Artificial Intelligence Based Test Data Compression Techniques

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

in

Electronics and Communication Engineering

(VLSI DESIGN)

By

Harikrishna Parmar

09MEC014

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

AHMEDABAD-382481

May 2011

Artificial Intelligence Based Test Data Compression Techniques

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology

in

Electronics and Communication Engineering

(VLSI DESIGN)

By

Harikrishna Parmar

09MEC014

Guided by

Prof. Usha Mehta

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

AHMEDABAD-382481

May 2011

Declaration

This is to certify that

i) The thesis comprises my original work towards the degree of Master of Technology in Information and Communication Technology at Nirma University and has not been submitted elsewhere for a degree.

ii) Due acknowledgement has been made in the text to all other material used.

Harikrishna Parmar

Certificate

This is to certify that the Major Project entitled ”Artificial Intelligence Based Test Data Compression Techniques” submitted by Your Name (09MEC014), towards the partial fulfillment of the requirements for the degree of Master of Technology in Electronics and Communication Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Usha.S Mehta
Guide, Professor,
PG VLSI Design,
Institute of Technology,
Nirma University, Ahmedabad

Dr. N.M Devashrayee
P.G.Coordinator,
PG VLSI Design,
Institute of Technology,
Nirma University, Ahmedabad

Prof. A. S. Ranade
Head of Dept, Professor,
Institute of Technology,
Nirma University, Ahmedabad

Dr. K.Kotecha
Director, Professor,
Institute of Technology,
Nirma University, Ahmedabad

Abstract

The circuit consumes more power during test mode compared to functional mode. So test power has been major big concern in large System-on-Chip designs from last decade. In the first part of report, the state-of-the-art in low power testing is presented. The first part contains the detailed survey on various power reduction techniques proposed for both the aspects of testing i.e. external testing as well as Built-In-Self-Test. The advances in DFT techniques emphasizing low power is also included in this part.

In the second part novel methods are presented which aims at minimizing the total power consumption during testing. This is achieved by minimizing the switching activity in the circuit by reducing the Hamming Distance between successive test vectors. In this method the test vectors are reordered for minimum total hamming distance and the same vector set is used for testing. Artificial intelligence is another approach that is described to reduce switching activity. Also Test power has become a serious problem with scan-based testing. It can lead to prohibitive test power in the process of test application. During the process of scan shifting, the states of the flip-flops are changing continually, which causes excessive switching activities. Test vector reordering for reducing scan in scan out power is one of the general goal of low power testing. WTM based reordering technique have proposed here to reorder the test vectors in an optimal manner to minimize switching activity during testing.

In the third part it is shown that MT Fill algorithm is preferable when power is a big concern. Frequency directed bit filling approach is preferable when compression is a major goal. But the proposed approach that is a combination of MT fill and frequency directed bit filling is preferable when one wants to achieve moderate power and compression.

The final part addresses error-resilience that is the capability to tolerate bit-flips in a compressed test data stream (which is transferred from an Automatic Test Equipment

(ATE) to the Device-Under-Test (DUT)). In an ATE, bit-flips may occur in either the electronics components of the loadboard, or the high speed serial communication links (between the user interface workstation and the head). It is shown that errors caused by bit-flips can seriously degrade the test quality (as measured by coverage) of the compressed data streams. The effects of bit-flips on compression are analyzed and various test data compression techniques are evaluated.

Acknowledgements

No Project work is ever the outcome of efforts or talent of single individual. This is no exception. Many teachers, friends, well-wishers have contributed to this work directly or indirectly and made it possible for me to present it in its present shape. Although it is not possible for me to name and thank them all individually, I must make special mention some of the personalities and acknowledge my sincere indebtedness to them and preserve gratitude to many others in my heart.

"You can even do right things you want and you can even do it for the right reasons you want. But if you don't have the right and proper guidance, you can never hit a right target." For that I express my deep and sincere gratitude to P.G.Coordinator Dr. N. M. Devashrayee for his constant encouragement, valuable guidance and constructive suggestions during all the stages of the Project work. He has been great source of inspiration and help me during my whole course by providing me the necessary guidance in every way to build the better future.

I am deeply indebted to my guide Prof. Usha Mehta for her constant guidance and motivation. she has devoted significant amount of his valuable time to plan whole project work. Without her experience and insights, it would have been very difficult to do quality work.

I am also thankful to Prof. N. P. Gajjar, Prof. Amisha P. Naik and all the faculty members of Department of Electronics & Communcnication Engineering, Institute of Technology, Nirma University, Ahmedabad, for providing me all the necessary guidance through out the term which provides lots of help in course of my Project work. I also gratefully acknowledge Ms. Niti Avasthi for providing me the full laboratory support at PG-VLSI Design Lab.

