TEST DATA COMPRESSION USING ADAPTIVE BLOCK MERGING WITH EIGHT CODING RUN-LENGTH TECHNIQUE

Thesis report submission in the partial fulfilment of the requirement for the award of the degree of

MASTERS OF TECHNOLOGY

IN

VLSI DESIGN

Submitted by

Deepinder Kaur

Roll No: 601461008

Under the Guidance of

Ms. Harpreet Vohra

Assistant Professor, ECED

DEPARTMENT OF ELECTRONICS AND COMMUNICATION

ENGINEERING

THAPAR UNIVERSITY, PATIALA (PUNJAB)-147001

JUNE-2016

CERTIFICATE

I hereby declare that the work which is being presented in the thesis entitled **"Test Data Compression using Adaptive Block Merging with Eight Coding Run Length Technique"** in the partial fulfilment of the requirement for the award of degree of M.Tech. (VLSI Design) at Electronics and Communication Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Ms. Harpreet Vohra, Assistant Professor, ECED.

The matter presented in this thesis has not been submitted in any other University/Institute.

Deepinder Kaur

Date: 15-7-2016

Deepinder Kaur

Roll. No.: 601461008

It is certified that the above statement made by the student is correct to the best of my knowledge and belief.

Ms. Harpreet Vohra

Assistant Professor

ECED, Thapar University

Counter signed by:

Dr. Sanjay Sharma

Professor & Head

ECED, Thapar University

Patiala- 147004

Dr. S. S. Bhatia

Dean of Academic Affairs

Thapar University

Patiala- 147004

ACKNOWLEDGEMENT

I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this thesis. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various people. Words are often too less to reveal one's deep regards. I acknowledge with gratitude to **Ms. Harpreet Vohra, Assistant Professor**, Electronics and Communication Engineering Department, Thapar University, Patiala, under whose guidance I had the privilege to complete this thesis. I wish to express my deep gratitude towards her for providing individual guidance and support throughout the thesis work.

I convey my sincere thanks to **HEAD OF THE DEPARTMENT, Dr. Sanjay Sharma** as well as **PG Coordinator, Dr. Amit Kumar Kohli**, Associate Professor, **Programme Coordinator, Dr. Anil Arora**, Assistant Professor, Electronics and Communication Engineering Department, entire faculty and staff of Electronics and Communication Engineering Department for their encouragement and cooperation.

My greatest thanks to all who wished me success especially my parents and my other family members and friends without whom I would not have been able to complete my thesis work.

I thank and own my deepest regards to all of them and all others who have helped me directly or indirectly.

Deepinder Kaur

Deepinder Kaur

ABSTRACT

With the increase in the complexity, the amount of test data on chip is increasing rapidly. This increases the possibility of manufacturing imperfections at deep sub-micron level. Due to increase in test data volume, the amount of memory required to store huge test set data is increased. This puts constraints on ATE memory to store test data. One of the approaches to address this issue is the use of BIST, but for this, (intellectual property) IP core should be BIST ready. This rise in test data volume leads to an increase in test application time. So, there is a requirement to reduce the test data. If the test data is compacted then the ATE memory required to store the test patterns decreases. This will reduce the requirement of external hardware of memory to support ATE.

Thesis report consists of the study of several test data compression schemes. Various Code based techniques have been studied, such as, Run-length based (Golomb code, FDR, EFDR and AFDR), Statistical code based (Selective Huffman and Optimal Huffman) and Dictionary based (Bitmask and dictionary selection) compression schemes. A new Modified BM-8C test compaction scheme has been proposed based on the block merging and eight coded technique. ABMEC is compared with the existing techniques based on the compression ratio and test application time. When applied to the various benchmark circuits, ABM8C technique improves the compression ratio and reduces the test application time. This decreases the memory requirement of ATE to store the test patterns. It also reduces the dependency of cores to the BIST ready.

Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER-1

INTRODUCTION

With the development in technology, intricacy of VLSI designs is growing day by day. Due to improvement in integration technologies, millions of transistors are made-up on a particular Integrated Circuit (IC) or chip. As per Moore's law, the integration level of microchips doubles in every 18 to 24 months, due to decline in feature size. The reduction in feature size results in increasing the chance of defects during manufacturing process. A faulty IC results from a solo faulty transistor or wire. Therefore, testing of designs becomes vital to have fault free yields.

1.1 Need for Testing

Due to rise in integration level, the size of the design space enormously increases. It is not feasible to confirm accuracy of the design under all probable circumstances. So, methods are required that can validate accuracy of design without having in-depth input-output arrangements and the design meet all the input conditions; this method is called formal verification. Testing is a more intricate procedure than verification. More time is required for testing than verification as merely one design is to be verified while all chips are required to be tested. Testing is ensures the elimination of manufacturing faults. Improper circuit performance leads to failure of device. Testing is vital as it has influence on the cost.

With surge in device intricacy, testing complication rises. Size of test data needed to test the circuits rises with rise in concentration of integrated circuits. Huge amount of test patterns are required to expand error handling. This additionally raises cost of testing. Two concerns related to testing devices for failures are addressed as follows:

i. Huge test set size which influences test loading necessities.
ii. Testing time of device

1.2 Testing of VLSI circuits

Generally, test data is moved in dual phases. In first phase, test vectors are transferred from workstation to ATE. In second phase, they are moved between ATE and CUT. The compression techniques are applied in phase 1 which can probably be complex. Decompression is done in phase 2, which needs to be simple. Using test

compaction techniques the time required to download test patterns from workstation to ATE is reduced. The two phases of test data transfer are shown in Figure 1.1

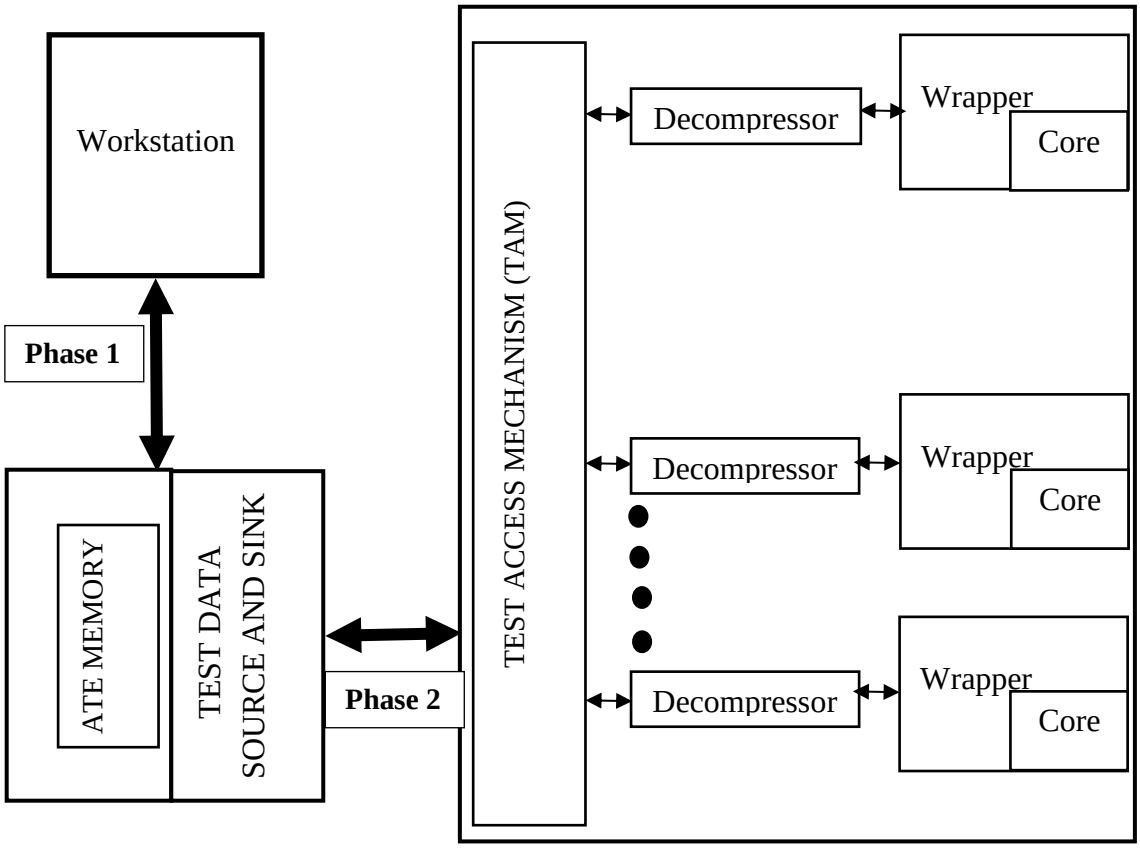

Figure 1.1 Test Data Transfer and CUT Testing Mechanism [19]

Basic structure of testing consists of three components: Automatic Test Equipment (ATE), Circuit-Under-Test (CUT) and ATE memory to store test patterns and test responses attained by ATPG tool as shown in Figure 1.2. Yield depends on technology, chip area and layout. Hence, testing is used to expand yield and to assure error-free system operation. However, writing test patterns physically is very costly and time consuming. Therefore, automation is required.

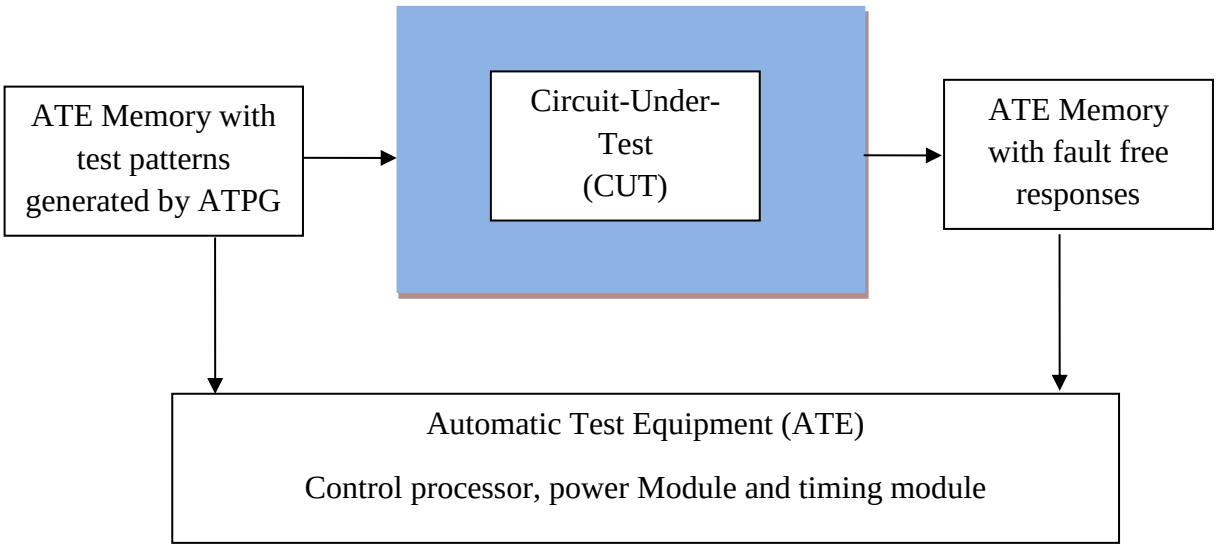

Figure 1.2Testing of a circuit [13]

1.3 Time Restriction in ATE

Test vectors and their test responses are kept on an exterior tester-that is, Automatic Test Equipment (ATE) in exterior testing. ATPG is a process that includes generation of input patterns that can determine existence and lack of error(s) at some position(s) in a circuit. Test patterns are spontaneously created by Automated Test Pattern Generator tool. ATPG is essential as whole functional test is unfeasible. Test arrangements created by ATE permits ATE to discriminate between the correct and defective performance of circuits caused by faults.

With the growth of CMOS process, core based design practice is extensively used to design a system-on-chip (SOC). Though, there are many test challenges that need to be addressed such as huge test data, test application time, test power etc. The volume of test data increases quickly in more complex designs.

