merging and Eight Coding', *Journal of. Electronic testing*, Vol. 29, No. 6, pp. 849–859.
Wu, Angus, Peter WM Tsang, and Johnson Tang. "FPGA implementation of a near computation free image compression scheme based on adaptive decimation." Integration 36, no. 3 (2003): 121-143.
Wang, Hai, Hao Yu, and Sheldon X-D. Tan. "Fast timing analysis of clock networks considering environmental uncertainty." Integration 45, no. 4 (2012): 376-387.
Xiang, D. et al. (2007) 'Reconfigured scan forest for test application cost, test data volume and test power reduction', *IEEE Transactions on Computers*, Vol 56, No.4, pp. 557-562.
Xiang D, Zhao Y, Chakrabarty K, Fujiwara H. (2008) 'A reconfigurable scan architecture with weighted scan-enable signals for deterministic BIST'. *IEEE Trans. Computer Aided Design Integrated Circuits Systems*, Vol. 27. No. 6, pp. 999–1012.
Yuan H., Mei J., Song H. Guo, K. (2014),'Test Data Compression for System-on-a-Chip using Count Compatible Pattern Run-Length Coding', *Journal of Electron Test*, Vol. 30, No. 2, pp. 237–242.
Yuan, H., Guo, K., and Ju, Z. (2016), 'A Power Efficient Test Data Compression Method for SoC using Alternating Statistical Run-Length Coding', J*ournal of Electron Test,* Vol. 32, No.1, pp. 59–68

Author's Biography

Harpreet Vohra has a rich experience of teaching Electronics and VLSI courses. She joined Thapar Institute of Engineering & Technology, Patiala in 2006 and worked at the capacity of Assistant professor in Electronics and Communication Engineering Department. She received her Master's degree in VLSI.Design. and doctorate in test solution development for System on chip. She has guided more than 15 Masters. thesis in the areas of communication, low power VLSI design and Design for testability. Her areas of interest include machine learning and its applications in the domain of VLSI test. She is a professional member of IEEE. She has published papers in several reputed international conferences and journals.

Ashima Singh is Assistant Professor in Computer Science and Engineering Department, at Thapar Institute of Engineering & Technology, Patiala. Ashima Singh was appointed, in 2006, to Computer Science and Engineering Department as Assistant Professor. She has earned Ph.D. in Computer Engineering and holds a Master's Degree, with distinction in Computer Science. She has more than 12 years of experience in teaching and research both at UG/PG levels. She has guided 35 theses in M.E. Software Engineering and M.E. Computer Science and Engineering. She has more than 45 research publications in reputed peer reviewed journals and conferences. She is professional member IEEE and Branch Counselor, IEEE Student Chapter at Thapar Institute of Engineering & Technology. Her research interest includes machine learning and data analytics, computational bioinformatics cloud computing and IoT applications.

Sukhpal Singh Gill is a Lecturer (Assistant Professor) in Cloud Computing at School of Electronic Engineering and Computer Science (EECS), Queen Mary University of London, UK. Prior to this, Dr. Gill has held positions as a Research Associate at the School of Computing and Communications, Lancaster University, UK and also as a Postdoctoral Research Fellow at the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia. Dr. Gill was a research visitor at Monash University, University of Manitoba and Imperial College London. He was a recipient of several awards, including the Distinguished Reviewer Award from Software: Practice and Experience (Wiley), 2018, and served as the PC member for venues such as UCC, SE-CLOUD, ICCCN, ICDICT and SCES. His one review paper has been nominated and selected for the ACM 21st annual Best of Computing Notable Books and Articles as one of the notable items published in computing - 2016. He has published 50+ papers as a leading author in highly ranked journals and conferences with H-index 18+. Dr. Gill has reviewed 140+ research articles of high ranked journals and prestigious conferences. His research interests include Cloud Computing, Fog Computing, Software Engineering, Internet of Things and Big Data. For further information on Dr. Gill, please visit: www.ssgill.me.