- Harikrishna Parmar

09MEC014

Contents

Declaration **iii**

Certificate **iv**

Abstract **v**

Acknowledgements **vii**

List of Tables **xi**

List of Figures **xii**

1 Introduction **2**
- 1.1 The challenge to test 2

2 Literature Survey **5**
- 2.1 Terminology 5
- 2.2 Energy and Power Modelling 6
- 2.3 Low Power External Testing Techniques 8
  - 2.3.1 Low Power ATPG Algorithm 8
  - 2.3.2 Ordering Techniques 9
  - 2.3.3 Input Control 10
  - 2.3.4 Vector Compaction and Data Compression 10
  - 2.3.5 Scan Chain Transformation 11
  - 2.3.6 Clock Scheme Modification 12
- 2.4 Low Power BIST Techniques 12
  - 2.4.1 Low Power Test Pattern Generators 13
  - 2.4.2 LFSR Tunning 14
  - 2.4.3 Vector Filtering BIST 15
  - 2.4.4 Circuit Partitioning 16
  - 2.4.5 Low Power RAM Testing 16
- 2.5 Low Power DFT Techniques 17

3 Reordering Techniques **18**
3.1 Hamming Distance based Reordering technique 18
3.1.1 Problem formulation 21
3.1.2 Total Hamming Distance(THD) 21
3.1.3 Reordering Algorithm: 22
3.2 Artificial Intelligence approach 24
3.2.1 Brief overview 24
3.2.2 Prior Work 25
3.2.3 Motivation of the work and problem formulation 28
3.2.4 An A* based method for dynamic power minimization by test vector reordering 29
3.2.5 Cost function g(n) 30
3.2.6 Computing lower bound of switching activity: Heuristic function h(n)) 31
3.2.7 The A* based algorithm:AITVR 32
3.2.8 Time complexity of AITVR 33
3.2.9 Empirical Observation 33
3.3 WTM based Reordering of Combine Test Vector and Output Response 35
3.3.1 Problem formulation 35
3.3.2 Cost Function g(n) 35
3.3.3 Empirical observation 37

4 Bit Filling Algorithm **39**
4.1 MT Fill Algorithm 39
4.2 Frequency Directed Bit filling Algorithm 40
4.2.1 For Optimization of Compression and Power Both 41
4.2.2 Zone wise Bit Filling 43
4.3 Mixed approach : Frequency directed + MT Fill 46

5 Compression Algorithm **50**
5.1 Test Stimulus Compression 52
5.2 Code-Based Schemes 53
5.2.1 Dictionary Code (Fixed-to-Fixed) 54
5.2.2 Huffman Code (Fixed-to-Variable) 55
5.2.3 Run-Length Code (Variable-to-Fixed) 58
5.2.4 Golomb Code (Variable-to-Variable) 59

6 Error Resilience **64**
6.1 Brief Overview 64
6.1.1 Previous Work done 65
6.1.2 Basic Principles 66
6.2 Analysis 75
6.2.1 Propagation and Shifts 76

6.2.2 Synchronization Loss 78
6.3 Improving Error-Resilience 78
6.3.1 Traditional Techniques 78

7 Hamming Code based technique **80**
7.0.2 Generating Algorithm 81

8 Cyclic redundancy check (CRC) **89**
8.0.3 Computation of CRC 90

9 Two dimensional Parity check Code **93**

10 Conclusion and Future Scope **98**
10.1 Conclusion 98
10.2 Future Scope 99

References **100**

List of Publication **105**

List of Tables

I Comparison of Switching activity in Hamming and AI 38

I Test data set for frequency directed bit filling 41
II Comparison of Switching activity in different ISCAS Circuits 48
III Comparison of Compression in different ISCAS Circuits 49

I Bit Flip without Compression 68
II Bit Flip after Compression for Huffman Coding 69
III Bit Flip after Compression for Selective Huffman Coding 70
IV Bit Flip after Compression for FDR Coding 71
V Bit Flip after Compression for EFDR Coding 72
VI Bit Flip after Compression for SAFDR Coding 73
VII Bit Flip after Compression for GOLOMB Coding 74
VIII Bit Flip after Compression for VIHC Coding 75

I Area Overhead with Huffman Code 85
II Area Overhead with selective Huffman Code 86
III Area Overhead with FDR Code 86
IV Area Overhead with EFDR Code 87
V Area Overhead with SAFDR Code 87
VI Area Overhead with GOLOMB Code 88
VII Area Overhead with VIHC Code 88

I Bits Overhead with Huffman Code 94
II Bits Overhead with Selective Huffman Code 95
III Bits Overhead with FDR Code 95
IV Bits Overhead with EFDR Code 96
V Bits Overhead with SAFDR Code 96
VI Bits Overhead with GOLOMB Code 97
VII Bits Overhead with VIHC Code 97

List of Figures

3.1 a.)Test Vectors for Full adder circuit b.) Switching matrix hd[n][n] . . 23
3.2 Test Vectors of C17 30
3.3 Transition graph of C17 30
3.4 Finding optimal path from informed search space 34
3.5 a.) Test Vector b.)Output Response 36
3.6 Test Vector passing through Output Response 36
3.7 Finding optimal path in informed search space 37

4.1 Test Vector Sequence 44
4.2 Transitions in case of Opposite Bit Filling 44
4.3 Transitions in case of Same Bit Filling 45
4.4 Test Vectors 47
4.5 Part of data for a.)Frequency Directed b.MT Fill) 47
4.6 MT Filled data is mapped in above fig. 47