Test data sets are usually generated and stored on workstations. These test data sets are transferred from workstation to Automatic Test Equipment. Due to large test sets for Application Specific Integrated Circuit (ASCIs), the time required to transfer the test set data from workstation to ATE is significant. Test set data stored at workstation is transferred to the user interface workstations through a network. This transfer takes from several tens of minutes to several hours. Then a high dedicated speed bus is used to transfer test data from user interface workstations of ATE to main pattern memory. This usually takes several minutes. As the complexity of IC

designs is increasing rapidly, therefore, new testers with more memory, channels and higher speed of operation is required. This increases the cost further. Test data storage in growing rapidly in ATE. Test data bandwidth requirements between the tester and the chip are also increasing.

Tests on SOC comprised of various cores are applied by ATE. Test patterns are decompressed by test pattern decoder which is implemented on chip. The test patterns are delivered according to test schedule in Test Access Mechanism (TAM). Design for Test circuitry will reduce the load on ATE. Hardware is added onto the chip, which is working in conjunction with external tester.

There is significant rise in operating frequency (i.e. speed), memory size and channel capacity. Thus, Due to limited channel capacity ATE requires modification to support large volume of test data. Besides, test application time directly depends on the volume of test data.

The first solution is Built-in- Self-Test (BIST), in which redundant circuitry are placed on chip to remove faults. BIST requires considerable redesign effort and covers less fault area. It also introduces performance penalties. The second solution to reduce memory and channel requirements of ATE is to use test data compression/decompression techniques. For test data compression, existing embedded cores are reused and minor modifications are required during system test application.

1.4 ATE vs BIST for System-on-Chip Test

Testing has become an important part of System-on-chip. Automatic Test Equipment is an external testing in which all the test vectors and test responses are stored externally. But sometimes the external storage is not possible for test vectors in a circuit. For such circuits, sometimes redundant circuitry is kept on chip which replaces the faulty parts. Testing a circuit every time before they start up, is called Built-In-Self-Test (BIST). The test application circuit and test analysis circuit is already in the chip. It provides the capability of circuit to test itself. Therefore, BIST plays important role in component level testing, where it is difficult to access embedded virtual components. BIST solution will be economically and technically lower as compared to ATE. Thus, BIST plays important role in devices that need to perform diagnostic function upon themselves in the field. However, BIST has its drawbacks, including, fault coverage, complicated timing closure, tool complexity,

and power consumption. It suffers from bus conflict and random resistant faults during test application which leads to inadequate fault coverage. This is because the circuitry in BIST will be more complex. Fault coverage is less in BIST as compared to ATE. Very long test sequences are required. This will lead to additional area and routing overhead.

ATE plays important role in system level testing. System level testing is an important part of testing system on chip device. In deep sub-micron level implementation, the data rates are high. This can only be tested using external ATE, in system level testing. ATE provides traceable and accurate measurements comparing to BIST. They can be reused from application to application. ATE is cost efficient for high volume testing. This is because it is easier and cheaper to put a circuitry inside a single tester than inside every device.

Both approaches will have their own advantages and disadvantages. Thus right combination of ATE and BIST is required to grow the system on chip market. Strengths and weaknesses of each approach need to be recognized to design a complete test strategy that will produce a high quality device at the lowest overall cost.

1.5 Test Data Compression

Due to the advancement in VLSI technology, the numbers of transistors integrated on a single chip are increasing. The increase in density of integrated circuits in system-on-chip (SOC) designs has led drastic increase in test data volume required to test the circuits. The increase in test data volume increase the test cost rapidly. The contradiction between test quality and test efficiency sharpens. Due to this increase in test data size, the memory requirements are increased. Test application time also depends on size of test data volume. So with increase in test data volume, test application time also increases. One way to reduce test data volume is by test data compression. With the help of test data compression, this problem is reduced by reducing the test data volume without affecting the overall performance of the system. Test data compression is very effective means to save network bandwidth and storage space. Using these techniques, the available memory can be used as efficiently as possible. Test data compression is an encoding technique in which some rules are predefined according to which data size is reduced. Several test data

compression schemes have been proposed to decrease the test data volume. A large number of test data compression schemes have been based on encoding or on detection of repetitive string.

Testing in test data compression implies sending of compressed test data from the ATE to the on chip decoder, decompressing of test set data on chip and transferring the decompressed test data to the Circuit-Under-Test (CUT) [27].The objective of test data compression is to reduce the number of bits needed to be stored in ATE memory. Test data compression plays a very important role in reducing the cost of testing very large and complex designs by reducing test application time. This also impacts the storage requirements of the test set patterns in the testing hardware. To address this issue, several test data compression techniques have been proposed which reduces the size of test pattern. The test data compression methodology is shown in Figure 1.3

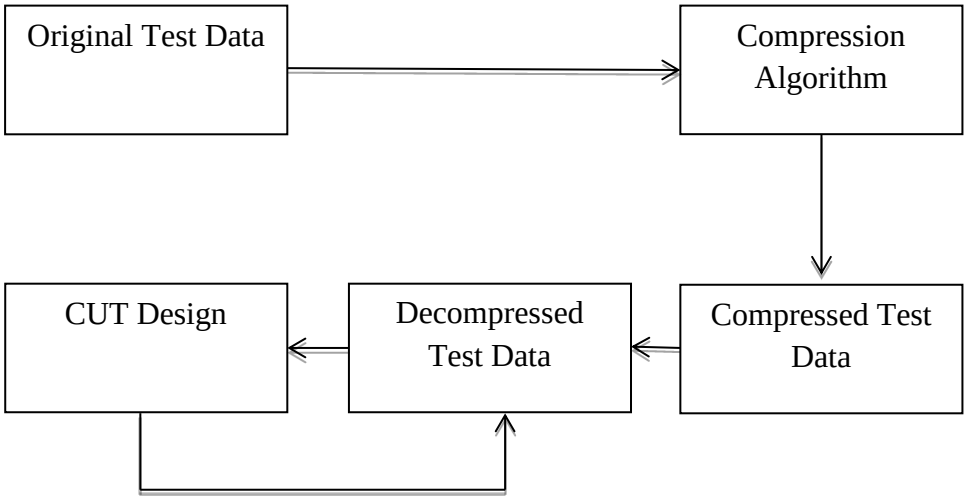

Figure 1.3 Test Data Compression [27]

1.6 Parameters for Test Data Compression

The following parameters characterized the test data compression:

i. Compression Ratio recognizes the performance of compression scheme and memory requirements on the ATE.
ii. Test Application Time, the time required to transfer and decode the compressed test set.
iii. Power Dissipation, during the transitions between successive test vectors.

These parameters are explained as follows:

i. Compression Ratio

In test data compression, several methods those are using diverse types of patterns and diverse lengths exploit diverse features of the test set data. Thus, the test data compression ratio is influenced by the recording and reordering algorithm, the type of input patterns, and the extent of the pattern and by the compaction algorithm.

ii. Test Application Time (TAT)

TAT is influenced by the compression ratio, the length of the pattern and then by the type of on chip decoder. Minimum TAT is given by the size of the compressed test set in the ATE clock cycles [10]. Minimum TAT is obtained when the on chip decoder can compress the currently compressed bit before the next one is sent by the ATE. ATE frequency and the frequency at which the on chip decoder works have to meet certain conditions. One of the primary goals of any SOC testing is to reduce test application time (TAT). Test data compression decreases testing time and allows the use of low cost ATE.

iii. Power Dissipation

A circuit consumes much higher power in test mode as compared to the normal mode. This is because the correlation between successive test set patterns is low. Therefore, the switching activity of all the nodes is several times higher in test mode [3]. This reduces the circuit reliability. Thus, there is a need of low power testing.

Considering the CMOS integrated circuits, power dissipation is caused by static power and dynamic power. Static power is due to the leakage current. Dynamic power is due to the switching of the circuit. Dynamic power consumption during testing depends on number of transitions that occur during switching activity of

scan-in and scan-out operations. Thus, dynamic power consumption and hence total power consumption can be reduced by reducing the switching activity.

CHAPTER-2
LITERATURE REVIEW

When testing is done using stored patterns, the test sets are moved from ATE to SOC. With large size of these test patterns the memory capacity of the tester exceeds. This results in huge test data volume and time during the testing. This can be done using Built-In-Self-Test (BIST) or test data compression.

Memory testing is done by using BIST. But it is not used for logic testing. Test application time of BIST is large. Another drawback of BIST is coverage of faults. The complexity of BIST is also more due to the presence of test application circuit on the chip itself. Hence, test data compaction is better substitute to have more fault coverage with less complexity. It reduces volume of large test sets with less additional area overhead. This decreases the time of testing. Test data compression methods belong to the following types:

i. *Broadcast Scan Based Scheme*: Same value is broadcasted to various chains. Multiple chains are controlled by a single chain [1]. It uses the fact that different test cubes can be used to detect the faults. The enacted constraints and decompressor are easily combined using broadcast scanning. Dependent inputs are given to ATPG which will produce only test patterns those can be encoded easily [27]. However, it produces restricted range of test cubes to be tested as compared to linear decompression scheme.

ii. *Linear Decompression Based Scheme*: Here, only linear operations are used to decompress the test data. The enacted constraints and XORs cannot be easily combined as this can degrade the performance of ATPG. Test patterns those can be encoded are required to be checked by solving linear equations. Thus, it is less efficient as compared to broadcast scan scheme to have test patterns that can be encoded.

iii. *Code Based Schemes*: Data compression codes are used to encode the test patterns. This involves dividing the original data into symbols. Each symbol is then substituted by a code word to form a compressed data [23]. For

decompression, decoder converts each code word in the compacted data back to its corresponding symbol.

2.1 Code Based Test Data Compression Techniques

In test data compaction, extra on chip hardware is added before and after the scan chains. Test pattern from the tester is decompressed using this extra hardware. Because of this, test data is stored on the tester in compressed form. In this method, the test set which is produced by ATPG is transferred to the CUT.

In code based schemes, the original data is segregated into symbols. Each symbol is then substituted with a code word to form the compressed data. For decompression, each code word is converted back into its equivalent symbol.

Some of the code based schemes are as follows:

2.1.1 Run-Length –Based Codes

Test vectors are highly compressible because only 1% to 5% of bits are specified (care). The rest are don't care which can take on any value with no impact on the fault coverage. ATPG does not arbitrarily fill don't care bits. Along with high fraction of don't cares, test cubes are also extremely interrelated because faults are structurally related to the circuit. Both these aspects are exploited to achieve the improved compression ratio. Don't care bits need to be filled to get extended runs. This increases the compaction. These don't care bits ("X") can be filled with either ones or zeros. The filling of don't care depends on the various types of codes.

Difference vector: Firstly ordering is done on the basis of hamming distance. Let the ordered test be defined as

$$T_{diff} = \{d_1; d_2.....;d_n\}$$

$$T_{diff} = \{t_1; t_1 \text{ xor } t_2.....;t_{n-1} \text{ xort}_n\}$$

It will increase test data compression by increasing the number of zeros.

2.1.1.1 Simple Run-Length-Based Code

In Simple Run-length based code scheme, a variable number of bits are encoded by a fixed number of bits. Jas and Touba used this scheme to encode runs of 0s using fixed length code words. Cyclical scan architecture is used to allow the application of difference vectors. The difference vectors between the vectors between test cubes

t_1 and t_2 is equal to t_1 XOR t_2. The numbers of 0s in the difference vectors are maximized by the careful ordering of test cubes [1]. Thus, the effectiveness of run-length coding is improved. 3-bits run length code is shown in following Table I.

Table I: 3-Bits Run Length Code [1]

Code word	Symbol of 3 bit run-length code	Symbol of modified 3 bit run-length code
000	1	10
001	01	11
010	001	01
011	0001	001
----	----	----
111	0000000	000000

2.1.1.2 Golomb Code

It is a variable to variable length code. Variable to variable means that if the runs of zeros varies then the code word also varies for it. First step is to determine the golomb parameter *m* (*m* is any power of 2). The run of zeros in are mapped to the group of size *m*. The prefix of group A_k will have (k-1) ones followed by a 0. Testing of multiple cores in parallel is allowed using a single ATE I/O channel. It increases the capacity of ATE I/O channel [2]. The Golomb coding is shown in Table II.