5.1 Block diagram illustrating test data bandwidth 51
5.2 Architecture for test compression 52
5.3 Test compression using a complete dictionary 55
5.4 Statistical Coding based on symbole frequencies 56
5.5 Example of test set divided into 4 bit block 56
5.6 Huffman Tree 57
5.7 3 Bit runlength code 58
5.8 Cyclical scan architecture for applying difference vectors 59
5.9 GOLOMB Code 60
5.10 FDR Code 61
5.11 EFDR Code 61
5.12 SAFDR Code 62
5.13 VIHC Code 63

6.1 Conceptual view of the shift effect due to a bit-flip 76
6.2 Conceptual view of the propagation effect due to a bit-flip. 77

7.1 Hamming code example 82

8.1 Padded bits with CRC 90
8.2 Computation if CRC 91
8.3 Rules for divisor 91
8.4 Correctness of CRC Check 92

9.1 Two dimensional Parity check Matrix 93
9.2 Effect of error on Two dimensional Parity check Matrix 94

Chapter 1

Introduction

THE SYSTEM-ON-A-CHIP (SoC) revolution challenges both design and test engineers, especially in the area of power dissipation. Generally, a circuit or system consumes more power in test mode than in normal mode. This extra power consumption can give rise to severe hazards in circuit reliability or, in some cases, can provoke instant circuit damage. Moreover, it can create problems such as increased product cost, difficulty in performance verification, reduced autonomy of portable systems, and decrease of overall yield. Low power dissipation during test application is becoming increasingly important in todays VLSI systems design and is a major goal in the future development of VLSI design.

1.1 The challenge to test

Modern chip design has greatly advanced with recent silicon manufacturing technology improvements that escalate transistor counts, increasing a chips complexity while maintaining its size. This phenomenon will continue, resulting in at least 10 of todays microprocessors fitting onto a single chip by 2005. Consequently, design and test of complex digital circuits imposes extreme challenges to current tools and methodologies. VLSI circuit designers are excited by the prospect of addressing these challenges efficiently, but these challenges are becoming increasingly hard to over-come. Test currently ranks among the most expensive and problematic aspects in a circuit design cycle, revealing the ceaseless need for innovative, test-related solutions. As a result, researchers have developed several techniques that enhance a designs testability through DFT modifications and improve the test generation and application processes. Traditionally, test engineers evaluated these techniques according to various parameters: area overhead, fault coverage, test application time, test development effort, and so forth. But now, the recent development of complex, high-performance, low-power devices implemented in deep-submicron technologies creates a new class of more sophisticated electronic products, such as laptops, cellular telephones, audio- and video-based multimedia products, energyefficient desktops, and so forth. This new class of systems makes power management a critical parameter that test engineers cannot ignore during test development. Test or DFT engineers find their main motivation in considering power consumption during test in a circuits consumption of more power in test mode than during normal operation. IT IS shown that the test power (the power consumed during test) could be twice as high as the power consumed during the normal mode. Several reasons cause this increased power usage. First, test efficiency correlates with toggle rate; therefore, in the test mode the switching activity of all nodes is often several times higher than during normal operation. Second, test engineers use parallel testing in SoCs to reduce the test application time, which might result in excessive energy and power dissipation. Third, the DFT circuitry designed to reduce the test complexity is often idle during normal operation but might be intensively used in the test mode. Fourth, successive functional input vectors applied to a given circuit during system mode have a significant correlation. In contrast, the correlation between consecutive test patterns can be low. For example, in a signal-processing circuit for speech recognition, the input vectors behave predictably, with the least-significant bits more likely to change than the most-significant bits. Similarly, in high-speed circuits that process digital audio and video signals, the inputs to most of those modules change relatively slowly. In fact, designers of low-power circuits take advantage of this consistent behavior when they determine a circuits thermal and electrical limits, and system packaging requirements. In contrast, there is no definite correlation between the successive test patterns generated by an automatic test-pattern generator for external testing or the patterns produced by a linear feedback shift register (LFSR) for built-in selftest (BIST). This lack of correlation can result in significantly greater switching activity in the circuit during test than during normal operation. Because power dissipation in CMOS circuits is proportional to switching activity, tests excessive switching activity can cause catastrophic problems, as detailed later. Although academic research on low-power design remains nearly independent of that for test, industrial practice requires ad hoc solutions for considering power consumption during test application[7]. Practiced solutions include

1. oversizing power supply, package, and cooling to withstand the increased current during testing (test engineers insert breaks into the test process to avoid hot spots);
2. testing with reduced operating frequency; and
3. system-under-test partitioning and appropriate test planning.

The first solution increases both hardware costs and test time. Although the second proposal uses less hardware, the reduced frequency increases test time and might lead to a loss of defect coverage because the reduced frequency can mask dynamic faults. Moreover, this solution reduces power consumption but lengthens test time, so it does not reduce the total energy consumed during test. The third solution of test partitioning and test planning detects dynamic faults, but increases hardware costs and test time. To provide an adequate response to these industrial needs, various researchers have proposed solutions for power problems encountered during test. I classify these solutions into those applicable for external testing and those applicable for BIST. But first, test engineers and designers need to understand the qualities of power under test, as defined in the Terminology sidebar (next page) and the next two sections.