Table II: Golomb Coding for m=4 [2]

Group	Run length	Group Prefix	Tail	Code word
A_1	0	0	00	000
	1		01	001
	2		10	010

	3		11	011
A_2	4	10	00	1000
	5		01	1001
	6		10	1010
	7		11	1011
A_3	8	110	00	11000
	9		01	11001
	10		10	11010
	11		11	11011

For test pattern decompression, no separate cyclical scan register (CSR) is required. The additional logic for the SOC channel is small which is easy to implement. Synchronization mechanism is required between the tester and the chip. It allows efficient encoding of longer runs because of the use of variable length code words.

2.1.1.3 Frequency Directed Run length (FDR) codes

It is a variable-to-variable-length compression codes those are designed using the runs of 0s in test set pattern. This compression scheme is based on the fact that frequency of runs of 0s is high with run-length less than 20. Hence, mapping of runs of 0s with shorter run-length to shorter code words is more efficient [7]

Test Resource Partitioning (TRP) technique is used that reduces test application time, test data volume and scan power. It is based on alternative run length codes for test data compression

Table III: Frequency Directed Run-length Code [7]

Group	Run-Length	Group Prefix	Tail	Code word
1	0	0	0	00
	1		1	01
2	2	10	00	1000

	3 4 5		01 10 11	1001 1010 1011
3	6 7 8 ---- 12 13	110	000 001 010 ---- 110 111	110000 110001 110010 ---- 110110 110111

For any code word, the tail and prefix are of equal length. As we move from one group to another, the code word size increases by two bits. This scheme is very efficient when the compressing data contains few 1s and long runs of 0s. Table III shows the Frequency-directed run-length code.

2.1.1.4 Extended Frequency Directed Run length code (EFDR)

Higher test data compression can be achieved using both runs of 1's and 0's to encode. In Extended frequency directed run length code and extra bit is added to the beginning of code word to indicate the type of run. In this code, the run-length of size 0 types is not there. This is because both runs of 1s and 0s are encoded.

Test Data Compression ratio is significantly increased for all considered test cases. EFDR code is better than previous FDR codes. The compression ratio improved from 19.36% to 80.8%. Extended FDR code is shown in Table IV.

Table IV: Extended FDR [4]

Group	Run-length	Group Prefix	Tail	Code word Runs of 0s	Code word Runs of 1s
1	1	0	0	000	100

	2		1	001	101
2	3	10	00	01000	11000
	4		01	01001	11001
	5		10	01010	11010
	6		11	01011	11011
3	7	110	00	0110000	1110000
	8		001	0110001	1110001
	9		010	0110010	1110010
	-------		-------	-------	-------

2.1.1.5 Alternate Frequency Directed Run-length Code (AFDR)

Generally, the test set is composed of alternating runs of ones and runs of zeros. Alternate frequency directed run-length code is a variable to variable length code. It consists of two parts group prefix and tail. An additional parameter, alternating binary variable *a*, is associated with this code. If $a = 0$, run length is treated as runs of 0s. The default value of *a* is 0, that is, the input data stream starts with a run of 0s. Table V. shows the Alternating frequency directed run length coding scheme.

Table V: Alternating Frequency Directed Run Length Code [5]

Group	Run length for a = 0/1	Group Prefix	Tail	Code word
1	0	0	0	00
	1		1	01
2	2	10	00	1000
	3		01	1001
	4		10	1010
	5		11	1011
3	6	110	000	110000
	7		001	110001
	8		010	110010
	----		----	----
	13		111	110111

2.1.1.6 Shifted Alternate Frequency Directed Run-length Code

This scheme is modified version of AFDR coding. In Shifted Alternating FDR there is no run length of zero size. So, code word of run length size 0 is not required. This code word is assigned to run length size 1. Due to this, each code word is shifted to one position higher. Each symbol is made up of only 1s and only 0s. The type of first run, i.e. 0 or 1, is indicated by the first bit of encoded data. The run length of alternating runs will be indicated by each code word. Higher test data compression is achieved in Shifted alternating frequency directed run-length code as compared to Alternating FDR coding scheme. Shifted alternating FDR scheme is shown in Table VI.

Table VI: Shifted Alternating Frequency Directed Run Length Code [16]

Group	Run-Length	Prefix	Tail	Code word
1	1	0	0	00
	2		1	01
2	3	10	00	1000
	4		01	1001
	5		10	1010
	6		11	1011
3	7	110	000	110000
	8		001	11001
	9		010	11010
	10		011	11011
	11		100	110100
	----		-------	-------

2.1.1.7 Modified Frequency Directed Run-length Code(MFDR)

Modified FDR scheme is based on the probability of 0s and FDR. In this scheme, further modification is done to the groups of FDR in such a way that it gives better compression ratio as compared to FDR. It is the class of variable to variable length prefix code. Table VII shows the coding scheme for Modified FDR.

Table VII: Modified Frequency Directed Run Length Code [5]

Group	Run length	Group Prefix	Tail	Code word
A_1	0	01	00	0100
	1		01	0101
	2		10	0110
	3		11	0111

A_2	4 5 6 7	10	00 01 10 11	1000 1001 1010 1011
A_3	8 9 10 11	001	00 01 10 11	00100 00101 00110 00111
A_4	12 ----	110	000 ----	110000 ----

2.1.1.8 2^n Pattern Run-Length for Test Data Compression

2^n Pattern Run Length (PRL) coding encodes the runs of compatible and inversely compatible patterns. In this technique, the test data is first divided into permanent length sections of L-bit. The power of L is 2. $2^{|n|}$ patterns are then compressed. Here, n is a signed integer. Encoding of test data is done based on whether it is performed inside a single test data segments or in multiple test data segments. The decompressor is easy to implement and simple. The test application time is saved considerably.

2.1.1.9 Selective Compression using Variable-to-Fixed and Fixed-to-Variable Codes

Variable-to-Fixed codes and Fixed-to-Variable codes are used for low power test data compression. This scheme is based on number of bits for representing the original test vector and the number of transitions per second. It achieves high compression ratio and low power consumption. It helps in attaining compact test data volume and small power than the previous methods for cases where number of stated bits in the test set is small comparatively.

2.1.1.10 Double Hamming Distance Reordering with Mixed RL-Huffman based Compression Scheme

Multi Code Compression (MCC) scheme is based on dissimilar run length coding approaches. The dissimilar run length coding methods used are Alternate run length, Frequency Directed Run length (FDR), Extended Frequency Directed Run length (EFDR) and IFDR code. Firstly, double hamming distance based reordering method is applied to the test set vectors. This scheme uses all the useful properties of all the used run length based compaction methods into a single scheme. This results in higher compaction ratio. Finally Huffman encoding technique is applied to increase the test data compression ratio. The area overhead of decompression architecture is decreased.

2.1.1.11 Hamming Distance based Reordering and Column wise Bit Stuffing with Difference Vector

It is method for "don't care" bits filling centered on nature of runs to forecast the highest compaction based on entropy. The maximum test data compaction is acquired when X filling is done using runs of zeros followed by one as well as runs of ones followed by zero, i.e. Extended Frequency Directed Run length code. When don't care bits are filled to create the long runs of zeros as well as ones, the average test power and peak power is least.

Hamming Distance Based Reordering and Column-wise Bit Stuffing with Difference-Vector (HDR-CBS-DV) technique can be used with any run length based code method for having improved compaction ratio. Matching fault free outputs that are kept in ATE must also be reordered. They have applied four methods in this scheme: Selection of first vector, Hamming Distance based Reordering, Column wise Bit Stuffing and Difference Vector. Test data compression ratio can be enhanced considerably through this scheme. The method does not comprise any additional silicon area overhead compared to the base run length code with which it is used. It needs an extra XOR gate and feedback path only.

2.1.1.12 Block Merging Test Data Compression Technique

One of the active test data compression techniques is block merging. Block Merging is one of the Code based schemes. It exploits the correlation in specified bits. It can be used on any test set pattern. Hence, it is independent of the test data. Hence, the decompression circuits need not to be modified if the original test data is changed.

Hence, the test data compression scheme will be non-test-independent if the decompression circuit needs to be modified if the original test data is changed.

In test data compaction based on block merging various successive blocks of the test data can be combined together. The blocks combined and the numbers of blocks merged are stored. In this technique, three cases are considered, i.e. Blocks containing both ones and zeros, blocks filled with all ones, or filled with all zeros. Simple circuitry is used for test data decompression which replicated the blocks combined the required number of times. This scheme achieves significantly more compression ratio as compared to the existing schemes.

This technique is based on the fact that the two successive blocks of the test data set can be combined together. The combined block and the number of combined blocks are stored in the encoded test data. In cases, where the merged block can be occupied by all zeros or all ones, it is encrypted by 2 bits. In all other cases, the content is stored. To have maximum cases of all zeros and all ones, the flexibility of don't care bits is used. This is for the reason that don't care bite (X) can be taken as either 0 or 1. Test data decompression is done on chip by means of simple circuitry.

Table VIII: Block Merging Encoding Scheme [12]

Group	**Types**	**Code word**
B=1	No Merging	**0**+b bits
B=2	1s and 0s	**10** + b bits
	Fill with 1s	**10**11
	Fill with 0s	**10**10
B=3 to 6	1s and 0s	**110** + 2 bits + 0 + b bits
	Fill with 1s	**110** + 2 bits + 11
	Fill with 0s	**110** + 2 bits + 10
B=7 to 14	1s and 0s	**1110** + 3 bits +0 + b bits
	Fill with 1s	**1110** + 3 bits + 11
	Fill with 0s	**1110** + 3 bits + 10

B=15 to 30	1s and 0s Fill with 1s Fill with 0s	**11110** + 4 bit + 0 + b bits **11110** + 4 bit + 11 **11110** + 4 bit + 10
B=31 to 62	1s and 0s Fill with 1s Fill with 0s	**11111** + 5 bit + 0 + b bits **11111** + 5 bit + 11 **11111** + 5 bit + 10

The Block Merging (BM) compression method is based on the splitting of test set into block size of b bits and then combining consecutive compatible blocks. Two consecutive blocks are compatible if they have same value or either one of them is don't care (X) bit. The method merges all the successive compatible blocks into one combined block. This merged block is stored along with the count which indicates the amount of blocks merged. The merged blocks are encoded in two dissimilar ways. First, if the block can be occupied by only one value, i.e. either 0 or 1, then it is encoded. Otherwise, if it contains mutual values, then the content is stored. In case of block merging, the number of bits those were representing the count of merged blocks is reduced. This is done by grouping the merged block counts into six groups. The group is represented by the prefix. The frequency of count of merged blocks is highest at the low values. This frequency reduces with growing values. In Table VIII, different encoding schemes are shown for six groups. Block size is assumed to be between 4 and 10 bits. It is represented by B. When two consecutive blocks cannot be merged, it is represented by B=1, which is the first group. When two blocks can be merged it is represented by the second group (B=2). To represent B=2 group, prefix 10 is used. If the next bit is 0, then it specifies that the block contain both 0s and 1s. Hence, its contents are stored using b-bits. However, 11 is used to specify if the block is occupied with all 1s. The number of blocks combined is symbolized by the next 2 bits. The next bit specifies whether the block contains both zeros (10) and ones (11), or it is filled with all zeros and all ones (0).

Table IX: Block Merging Compression Results [12]

Circuit	B=3	B=4	B=5	B=6	B=7	B=8	B=9	B=10	B=11
S13207	81.78	83.19	84.70	84.24	83.45	84.69	84.89	84.74	83.97
S15850	65.97	68.05	68.82	68.60	69.49	68.76	68.48	67.94	66.62
S35932	77.20	77.19	78.35	77.95	76.96	74.47	73.81	73.39	72.77
S38417	59.08	59.39	58.87	58.31	58.21	56.90	57.43	54.84	54.68
S5378	50.49	52.27	53.50	54.98	53.51	52.99	53.25	52.47	52.75
S9234	46.32	48.99	51.19	49.31	50.75	49.55	49.25	48.24	46.45

2.1.1.13 Block Merging with Eight Coding

It is a test data compression technique which is based on compression program that uses exactly eight code words. It provides noteworthy reduction in test application time and test data volume. This scheme encodes each combined block and the number of succeeding compatible blocks for each merged block to a code word as per eight properties. The test data volume is further reduced by including the properties of inverse compatibility between the encoded merged block and the previous block. It also uses properties including compatibility and inverse compatibility between two halves of the encoded merged block and the compatibility between each half and all zeros or ones in other half. These properties are presented to encode the merged block.