Chapter 2

Literature Survey

To provide an adequate response to the industrial needs, various researchers have proposed solutions for power problems encountered during test. These solutions are classified into those applicable for external testing and those applicable for BIST. But first, test engineers and designers need to understand the qualities of power under test, as defined in the Terminology.

2.1 Terminology

As discussed earlier, one of the current concerns, which may turn into a major engineering problem in the future of SOC development, is test power. These concerns involve energy, average power, peak power, instantaneous power and thermal overload. Below are some definitions of these parameter. Energy: the total switching activity generated during test application. Energy has impact on the battery lifetime during power up or periodic self-test of battery operated devices. Average Power: the total distribution of power over a time period, which is generally the amount of power consumed during the application of a test. The average power is given by the ratio between the energy and the test time. Elevated average power adds to the thermal load that must be vented away from the device under test. It may cause structural damage to the silicon (hot spots), to bonding wires or to the package. Peak Power:

the highest value of power at any given instant. The peak power determines the thermal and electrical limits of components and the system packaging requirements. If the peak power exceeds a certain limit, the correct functioning of the entire circuit is no longer guaranteed. - In fact, the time window for the definition of the peak power is related to the thermal capacity of the chip, and forcing this window to one clock period is sometimes just a simplifying assumption. For example, Test and the circuit has a peak power consumption during only one cycle but it has power consumption within the limit of thermal capacity of the chip for all other cycles, the circuit will not be damaged because the energy consumed, which corresponds to the peak power consumption times one cycle, will not be enough to elevate chip temperature over the limit of thermal capacity of the chip (unless the peak power consumption is far higher than normal power consumption). In order to damage the chip, high (not only highest) power consumption should last for several cycles to consume enough energy that can elevate chip temperature over the limit [6].

2.2 Energy and Power Modelling

Power consumption in CMOS circuits can be classified into static and dynamic. Static power dissipation is due to leakage current or other current drawn continuously from the power supply. Dynamic dissipation is due to (i) short circuit current and (ii) charging and discharging of load capacitance during output switching.

For the current CMOS technology, dynamic power is the dominant source of power consumption, although this may change for future developments of high scaled integration. The average energy consumed at node i per switching is 1/2CiV^{2}DD where Ci is the equivalent output capacitance and VDD the power supply voltage. Therefore, a good approximation of the energy consumed in a period is 1/2CisiV$^2{}_D$D, where si is the number of switching during the period. Nodes connected to more than one gate are nodes with higher parasitic capacitance. Based on this fact, and in a first approximation, capacitance Ci, is assumed to be proportional to the fanout of the node *Fi.* Therefore, an estimation of the energy *Ei* consumed at node *i* during one clock period is:

$$E_i = \frac{1}{2} s_i . F_i . C_0 . V_{DD}^2 \tag{2.1}$$

where C_0 is the minimum size parasitic capacitance of the circuit. According to this expression, the estimation of the energy consumption at the logic level requires the calculation of the fanout F_i and the number of switching on node i, s_i. The fanout of the nodes is defined by circuit topology, and the switching can be estimated by a logic simulator (note that in a CMOS circuit, the number of switching is calculated from the moment the input vector is changed until the moment the internal nodes reach the new stable state, including the hazard switching). The product $s_i F_i$ is named Weighted Switching Activity (WSA) of node *i* and represents the only variable part in the energy consumed at node *i* during test application. According to the above formulation, the energy consumed in the circuit after application of a pair of successive input vectors (V_{k-1}, V_k) can be expressed by:

$$E_{V_K} = \frac{1}{2} . C_0 . V_{DD}^2 \sum_i s(i,k) . F_i \tag{2.2}$$

where*i* ranges all the nodes of the circuit*s and s(i,k)* number of switching provoked*,* by V_k at node *i.* Consider now a pseudo-random test sequence of Lendthtest*,* the test length required to achieve the targeted fault coverage, the total energy consumed in the circuit during application of the complete test sequence is:

$$E_{total} = \frac{1}{2} . C_0 . V_{DD}^2 . \sum_k . \sum_i s(i,k) . F_i \tag{2.3}$$

Let us denote the clock period.*as T.* By definition, the instantaneous power is the power consumed during one clock period. Therefore we can express the instantaneous power consumed in the circuit after application of vectors (V_{k-1},V_k)

as $P_{inst}(V_k)=E_{vk}/T$ The peak power consumption corresponds to the maximum of the instantaneous power consumed during the test session. It therefore corresponds to the highest energy consumed during one clock period, divided by T. More formally, it can be expressed by:

$$P_{peak} = max_k[P_{inst}(V_k)] = \frac{max_k(E_{vk})}{T} \tag{2.4}$$

Finally, the average power consumed during the test session is the total energy divided by the test time, and is given as follows:

$$P_{ave} = \frac{E_{total}}{[(Length_{test}).T]} \tag{2.5}$$

According to the above expressions of the power and energy consumption, and assuming a given CMOS technology and supply voltage for the circuit design, the number of switching, of a node i in the circuit is the only parameter that has impact on both the energy, the peak power and the average power consumption. Similarly, the clock frequency used during testing has impact on both the peak power and the average power. Finally, the test length th$e$ number of test patterns applied to the CUT, has impact only on the total energy consumption. Consequently, when deriving a solution for power and/or energy minimization during test, a designer or a test engineer has to have these relationships in mind [2].