In Table X. Merged blocks are categorized into five groups, A0-A4. This is done on the basis of number of consecutive compatible blocks for each merged block count and a unique prefix is utilized to encode each group. For example, the merged block belongs to first group A0 if it is not merged with any consecutive blocks (MGC=1). If the count of merged block is between 8 and 15 ($8 \leq MGC \leq 15$), it belongs to fourth group, A3.

Table X: Block Merging with 8C Encoding Scheme [23]

Group	MGC	Types	Code word
A0	1	inv_c	0+010
		half_c	0+11+b/2
		half_i	0+10+b/2
		0U	0+01100+b/2
		1U	0+01101+b/2
		U0	0+01110+b/2
		U1	0+01111+b/2
		UU	0+00+b
A1	2-3	inv_c	10+1 bit+010
		half_c	10+1 bit+11+b/2
		half_i	10+1 bit+10+b/2
		0U	10+1 bit+01100+b/2
		1U	10+1 bit+01101+b/2
		U0	10+1 bit+01110+b/2
		U1	10+1 bit+01111+b/2
		UU	10+1 bit+00+b
A2	4-7	inv_c	110+2 bit+010
		half_c	110+2 bit+11+b/2
		half_i	110+2 bit+10+b/2
		0U	110+2 bit+01100+b/2
		1U	110+2 bit+01101+b/2
		U0	110+2 bit+01110+b/2
		U1	110+2 bit+01111+b/2
		UU	110+2 bit+00+b

A3	8-15	inv_c	1110+3 bit+010
		half_c	1110+3 bit+11+b/2
		half_i	1110+3 bit+10+b/2
		0U	1110+3 bit+01100+b/2
		1U	1110+3 bit+01101+b/2
		U0	1110+3 bit+01110+b/2
		U1	1110+3 bit+01111+b/2
		UU	1110+3 bit+00+b
A4	16-31	inv_c	1111+4 bit+010
		half_c	1111+4 bit+11+b/2
		half_i	1111+4 bit+10+b/2
		0U	1111+4 bit+01100+b/2
		1U	1111+4 bit+01101+b/2
		U0	1111+4 bit+01110+b/2
		U1	1111+4 bit+01111+b/2
		UU	1111+4 bit+00+b

2.1.2 Statistical Codes

In Statistical coding original data is segregated into n-bit symbols. Code words of variable length code are then assigned depending on rate of occurrence of each symbol. Statistical compression techniques are consisting of two phases. First phase is a modeling phase, in which statistical properties of input sequences are computed and a model is built. In second phase, model is used to compress the input sequence. This is a coding phase.

Statistical codes are fixed to variable codes. They denote data block of static length using variable length code words. Compression is attained by encoding the most repeatedly occurring blocks with small code words and less repeatedly occurring ones with large code words. This reduces the average length of the code word. Thus,

effectiveness of statistical codes is based on the frequency of occurrence of all diverse static length blocks in a set of data.

2.1.2.1 Huffman Coding

Huffman is a statistical code that is based on variable length encoding. It offers the shortest average code word length. Thus, it provides the best compression. A significant property of Huffman code is that no prefix is being used. Consequently, the decoding procedure is simplified. The decoder can identify the end of code word promptly without any look ahead. In the compressed test set, for distinguishing between a code word and an encoded data block, an extra bit is used before either of them. Each non-encoded data block is led by the '0' bit, while each code word is led by the '1' bit. The extent of compression that can be attained with statistical coding depends on the rate of occurrence of dissimilar code words.

A binary tree is made to create Huffman coding. It starts with the leaves and travels to the root. A leaf node is created for every individual block b_i. p_i is the rate of occurrence of block b_i. Weight equivalent to p_i is allocated to each leaf node. The couple of nodes with least weights are designated first. Parent node is created with weight equivalent to addition of weights of these two nodes. This is continued iteratively till the single node is left unselected which is a root. Each node can be nominated only once. Then all nodes are visited once, beginning from the root. Logic "1" ("0") value is allocated to every child edge. The code word to each block is the order of logic values of edges from root to leaf node while going through to path of that node.

The main difficulty of Huffman coding is the high hardware overhead of the decompressors. If the block size is augmented linearly, then there will be an exponential rise in different block volume. Thus, number of code words that need to be decoded also rises exponentially. The Huffman Algorithm is shown in Figure 2.1

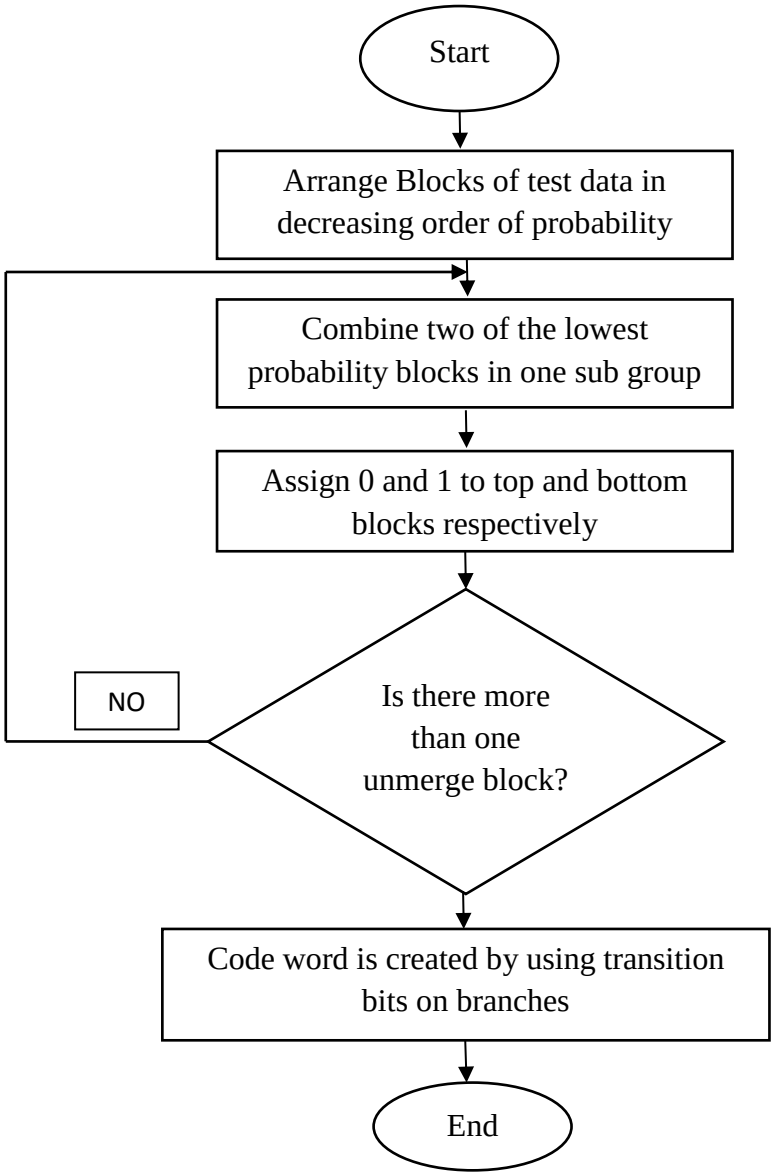

Figure 2.1 The Flowchart of Huffman Algorithm [29]

2.1.2.2 Selective Huffman Coding

Selective Huffman coding technique is not optimal as an additional bit is used beforehand a code word and non-encoded data block to differentiate among them. It reduces the decoder size considerably by losing the compression ratio slightly. Thus, it has low hardware overhead. Testing of complex SOC's comprises of several cores; there is rise in test data bandwidth among the tester and the SOC. In this scheme the test vectors of SOC are stored in tester memory in the compressed form. A revised Huffman code is used that fulfills certain properties. These properties confirm to decode the code words by pipelined decoder that requires smaller area relatively.

This technique selects a “simple-to-decode” statistical code for specific test pattern. The area overhead of an optimum Huffman code is less comparatively.

2.1.2.3 Optimal Selective Huffman Coding

The extra bit in Selective Huffman scheme extends all the data in compacted test set regardless of the rate of frequency. Thus, this technique is not optimal. Additional Huffman code word is used in front of only the non-encoded data blocks to have Selective Huffman coding which is optimal. The non-encoded blocks of data are generally a small portion of the test set after compaction. Thus, in compressed test set, the improvement from restricting the code words that appear repeatedly will overbalance the loss of using an entire code word instead of a single bit, in front of the non-encoded data blocks. In this approach, $m + 1$ Huffman code words are used. Here, m is for encoding the m most repeatedly occurring different blocks and 1 for representing the non-encoded blocks. The rate of frequency of the later code word is equal to the sum of the rate of frequencies of all the encoded different blocks. This permits the code words those are occurring repeatedly to get rid of overhead due to extra bit. Hence, it offers improved compression and has low area overhead.

2.1.2.4 Multilevel Huffman Coding for Test Data Compression

Multilevel Huffman Coding is compaction of dissimilar kinds of information using similar code word. Separate Huffman code is used for non-encoded data blocks instead of addition of extra bit. This method is appropriate for cores of unfamiliar structure. In this technique, majority of “X” values of test sets arbitrarily. This is done by using small LFSR. The compression effectiveness is enhanced considerably in this scheme. Decoder style is very simple for decoding the compacted data on chip. This is because main portion of the decompressor can be common among dissimilar cores. This scheme also increases the chance of the recognition of non-modeled faults. It offers the capability of using the tradeoff between the area overhead and compression ratio.

2.1.2.5 Variable length Input Huffman Code (VLHC)

The group prefix as well as tail needs to be recognized by the decoder to decompress an FDR code. In FDR code, the code is not dependent on group size. So, decoder has to identify the length of prefix for decoding the tail. So, decoder of FDR code is more intricate and with higher area overhead. Huffman and FDR are combined in

Variable Length Input Huffman coding scheme. In this scheme, patterns of fixed length are not used.

2.1.2.6 Dictionary Based Codes

Dictionary coding selects strings of symbols to create a dictionary. These strings of symbols can be either static or dynamic. These symbols are then encoded into tokens of same length using dictionary. In this scheme, pre-computed data is divided into n-bit symbols. Dictionary is used to store each exclusive symbol. It takes the benefit of number of frequently occurring sequences. Test data is compacted by encoding each n-bit using a b-bit code word that matches to the index of symbol in the dictionary. b is less than n when all probable symbols do not occur in test pattern. These codes provide twofold benefit of better compression ratio and fast decompression mechanism.

2.1.2.7 Efficient Bitmask and Dictionary Selection Method

The efficiency of Dictionary based techniques depends on efficient bitmask selection method creates maximum matching patterns. More matching patterns will be created using more number of bit variations. Due to this, there is a probability that the test data after compaction becomes greater than the size of pre-computed test data. This problem is addressed using Bitmask based test compaction by generating more patterns that are matched. Bitmask offers noteworthy improvement in compression ratio. It does not lead to any added decompression disadvantage. The major drawback of this technique is that it is not designed for test patterns with don't care bits ("X"). Thus, a dictionary needs to be selected that will provide the most optimized result.

2.1.2.8 Dictionary with Selective Entries and Fixed Length Indices

It is based on the small number of ATE channels to deliver test data in compacted form tester to chip. It removes the requirement for synchronization between SOC and ATE. Fixed length indices are used by dictionary. Its entries are selected to use dictionary effectively. It does not need gate-level circuit model for test generation. It does not need any multiple clock cycles to determine test data after decompression when the last bit of compacted set is moved from ATE to chip.

CHAPTER-3

PROPOSED METHODOLOGY

High compression can be obtained by efficiently using don't care bits. There are generally huge number of don't care bits in original test data set T_D. Former methods substitute don't cares bits present in original test set, T_D with 0 or 1 to create a compressed test set, T_E. T_D is created by on-chip decoder without don't care. Hence, different techniques are required to use a portion of don't care more efficiently. Here, new compaction technique is presented based on compression code with block merging and modified eight code words.