2.3 Low Power External Testing Techniques

Various techniques described in literature to ensure non destructive external testing of an SoC using ATE are categorized in following subsections.

2.3.1 Low Power ATPG Algorithm

The basic goal of automatic test pattern generation (ATPG) was to decide the test pattern for given design under Test (DUT) which gives maximum fault coverage.

The current research in this field focuses on ATPG algorithm which not only gives maximum fault coverage but also ensures the maximum fault coverage at lowest possible power dissipation. In this category first of all Podem algorithm proposed by Wang and Gupta and the method of exploiting the redundancy by fault dropping proposed by Corno et al. has been covered in previous survey paper by P.Girard [2]. Few other methods are is also researched in this area. Polian I. et al [32]. propose a heuristic method to generate test sequences which create worst-case power drop by accumulating the high and low-frequency effects. The generated patterns need to be sequential even for scan designs. They employ a dynamically constrained version of the classical D-algorithm for test generation, i.e., the algorithm generates new constraints on-the- fly depending on previous assignments. In an another approach Ho Fai Ko, Nicolici N [14] suggested that scan chain division has been successfully used to control shift power by enabling mutually exclusive flip-flops at different times during the scan cycle. However, to control capture power without losing transition fault coverage during at-speed scan test, the existing automatic test pattern generation (ATPG) flows need to be modified. He presented a novel scan chain division algorithm that analyzes the signal dependencies and creates the circuit partitions such that both shift and capture power can be reduced when using the existing ATPG flows. Whereas Sying-Jyan Wang et al. [41] present a low capture power ATPG and a power-aware test compaction method. Two goals are achieved by the proposed ATPG. (1) The growth of test pattern count is lower than the detection number n. (2) The peak power becomes smaller as the detection number n increases. The test compaction algorithm further reduces the number of test patterns as well as the average capture power.

2.3.2 Ordering Techniques

The researches have widely explored the test vector reordering techniques to reduce the switching power. The earlier method based on Hamming distance based reordering proposed by P Girard and Debholkar are described in survey paper [27]. Girard's approach of vector ordering is enhanced by Paramasivam K, Gunavathi K, Sathishkumar P They have shown that the switching activity can be reduced up to 35 %[31]. In another method Roy S, Sen Gupta I, Pal A [34] proposed an AI-based approach to order the test vectors in an optimal manner to minimize switching activity during testing.

2.3.3 Input Control

Here the idea is to identify an input control pattern such that by applying the pattern to the primary inputs of the circuit during the scan operation, the switching activity in the combinational part can be minimized or even eliminated. Huang and Lee's basic idea of input control technique with existing vector- or latch-ordering techniques that reduces the power consumption has been covered in previous survey paper by P. Girard [27]. In the same area ElShoukry. M et al.[9] presented a technique of gating partial set of scan cells. The subset of scan cells is selected to give maximum reduction in test power within a given area constraint. An alternate formulation of the problem is to treat maximum permitted test power and area overhead as constraints and achieve a test power that is within these limits using the fewest number of gated scan cells, thereby leading to least impact in area overhead. The area overhead is predictable and closely corresponds to the average power reduction.

2.3.4 Vector Compaction and Data Compression

ATPG generated uncompacted test data contains a large number of don't care bits. There are number of test data compression techniques which explores the don't care filling options to optimize the compression of test data. The next generation compression scheme does not only aims to maximum compression but also explores the don't care bit filling to give minimum switching activity and hence power reduction. Static compaction techniques to control scan vector power dissipation proposed by R. Sankaralingam et al. and a technique of Combining Low-Power Scan Testing and Test Data Compression for System-on-a-Chip proposed by A. Chandra and K. Chakrabarty has been covered in previous survey paper by P Girard [27]. Few other methods are also suggested after that, for Vector Compaction which are based on Don't care filling. Among them Sying-Jyan Wang et al. [40] proposed an automatic test pattern generation (ATPG) scheme for low power launch-off-capture (LOC) transition test. Two techniques are explored in the proposed ATPG. A bidirectional X-filling and vector replacement scheme. Whereas Kundu S, Chattopadhyay S [19] have used a Genetic Algorithm based heuristic to fill the don't cares. Their approach produces an average percentage improvement in dynamic power and leakage power over 0- fill, 1-fill, and Minimum transition fill (MT-fill) algorithms for don't care filling . In an another approach Z. Chen et al. [8] proposed segment-based X-filling to reduce test power and keep the defect coverage. The scan chain configuration tries to cluster the scan flip-flops with common successors into one scan chain, in order to distribute the specified bits per pattern over a minimum number of chains. Based on the operation of a state machine, Jing-Ling Yang and Qiang Xu [17] elucidates a comprehensive frame for probability-based primary-input dominated X-filling methods to minimize the total weighted switching activity (WSA) during the scan capture operation. Experimental results demonstrate that the proposed approach significantly reduces both average and peak WSAs. Whereas Tapas M and Santanu C [22] describe the effect of don't care filling of the patterns generated via automated test pattern generators, to make the patterns consume lesser power. It presents a trade-off in the dynamic and static power consumption. The effect is expected to be more prominent for technologies beyond 100 nm.

2.3.5 Scan Chain Transformation

Here the scan architecture is designed in such a way that it maintains the test time, enables reuse of the conventional scan architecture's test patterns, and avoids decreasing the scan clock rate. Previously L. Whetsel's approach of , Adapting Scan Architectures for Low Power Operation and K.J.Lee's approach of Peak-Power Reduction for Multiple-Scan Circuits during Test Application has been covered in previous survey paper by P Girard [27]. In addition to that Sinanoglu, O, Orailoglu A [37] proposed a scan chain modification methodology that transforms the stimuli to be inserted to the scan chain through logic gate insertion between scan cells, reducing scan chain transitions . Based on this analysis, they developed an algorithms for transforming a set of test vectors into power optimal test stimuli through cost-effective scan chain modifications.

2.3.6 Clock Scheme Modification

Giving a clock to a whole circuit will give some unnecessary transition, which affects on power and energy consumption. Here the techniques are described which manages the clock distribution in such a way that overall it consumes less power. Previously Pouya and Crouch's ideas on optimization trade-offs for vector volume and test power , Sankaralingam et al approach on reducing power dissipation during test using Scan chain disable, Bonhomme et al's approach based on a gated clock scheme for low power scan testing of logic ics or embedded cores have are covered in previous survey paper by P Girard [27]. It is found that many STUMPS architectures found in current chip designs allow disabling of individual scan chains for debug and diagnosis, such feature can be used for reducing the power consumption during test. Here Imhof M.E et al. [16] presented an automated generation of a test plan that keeps fault coverage as well as test time, while significantly reducing the amount of wasted energy by disabling of individual scan chains for debug and diagnosis .

2.4 Low Power BIST Techniques

Various authors reported on techniques to cope with power problems during BIST. In the following, these techniques for low power BIST are presented.

2.4.1 Low Power Test Pattern Generators

In this category the BIST architecture is designed in such a way that it decreases the circuits overall activity so that power consumption reduces significantly. Previously BIST strategy, called dual-speed LFSR was suggested by Wang and Gupta, based on two different speed LFSRs. Also Corno et al proposed an approach on low power BIST via non-linear hybrid cellular automata . Then a modified clock scheme for a low power BIST test pattern generator was proposed by P Girard, In addition to that Zhang et al suggested a POWERTEST tool , which is a Tool for Energy Conscious Weighted Random Pattern Testing. Whereas Wang and Gupta proposed LT-RTPG: A New Test-Per-Scan BIST TPG for Low Heat Dissipation. These all have been covered in previous survey paper by P Girard [27]. After that Ahmed N et al. [2]. presented a new low power test pattern generator using a linear feedback shift register (LFSR), called LP-TPG, which inserts intermediate patterns between the random patterns to reduce the transitional activities of primary inputs which eventually reduces the switching activities inside the circuit under test, and hence, power consumption. Hiirevren and Levent [18] proposed a polynomial-time algorithm that converts the test pattern generation problem into combinatorial problem called Minimum Set Covering. Solutions to that give the low-power design topology for the test pattern sequence. Youbean K et al. [46] presents a new low power BIST TPG scheme. It uses a transition monitoring window (TMW) that comprised of a TMW block and a MUX. The proposed technique represses transitions of patterns using the k-value which is a standard that is obtained from the distribution of TMW to observe over transitive patterns causing high power dissipation in a scan chain. K.Gunavathil et al. [11] proposed TPG based on Read Only Memory (ROM) which is carefully designed to store the test vectors with minimum area over the conventional ROM. This reduces the number of CMOS transistors significantly when compared to that of LFSR/Counter TPG. Bin Z. et al [5] proposed approach to reconfigure the CUT's partial-acting-inputs into a short ring counter (RC), and keep the CUT's partial-freezing-inputs unchanged during testing. S.Wang [44] presents a low hardware overhead test pattern generator (TPG) for scan-based built-in self-test (BIST) that can reduce switching activity in circuits under test (CUTs) during BIST and also achieve very high fault coverage with reasonable lengths of test sequences. The proposed BIST TPG decreases transitions that occur at scan inputs during scan shift operations and hence reduces switching activity in the CUT. M.Nourani et al. [26] proposed low-transition linear feedback shift register (LT-LFSR) technique. Transitions are reduced in two dimensions: 1) between consecutive patterns and 2.) between consecutive bits. The proposed architecture increases the correlation among the patterns generated by LT-LFSR with negligible impact on test length. Bei Cao et al.[43] presented an efficient algorithm to synthesize a built-in TPG from low power deterministic test patterns without inserting any redundancy test vectors. The structure of TPG is based on the non-uniform cellular automata (CA). And the algorithm is based on the nearest neighborhood model, which can find an optimal non-uniform CA topology to generate given low power test patterns. Li-gang Hou et al. [10] proposed a low power dynamic LFSR (LDLFSR) circuit which achieves comparable performance with less power consumption. Typical LFSR, a DFLSR[I], a LDLFSR are compared on randomness property and inviolability property. Multi-layer perceptron neural networks are used to test these LFSRs' inviolability property. H.-T. Lin J.C.-M. Li. [20] presented ATPG technique, which simultaneously reduces capture and shift power during scan testing. This ATPG performs power reduction during dynamic test compaction so the test length overhead is very small. This method implements several novel techniques, such as parity back trace, confined propagation, dynamic controllability and post-fill test regeneration T.