3.1 Analysis of BM-8C Technique

Two successive combined blocks are encoded according to the compatibility and inverse compatibility of consecutive blocks. Two bits are *compatible* if they are equal or any one of them is a don't care (X) bit. Dual bits are *inverse compatible* if they are contrary or either one of them is don't care (X). Consequently, two consecutive blocks are defined to be compatible (or inverse compatible) if every parallel bit is compatible (or inverse compatible). If several succeeding blocks are compatible to each other, they are succeeding compatible blocks and can be merged into one *merged block* which is a representative block.

Eight-Coded compression technique is based on following eight properties of the combined blocks those are attained after block merging. After combining two succeeding compatible blocks of the original test data, the combined blocks will belong to one of the following categories:

1. Inverse Compatible (**inv_c):**The block is inv_c when it is inverse compatible with the previous block.

2. Half Compatible (**half_c**):The combined block has property of half_c if two halves of the combined block are compatible.

3. Half Inverse (**half_i**): The combined block is half_i if two splits of the combined blocks are inverse compatible.

4. **0U**: The combined block is 0U when the initial half bits are all 0s and other half is varied (0, 1 or don't care(X)).

5. **1U**: The combined block has property of 1U when the first half bits are all 1s and other half is varied (0, 1 or don't care(X)).

6. **U0**: The combined block belongs to this type when the initial half bits are varied (0, 1 or don't care(X)) and other half bits are all 0s.

7. **U1:** The combined block is U1 when the first half bits are varied (0, 1 or don't care(X)) and other half bits are all 1s.

8. **UU**: The combined block is UU when whole block is mixed (0, 1 or don't care(X)).

Figure 3.1 shows the proportion of occurrence of merged blocks of eight different cases. It has been observed that more numbers of blocks are merged in case of inv_c, half_c and half_i as compared to 0U, U0 and U1. Thus, most of the compression in case of BM-8C is due to the inverse property of merged blocks. Inverse property of merged blocks is exploited in this technique. Thus, Block merging with eight coding technique gives the high compression as compared to most of the other test compression schemes.

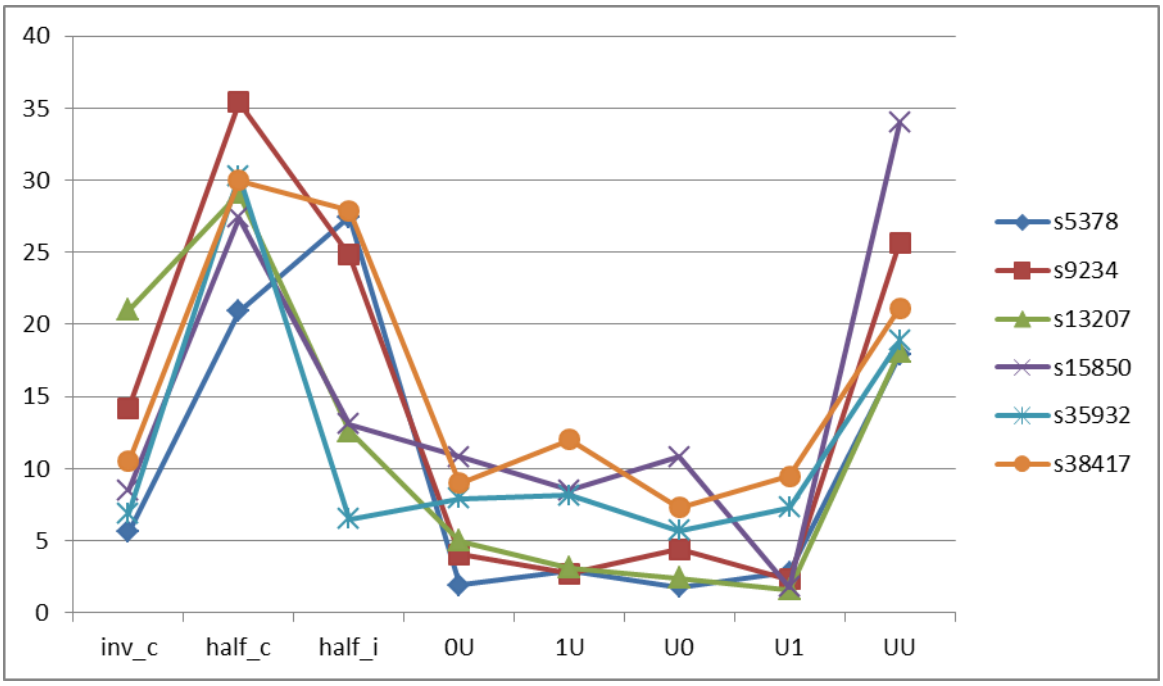

Figure 3.1 Percentage of Occurrence of Merged Blocks

3.2 Limitations of Block Merging with Eight-Coded Compression Technique

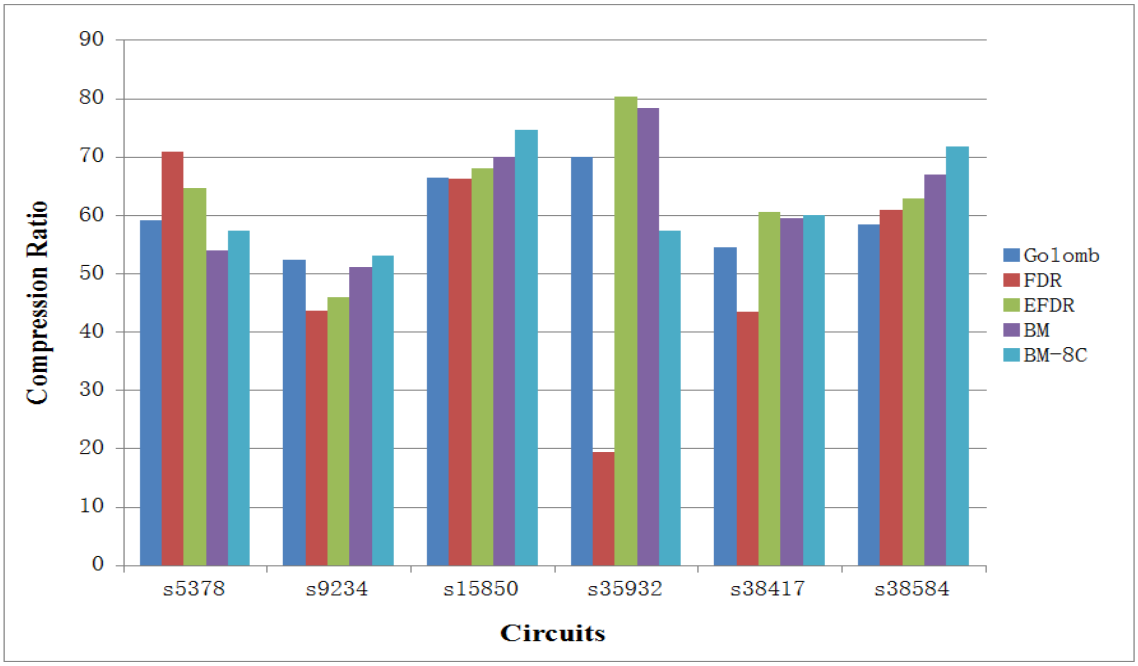

Figure 3.2 Comparison of Compression Ratio with other Compression Techniques

In Figure 3.2 It has been observed that the compression with BM-8C of s38417 standard circuit is comparable with other test-independent compaction schemes such as Block merging scheme [12] and Extended frequency dependent run-length coding scheme [4]. Hence, BM-8C [23] will have low test data volume as compared to other test-independent compression schemes.

Run length based techniques are better than BM-8C for various benchmark circuits. Here, FDR [7] and EFDR [4] are better than BM-8C [23] in case of s5378 and s35932 benchmark circuit. This is because BM-8C is not exploiting the case of UU efficiently. In case of UU, complete b-bits need to be used in code word which ultimately increases the compressed test set data. Thus, UU property of block size need to be exploited further using the frequency of run-length.

Also, in case of inv_c, the block is compared with the previous block. Thus, at least two blocks are required to apply this property. It is also not always possible to apply inc_c property to the last block. This is because if the last second block is encoded according to BM-8C, then it will be the output and last block cannot be compared to check the inv_c property.

3.3 Adaptive Block Merging and Eight-Coded Run-length Compaction Technique (ABMEC)

Block merging with eight-coded technique [23] is modified to reduce the compression ratio further by modifying case of UU. In ABMEC the block merging is done initially by combining the successive compatible blocks of original test data. The numbers of combined blocks are stored in Merged Block Count (MGC). It encodes each combined block and the number of successive compatible blocks for each combined block to eight code words based on different properties of successive combined block. The prefix is applied according to the MGC value. During encoding using eight code words in case of UU, the code word for UU is modified depending on the run length of '0' and 'X'. In this scheme, the code word is according to the maximum run length of zeros and 'X' in a block size. The maximum run-length count is given by r. Depending on the value of r, tail is applied after prefix in code word. The proposed technique is given below:

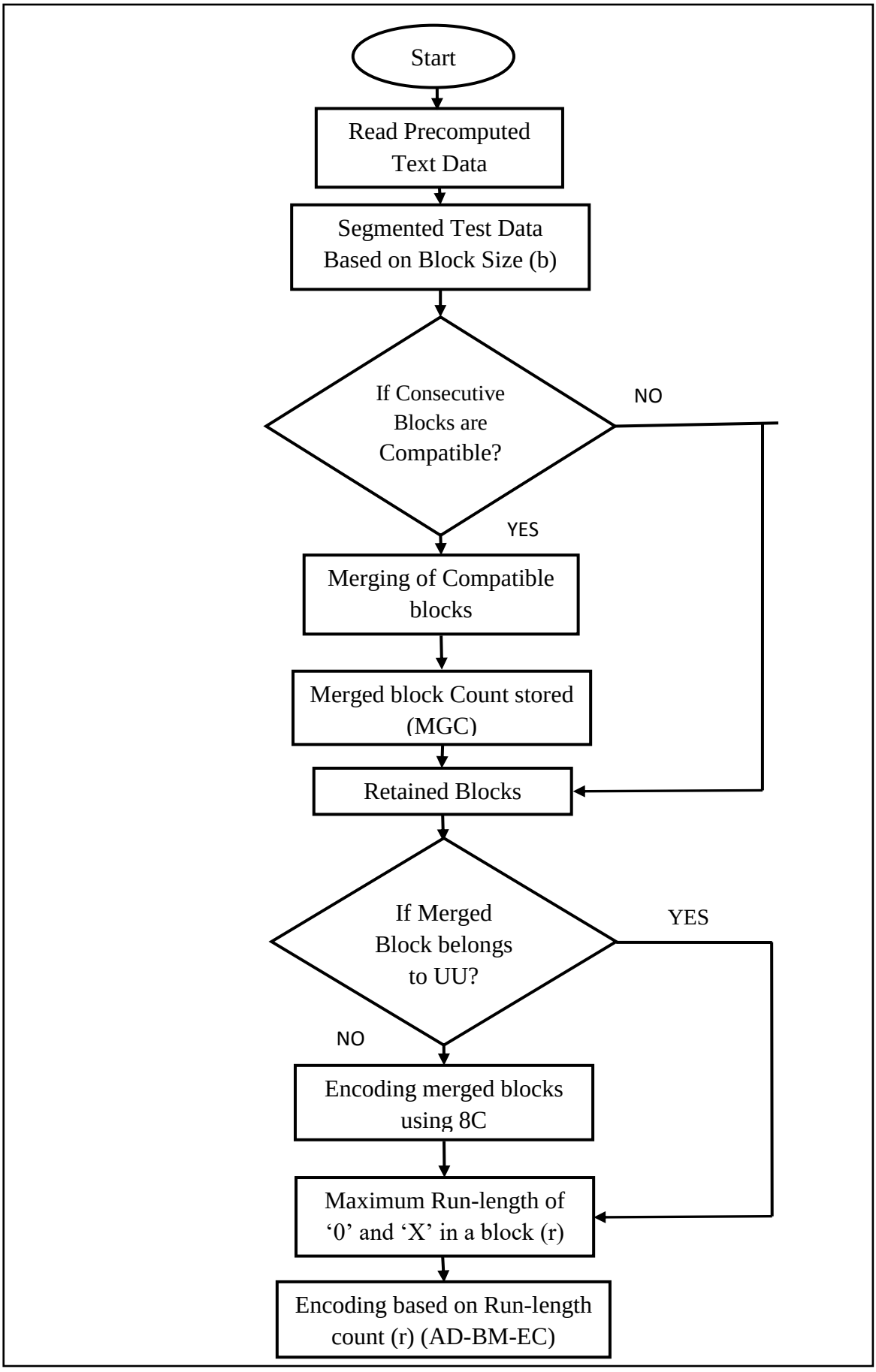

Figure 3.3 ABMEC Test data compaction scheme

The encoding scheme for ABMEC is shown in Table XI.

Table XI: ABMEC Encoding Scheme

Group	MGC	Prefix	Types	Code word
A0	1	0	inv_c	0+010
			half_c	0+11+b/2
			half_i	0+10+b/2
			0U	0+01100+b/2
			1U	0+01101+b/2
			U0	0+01110+b/2
			U1	0+01111+b/2
			UU	
			r=0	0+0000
			r=1	0+0001
			r=2	0+0010
			r=3	0+0011
			----	----
			r=15	0+1111
A1	2-3	10	inv_c	10+1 bit+010
			half_c	10+1 bit+11+b/2
			half_i	10+1 bit+10+b/2
			0U	10+1 bit+01100+b/2
			1U	10+1 bit+01101+b/2
			U0	10+1 bit+01110+b/2
			U1	10+1 bit+01111+b/2
			UU	
			r=0	10+0000

			r=1	10+0001
			r=2	10+0010
			----	----
			r=15	10+1111
A2	4-7	110	inv_c	110+2 bit+010
			half_c	110+2 bit+11+b/2
			half_i	110+2 bit+10+b/2
			0U	110+2 bit+01100+b/2
			1U	110+2 bit+01101+b/2
			U0	110+2 bit+01110+b/2
			U1	110+2 bit+01111+b/2
			UU	
			r=0	110+0000
			r=1	110+0001
			r=2	110+0010
			----	----
			r=15	110+1111
A3	8-15	1110	inv_c	1110+3 bit+010
			half_c	1110+3 bit+11+b/2
			half_i	1110+3 bit+10+b/2
			0U	1110+3 bit+01100+b/2
			1U	1110+3 bit+01101+b/2
			U0	1110+3 bit+01110+b/2
			U1	1110+3 bit+01111+b/2
			UU	
			r=0	1110+0000
			r=1	1110+0001

			r=2 ---- r=15	1110+0010 ---- 1110+1111
A4	16-31	1111	inv_c half_c half_i 0U 1U U0 U1 UU r=0 r=1 r=2 ---- r=15	1111+4 bit+010 1111+4 bit+11+b/2 1111+4 bit+10+b/2 1111+4 bit+01100+b/2 1111+4 bit+01101+b/2 1111+4 bit+01110+b/2 1111+4 bit+01111+b/2 1111+0000 1111+0001 1111+0010 ---- 1111+1111

CHAPTER-4

SIMULATION RESULTS AND ANALYSIS

The experimental results have verified that ABMEC approach is capable for improving compression of the test patterns. Furthermore, it reduces the time of testing. ABMEC is implemented on various ISCAS standard circuits.

4.1 Results of ABMEC Technique

4.1.1 Compression Ratio

The test set T_D is original test data for an Intellectual Property core. T_E is the test set after compression which is obtained by encoding T_D to a considerably reduced set. The test data set size is represented by $|T_D|$ whereas $|T_E|$ is the compacted set size.

The compacted set T_E is kept in ATE memory. For decompression, during testing decoder is used to obtain T_D from T_E. The size of compressed test set $|T_E|$ is decided by the test data compaction method used and by the original T_D. Thus, the final T_E depends on the choice of T_D. This is because test data compression is challenging for test set with haphazard distribution of 0s and 1s. The test data Compression Ratio (CR) is given by the ratio of the number of bits per sample compressed to the original data rate in test pattern. It is a degree of complexity of test data set.

$$\text{Compression Ratio} = \frac{\text{Compressed Test vector length}}{\text{Original Test vector length}}$$

$$\text{Compression Ratio} = \frac{T_D - T_E}{T_D} \times 100$$

where,

T_D : Original bits in initial test pattern

T_E : Compressed bits of test set.

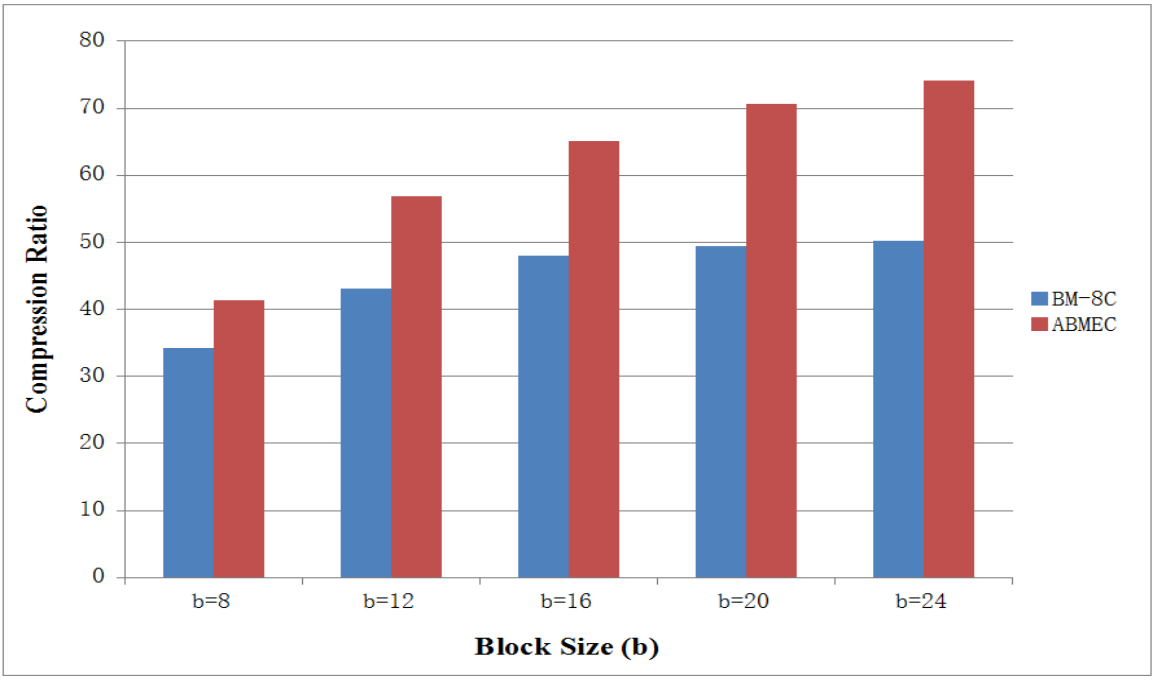

Figure 4.1Comparison of Compression Ratio for circuit s9234 with BM-8C [23]

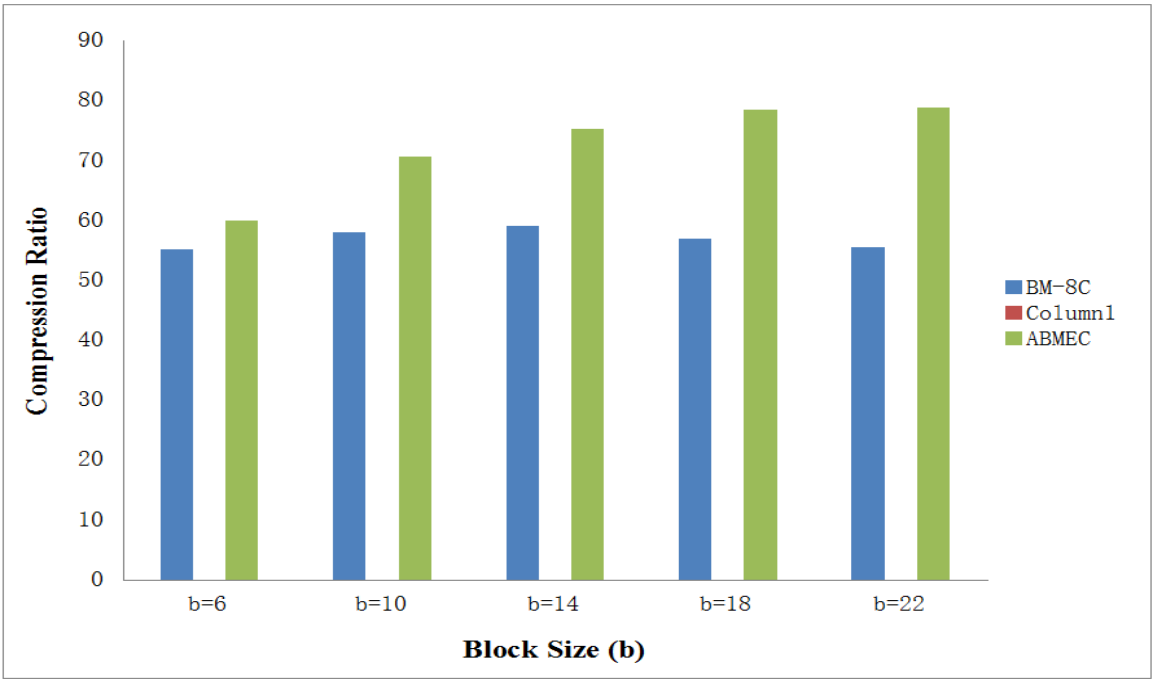

Figure 4.2 Comparison of Compression Ratio for circuit s5378 with BM-8C [23]

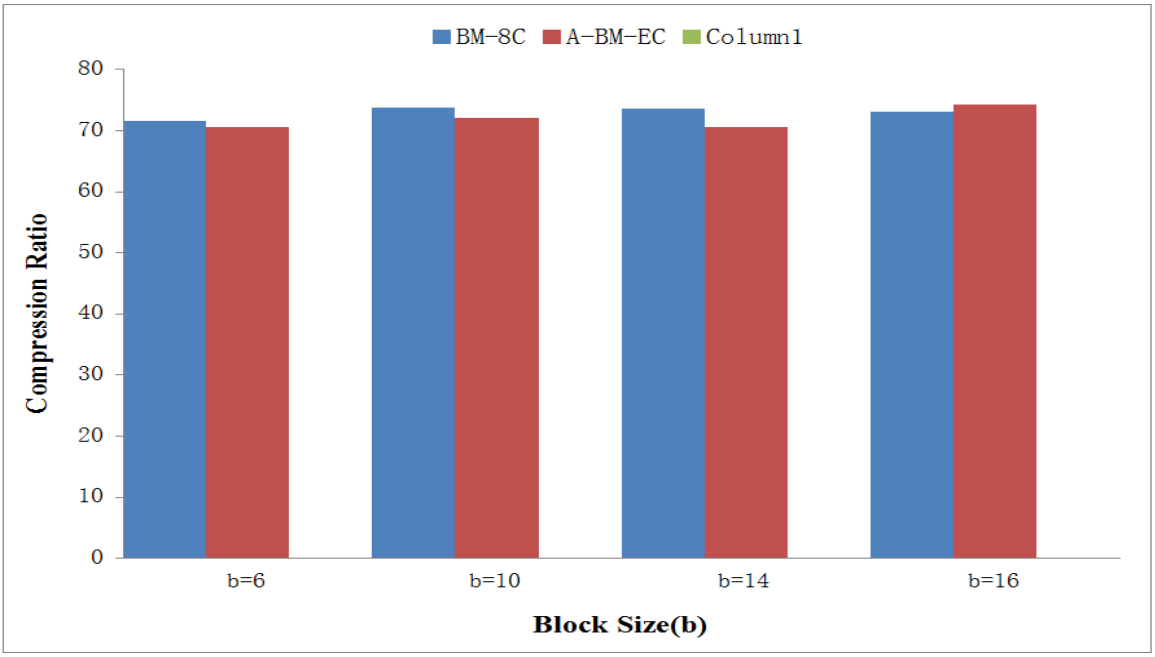

Figure 4.3 Comparison of Compression Ratio for circuit s15850 with BM-8C [23]