2.4.2 LFSR Tunning

Here this category mainly emphasized on reduction energy consumption without modifying the fault coverage. Various BIST techniques are described here. Earlier Girard et al. address the problem of energy minimization during test application for BIST enabled circuits[27]. In a random testing environment, a significant amount of energy is wasted in the LFSR and in the CUT by useless patterns that do not contribute to fault dropping. In this work, a new built-in self-test scheme for scan-based circuits is proposed by Bhattacherya B.B. et al. [3] for reducing such energy consumption. A mapping logic is designed which modifies the state transitions of the LFSR such that only the useful vectors are generated according to a desired sequence. Experimental results on ISCAS-89 benchmark circuits reveal a significant amount of energy savings in the LFSR during random testing.

2.4.3 Vector Filtering BIST

Here main idea is to filter out some non detecting sequences so that over all switching can be reduced and hence power consumption. Previously Girard et al. proposed a test-vector-inhibiting technique to filter out some non detecting subsequences of a pseudorandom test set generated by an LFSR. His work was extended by Manich et al. by the filtering action to all the nondetecting subsequences. The authors use a decoding logic to store the first and last vectors of the nondetecting subsequences to be filtered and the same idea was implemented by Gerstendrfer and Wunderlich. These authors combine a pattern-filtering technique with Hertwig and Wunderlich's technique to avoid scan-path activity during scan shifting. These all have been covered in previous survey paper by P Girard [27]. After that S. Hatami et al. [13] proposed a scan cell architecture that decreases power consumption and the total consumed energy. In the method which is based on the data compression, the test vector set is divided into two repeated and unrepeated partitions. The repeated part, which is common among some of the vectors, is not changed during the new scan path where new test vector will be filled. As a result, the test vector is applied to the circuit under test in a fewer number of clock cycle, leading to a lower switching activity in the scan-path during test mode.

2.4.4 Circuit Partitioning

Main goal here is to partition the circuit in to sub circuit so that parallel testing can be achieved . Girard et al. propose a novel low-power BIST strategy based on circuit partitioning. This strategy partitions the original circuit into two structural sub circuits so that two different BIST sessions can successively test each sub circuit. This idea has been covered in previous survey paper by P Girard [27]. To address the power in the scan chain, Swarup B et al [4] propose an efficient scan partitioning technique that reduces both average and peak power in the scan chain during shift and functional cycles. Whereas Qiang Xu et al. [33] proposed a novel low-power virtual test partitioning technique where faults in the glue logic between subcircuits can be detected by patterns with low power dissipation that are applied at the entire circuit level, while the patterns with high power dissipation can be applied within a partitioned subcircuit without loss of fault coverage. Experimental results show that the proposed technique is very effective in reducing test power.

2.4.5 Low Power RAM Testing

Various RAM trasition reduction technique by reordering read and write access are described for low power consumption. Cheung and Gupta propose a methodology for low-power test of RAMs. The authors base their strategy on RAM transition reduction by reordering the read and write accesses and the address counting scheme. These measures decrease the energy consumption and keep test time the same, so they also minimize the average power[2]. The idea has been covered in previous survey paper by P Girard [27] A row bank-based precharge technique based on the divided wordline (DWL) architecture is proposed by Shyue-Kung Lu et al. [21] for low-power testing of embedded SRAMs. In low-power test mode, instead of precharging the entire memory array, only the current accessed row bank is precharged. This will result in significant power saving for the precharge circuitry. With the ever increasing number of memories embedded in a system-on-chip (SoC), power dissipation due to test has become a serious concern. Here Yuejian Wu et al. [47] proposed a novel low power memory BIST. Its effectiveness is evaluated on memories in 130 and 90 nm technologies. A significant power reduction can be achieved with virtually zero hardware overhead.

2.5 Low Power DFT Techniques

Apart from internal and external techniques here DFT techniques are described for low power testing. Here main goal is to reduce switching activity by adding some hardware during test. Xiaoming Yu and Miron Abramovici [45] introduce two design-for-testability (DFT) techniques based on clock partitioning and clock freezing to ease the test generation process for sequential circuits. In the first DFT technique, a circuit is mapped into overlapping pipelines by selectively freezing different sets of registers so that all feedback loops are temporarily cut. An opportunistic algorithm takes advantage of the pipeline structures and detects most faults using combinational techniques. This technique is feasible to circuits with no or only a few self-loops. In the second DFT technique, they use selective clock freezing to temporarily cut only the global feedback loops. These DFT techniques do not introduce any delay penalty into the data path, have small area overhead, allow for at-speed application of tests, and have low power consumption. Min-Hao Chiu Li, J.C.-M [23] presents a Jump scan technique (or J-scan) for low power testing. The J-scan shifts two bits of scan data per clock cycle so the scan clock frequency is halved without increasing the test time. The experimental data show that the proposed technique effectively reduces the test power by two thirds compared with the traditional MUX scan. The presented technique requires very few changes in the existing MUX-scan design for testability methodology and needs no extra computation. The penalties are area overhead and speed degradation.