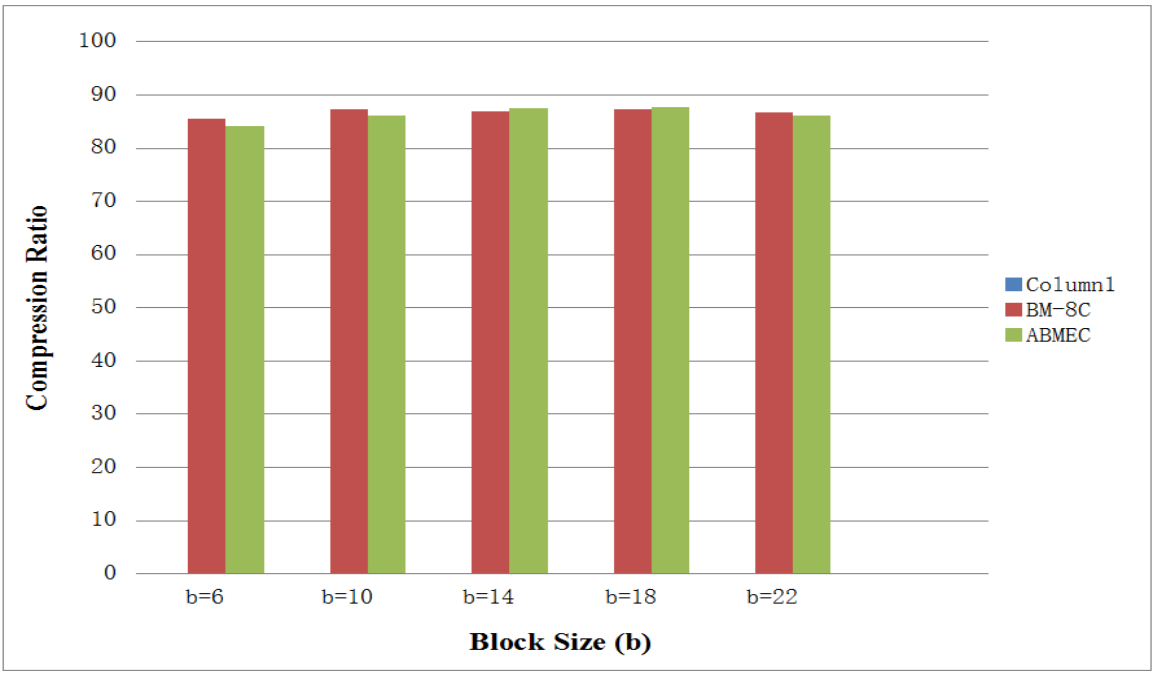

Figure 4.4 Comparison of Compression Ratio for circuit s13207 with BM-8C [23]

When compression ratio of ABMEC is compared with BM-8C [23] then it has been observed that ABMEC has outperformed BM-8C [23] in all the benchmark circuits. Thus, ABMEC gives the maximum compression ratio due to the efficient use of run lengths of '0' and 'X' in the case of UU that consists of random bits (mixed bits- 1, 0 or X).

4.1.2 Test Application Time

One of the crucial goals of any SOC testing is to reduce test application time (TAT) along with decrease test data volume. It depends on the time needed to move the encoded test patterns from the tester to the SOC and the time needed to decode the data after compression. Test data compression drops testing time and allows the use of low cost ATE.

Assume the compressed data has N code words, C_1 – C_N. The minimum test application time is given as:

$$TAT_{min} = \sum_{i=1}^{N} W_i + \frac{\lceil \max(H_N - (\alpha - 2), 0 \rceil}{\alpha}$$

where,

W_i : Length of each code word (i=1,2,…….N)

H_i : Length of decompressed test data for corresponding code word C_i.

An optimum frequency ratio is to be less than the frequency ratio to have minimum TAT. It is given by the extent of test data set after compression. Therefore, by increasing the compression ratio, the TAT will be reduced further.

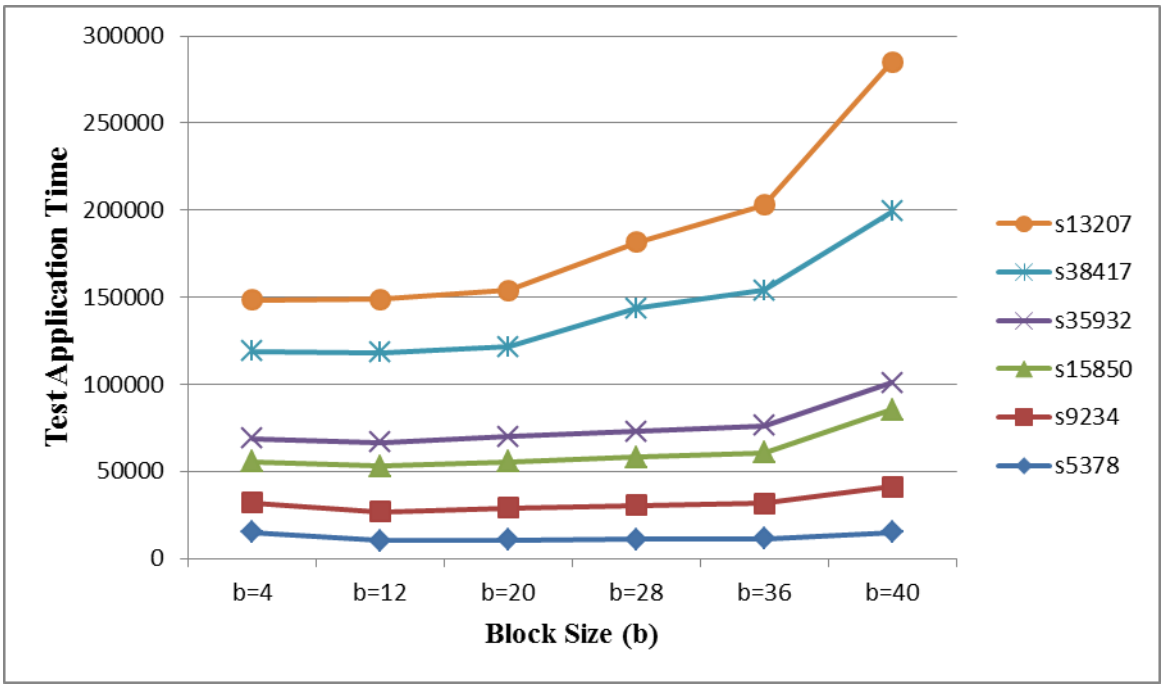

Figure 4.5 Variation of Test Application Time with Block size

Variation of test application time with respect to block size (b) is shown in Figure 4.5. It has been observed that when the block size for test set decreases, the test application time for benchmark circuits decreases. Thus, testing time is low when small block sizes are used. This is because for low block sizes the number of blocks lie in the eight categories will be large in eight coding. Hence, more number of bits will be compressed using BM-8C coding scheme. Therefore, the number of blocks merged is more for low block size which will decrease the test application time further.

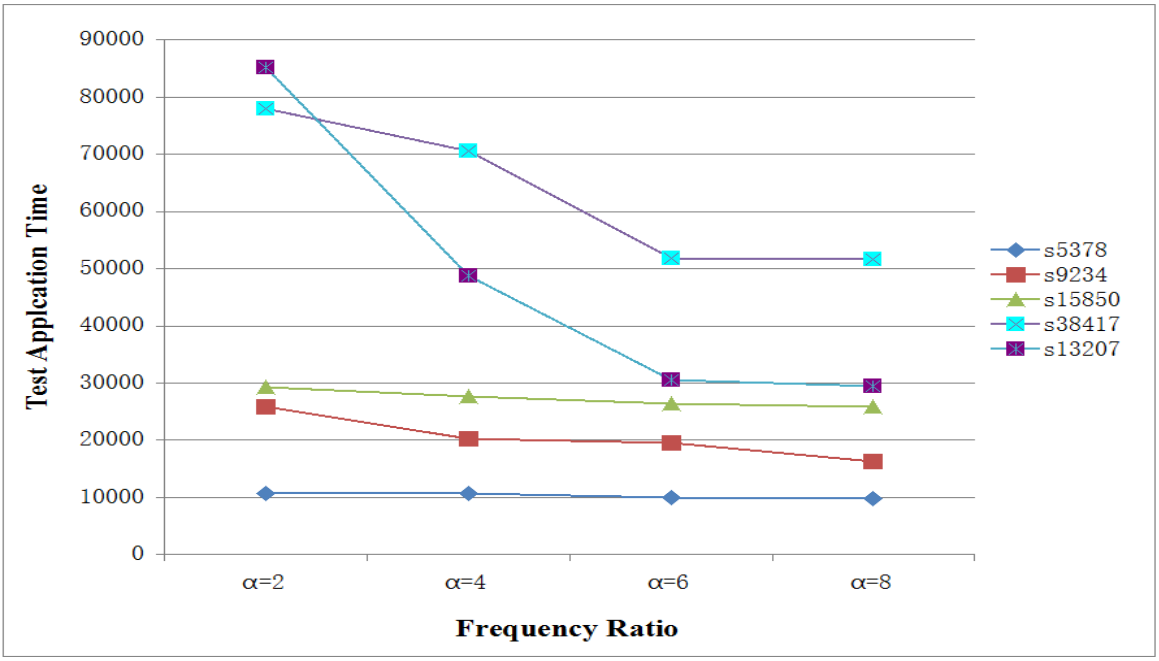

Figure 4.6 Variation of Test Application Time with different frequency ratios.

Ratio between the ATE clock frequency and on chip test frequency is known as frequency ratio. The parameter α should preferably be power of 2. This is because ATE clock is to be synchronized with test clock which is possible only for these values of α. f_{CUT} is the testing frequency of CUT. f_{ATE} is the ATE clock frequency.

$$Frequency\ Ratio, \alpha = \frac{f_{CUT}}{f_{ATE}}$$

Variation of TAT with diverse frequency ratio,α is shown in Figure 4.6. It has been observed that TAT is highest for lower values of frequency ratio. It decreases promptly when the frequency ratio rises initially. But there is very minor and negligible change in test application time when frequency ratio is increases further.

4.2 Results Comparison

4.2.2 Compression Ratio Comparison

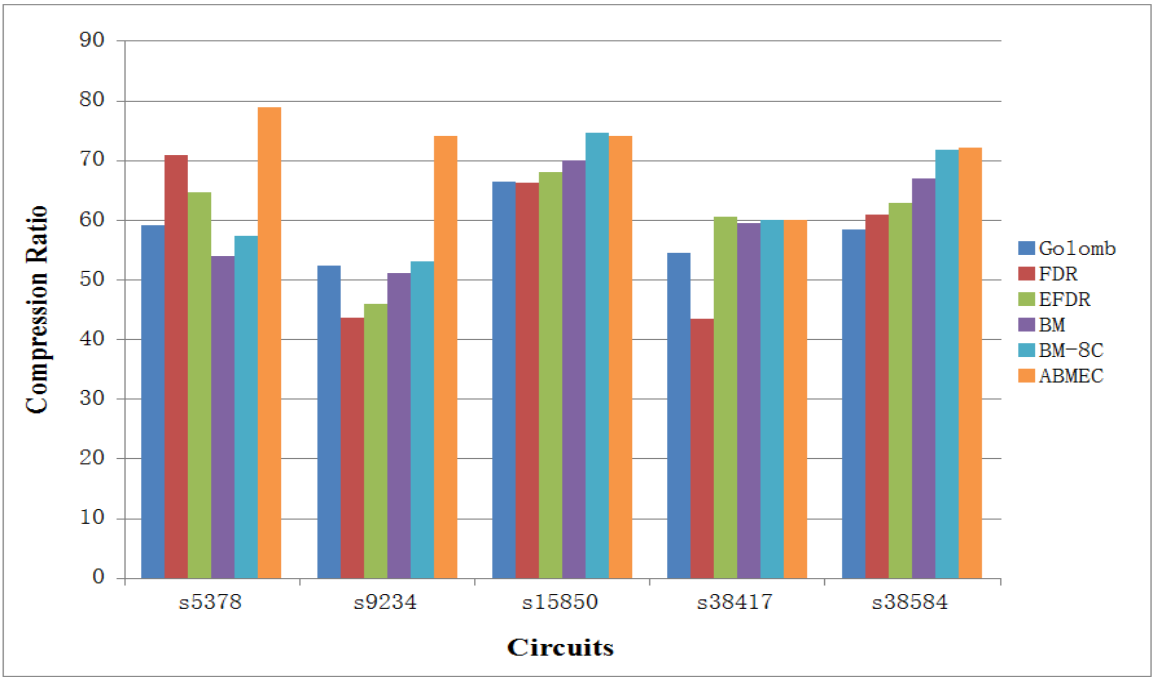

Figure 4.7 Comparison of Compression Ratio with other Code Based schemes for different circuits.

In Figure 4.7 different Code based test compaction techniques based on run lengths and X-filling are compared with proposed Adaptive BM-8C (ABMEC) technique. It has been observed that the compression ratio of Golomb code [2] is low as compared to other techniques. Compression ratio of ABM8C run length technique is comparable with BM-8C [23] in case of s15850, s38417 and s38584 ATE benchmark circuits. It has outperformed other compaction techniques in case of s5378 and s9234 circuits. Thus, the number of bits compressed using ABMEC has increased which results in low volume of test data. This is because ABMEC technique has used the advantageous factors of both BM-8C [23] and run-length based coding schemes. It has used the frequency of occurrence of "0" and "X" in case of presence of random bits in a block. Thus, gives the improved compression ratio. Hence, the test pattern that required to be stored in ATE decreases. This reduces the requirement of memory capacity. Therefore, the test application time of various circuits' decreases further by using ABMEC.

4.2.3 Test Application Time Comparison

Table XII: Comparison of Test Application Time of ABMEC with other Compression Schemes

Circuit	α	FDR [7]	EFDR [4]	BM [12]	BM-8C [23]	ABMEC
S5378	2	24833	16975	16118	15088	10733
	4	16703	13072	12339	11191	10690
	6	15159	12196	11083	10348	9989
	8	13939	11752	11899	10089	9876
S9234	2	42066	26129	26336	24281	22892
	4	29206	21424	20828	18410	17277
	6	26675	20557	19762	17278	16080
	8	24086	20318	19426	16921	14205
S15850	2	65020	46076	46076	44110	46356
	4	42270	32517	32084	29553	32707
	6	36732	28798	28216	25522	26333
	8	32362	27172	26518	23673	25923
S38417	2	186261	104569	109180	106725	100062
	4	123700	75614	80273	79074	70562
	6	113451	68212	73286	71564	71289
	8	110521	65509	70202	69069	62697
S13207	2	116101	88487	88045	87319	86219
	4	70361	52711	50784	49730	48728
	6	57089	41898	39177	38138	36476
	8	48358	36946	33768	32326	29545

Test Application Time (TAT) of ABMEC is compared with other test data compression techniques such as FDR [7], EFDR [4], Block Merging [12]and BM-8C [23] in Table XII. The frequency ratio, α varies from 2 to 8. ABMEC technique has the reduced test application time as compared to other test independent compression techniques under diverse frequency ratio, α.

CHAPTER-5
CONCLUSION AND FUTURE SCOPE

Various run-length techniques have been proposed in the past. The ABMEC technique uses the run-lengths of '0' and 'X' to exploit the case of UU further to get compressed data. When applied to various standard circuits, compression of test vector increases significantly using ABMEC. ABMEC addresses the issue of less compression ratio of BM-8C [23] as compared to run-length codes FDR [7] and EFDR [4].

Experimental results have shown that ABMEC can attain significantly high test data compression ratio. It attains considerably low TAT. This is because TAT directly depends on the compressed test data. But, it is more complex when compared with other code based schemes.

Low power testing is required to increase the circuit reliability. This can be done by reducing the switching activity between the successive test sets. Switching activity can be lowered by exploiting don't care bits of the original data set. This can be done by using the rate of occurrence of '1' and 'X' in the case of UU along with ABMEC. Thus, run-length codes can be combined with ABMEC to have lower switching activity in corresponding test sets. The length of code words in cases of U can be reduced further.

REFERENCES

[1] A. Jas and N. Touba, "Test vector compression via cyclical scan chains and its application to testing core-based designs", in Proceedings of the IEEE International Test Conference (ITC'98), pp. 458-464, IEEE CS Washington, DC, USA, October 1998.

[2] A. Chandra and K. Chakrabarty, "Efficient Test Data Compression and Decompression for system-on-a-chip using internal scan chains and Golomb coding" in Proceedings of Design, Automation and Test in Europe, pp. 145-149, March 2001.

[3] A. Chandra and K. Chakarbarty, "Reduction of SOC Test Data Volume, Scan Power and Testing Time Using Alternating Run length Codes",in Proceedings of the 39th conference on Design automation, pp. 673-678, June 2002.

[4] Aiman H. El-Maleh and Raslan H. Al-Abaji, "Extended Frequency-Directed Run-Length Code with Improved Application to System-on-a-Chip Test Data Compression", International Conference on Electronics, Circuits and Systems, vol. 2,449-452,September 2002.

[5] S. Hellenbrand and A. Wurtenberger, "Alternating run-length coding – a technique for improved test data compression" in Proceedings of the 3rd IEEE International Workshop on Test Resource Partitioning (TRP'02), October 2002.

[6] Abhijit Jas, Jayabrata Ghosh-Dastidar, Mom-Eng Ng, and Nur 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, June 2003.

[7] A. Chandra and K. Chakarbarty, "Test Data Compression and Test Resource Partitioning for System-on-Chip Using Frequency-Directed Run-Length (FDR) Codes," IEEE Trans. Computers, vol. 52, no. 8, pp.1076-1088, Aug 2003.

[8] Lei Li, KrishnenduChakrabarty and Nur A. Touba, "Test Data Compression Using Dictionary with Selective Entries andFixed-Length Indices", Journal of ACM Transactions on Design Automation of Electronic Systems (TODAES), vol. 8, Issue 4, October 2003.

[9] Mohammad Tehranipour, MehrdadNourani, KrishnenduChakrabarty, “Nine-Coded Compression Technique with Application to Reduced Pin-Count Testing and Flexible On-Chip Decompression” in Proceedings of the Design, Automation and Test in Europe Conference, pp. 1530-1591, April 2004.

[10] Anshuman Chandra and KrishnenduChakrabarty, “Analysis of Test Application Time for Test Data Compression Methods Based on Compression Codes”, Journal of Electronic Testing: Theory and Applications 20, pp.199-212, 2004.

[11] Yannick Bonhomme, Patrick Girard, Lois Guiller, Christian Landrault, serge, “A Gated Clock Scheme for Low Power Testing of Logic Cores”, Journal ofElectronic Testing: Theory and Applications 20, pp. 89–99, 2006.

[12] Aiman El-Maleh, “Efficient Test Vector Compression Technique Based on Block Merging”, in Proceedings of conference on Circuits and Systems, May 2006.

[13] XrysovalantisKavousianos, EmmanouilKalligeros, and Dimitris Nikolos, “Optimal Selective Huffman Coding for Test-Data Compression”, IEEE transactions on computers, vol. 56, no. 8, August 2007.

[14] XrysovalantisKavousianos and Dimitris Nikolos, “Multilevel Huffman Coding: An Efficient Test-Data Compression Method for IP Cores”,IEEE transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 26, no. 6, June 2007.

[15] Santiago Remersaro, “On low power test and DFT techniques for test set Compaction”, Ph.D. Dissertation, University of Iowa, 2008.

[16] J. Feng and G. Li, “A test data compression method for system-on-a-chip”, in Proceedings of the 4th IEEE International Symposium on Electronic Design, Test and Applications (DELTA’08), January 2008.

[17] KandBasu, Prabhat Mishra, “Test Data Compression Using Efficient Bitmask and Dictionary Selection Methods”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, no. 9, September 2010.

[18] Usha S. Mehta, K.S. Dasgupta, Nranjan M. Devashrayee, “Hamming Distance Based Reordering and Column wise Bit Stuffing with Difference Vector: A Better Scheme for Test Data Compression with Run Length Based Codes”,23rd International Conference on VLSI design, 2010.

[19] Deepika Sharma, Debbrat Ghosh and Harpreet Vohra, “Test Data Volume Minimization using Double Hamming Distance Reordering with Mixed RL-Huffman based compression scheme for System-on-chip”,Nirma University International Conference on Engineering, NUiCONE-2012, pp.06-08, December 2012.

[20] Lung-Jen Lee, Wang-Dauh Tseng, Rung-Bin Lin, and Cheng-Ho Chang, “Pattern Run-Length for Test Data Compression”, IEEE Transaction on Computer –Aided Design of Integration Circuits and System, vol. 31, no. 4, April 2012.

[21] Panagiotis Sidmanoglou and Dimitris Nikolos, “Input Test Data Compression Based on the Reuse of Parts of Dictionary Entries: Static and Dynamic Approaches”, IEEE Transaction on Computer–Aided Design of Integration Circuits and System, vol.32, no.11, November 2013.

[22] Sathiyapriya.R ,YuvasrriSindhu.M , Immanuel Rairosario.P, “FPGA Implementation of Hybrid Test Data Compression Method Using Scan Chain Compaction and Dictionary based Scheme”, International Journal of Scientific and Research Publications, ISSN 2250-3153, April 2013.

[23] Tie-Bin Wu, Heng-Zhu Liu and Peng-Xia Liu, “Efficient Test Compression Technique for SoC Based on Block Merging And Eight Coding”, Journal of electronic testing: Theory and Applications, vol. 24, pp. 849–859, 2013.

[24] Karen Thangam Jacob, K.S. Ganesh Kumar and B. Manjurathi, “Selective Compression Techniques using Variable-to-Fixed and Fixed-to-Variable Codes”, International Journal of Computer Applications (ISSN 0975-8887), vol. 89, March 2014.

[25] BaldeLavanya and G. Prasad Acharya, “Huffman Based SOC Test Data Compression” International Journal of Scientific Engineering and Technology Research (ISSN 2319-8885),vol. 03, pp. 7477-7480, November 2014.

[26] RajitKarmakar and Santanu Chattopadhyay, “Thermal-Aware Test Data Compression Using Dictionary Based Coding”, 28th International Conference on VLSI Design, ISSN 1063-9667 , January 2015.

[27] G.Pavithra Devi and K.S. NeeluKumari, “Test Data Compression and optimal power seed selection for scan power reduction”, International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, Issue 4, April 2015.

[28] G. MeenaKumari and T.Sivarama Krishnan, “Data Compression using Bitmask and Dictionary Selection Methods”, International Journal of Innovative Research in Computer and Communication Engineering, vol. 3, Issue 5, May 2015.

[29] Dr, Ahmad Odat, Dr, Mohammed Otair and Mahmoud Al-Khalayleh, “Comparative Study between LM-DH Technique and Huffman-Coding Technique”, International Journal of Applied Engineering Research, vol. 10, pp.36004-36011, November 2015.

[30] C. Kalamani and K. Paramasivam, “Test Data Compression Using a Hybrid of Bitmask Dictionary and 2^n Pattern Run-length Coding Methods”, International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol. 9, 2015.

[31] S.Indu, M.Priyanga, R.Rajasanthiya and R.Sowmiya, “Efficient Test Data Compression Using Dictionary and Bit masking on FPGA”, International Journal of Advanced Research Trends in Engineering and Technology, vol. 3, Issue 2, March 2016.

601461008_Deepinder_Kaur.DOCX

ORIGINALITY REPORT

24%	8%	24%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Usha S. Mehta. "Run-Length-Based Test Data Compression Techniques: How Far from Entropy and Power Bounds?—A Survey", VLSI Design, 2010 Publication	3%
2	Wu, Tie-Bin, Heng-Zhu Liu, and Peng-Xia Liu. "Efficient Test Compression Technique for SoC Based on Block Merging and Eight Coding", Journal of Electronic Testing, 2013. Publication	2%
3	eprints.kfupm.edu.sa Internet Source	2%
4	Xrysovalantis Kavousianos. "", IEEE Transactions on Computers, 8/2007 Publication	1%
5	www.ijcaonline.org Internet Source	1%
6	N. Kelly. "BIST vs. ATE for testing system-on-a-chip", Proceedings International Test Conference 1998 (IEEE Cat No 98CH36270) TEST-98, 1998 Publication	1%