Chapter 3

Reordering Techniques

Test power has become a serious problem with switching activity as well as in scan-based testing. It can lead to prohibitive test power in the process of test application. During the process of scan shifting, the states of the flip-flops are changing continually, which causes excessive switching activities. Test vector reordering for reducing switching power is one of the general goal of low power testing. Here two techniques Hamming distance based and AI-based approach is proposed to order the test vectors in an optimal manner to minimize switching activity during testing.

3.1 Hamming Distance based Reordering technique

Very Large Scale Integration (VLSI) design plays a significant role in the fabrication of modern Integrated Circuits(ICs) with smaller in size and with more features for any electronics systems. Energy consumption and power dissipation are the major concern in the VLSI design. Several factors have contributed to this trend. With the advent of portable devices, for example low energy consumption has become one of the major design goals in order to prolong battery life. Moreover the amount of energy a circuit consumes is directly reflected in its heat dissipation, however requires expensive packaging and cooling techniques which in turn increases system cost [1]. In addition, as power consumption increases, circuit reliability gets affected adversely due to electro-migration. This is applicable for both Design power and testing power. Testing is a process of checking the fabricated ICs for any incorrect behavior due to faults like logical fault, delay fault, fabrication faults[2], etc. Testing is done by generating and applying a set of binary vectors called test vectors to the input of the circuit. Fault is detected by verifying the output for the given test vector with stored responses. Testing has to be done for all possible faults in the circuit. Hence more test vectors may be required to test all the faults in complete circuit. Single test vector may detect more than one fault and more than one test vectors can be generated for a single fault. A set of test vectors[4] must be generated such that more faults are covered with minimum number of test vectors. Testing is not only done after the fabrication of ICs but also required when ICs are in usage. This is called periodic testing which is required for all type of systems like PC, Laptop, cell phones etc. The main problem under testing environment is that it results in considerably higher circuit activity rate compared to normal mode operation, hence causing above normal power dissipation. However if test vector sets are not optimized for power[10], low power circuits dissipate two fold power under test as they do at normal operating condition[2]. When the circuit is tested with pseudo-random patterns, consecutive input test vectors are statistically independent which results in increased switching activity in the circuit under test. Since in CMOS circuits energy is primarily consumed by signal transition, the average power consumption during testing is significantly higher than normal mode of operation. The Relationship between hamming distance [6] and the average power of a circuit plays a significant role to optimize the test power. In order to optimize, the hamming distance between successive test vectors is used to arrange the test vectors in specific order so that Total Hamming Distance(THD) is minimum. A test vector set with least hamming distance is obtained by optimization technique. The test power obtained by applying test patterns in the optimal order is regarded as the optimized test power. In order to guarantee the proper operating conditions during test, the total power consumption must not exceed the maximum power allowance for the circuit under test[2]. Another problem is that even if the average power dissipation over a sequence of test vectors is small, the peak (or instantaneous) power dissipation may be sufficiently high to destroy the CUT. In practice, destruction really occurs when the instantaneous power exceeds the maximum power allowance during several successive test vectors. For this reason, it is essential to take care of both average and peak power dissipation during test application.

Many low power design techniques have been proposed at all levels of the design hierarchy. However, all these techniques focus on low power dissipation during system mode or standby mode, and during test mode. The simplest way to ensure non-destructive testing of a CUT is to use ordering of test vectors, which causes switching activity that is comparable to that during normal circuit operation. Several categories of techniques can be found in the literature related to the low power testing. The first category consists of ATPG (Automatic Test Pattern Generator) techniques[5], in which new ATPGs are proposed with the intent of generating test patterns that can reduce the power dissipated during test application in addition to the normal ATPG objectives. The second category consists of ordering techniques in which the switching activity is reduced by modifying the order in which test vectors of a given test sequence are applied to the CUT. The paper [14] discussed about two methods used for reordering of test vectors in order to reduce the dynamic power dissipation during testing of combinational circuits. Two search methods 2-opt heuristic and a genetic algorithm based approach have applied and results obtained for combinational circuits. These techniques can be applied during external testing or deterministic BIST (Built-In Self Test). In this paper we present a novel method, which aims at minimizing switching activity during testing of combinational circuit. The test vectors are reordered based on the hamming distance between successive test vectors. Graph theory based reordering algorithm is proposed to solve the problem. Since this problem is NP complete, the algorithm is developed through heuristic approach which gives better optimum solution for such problems. Random heuristics are used in previous approaches that may or may not give better solution. In this proposed algorithm structured heuristic is used to obtain better results than random heuristics.

3.1.1 Problem formulation

The power dissipation during testing [2] is minimized by reducing the number of transition in the circuit. This is achieved by reducing the hamming distance between successive test vectors. Usually test vectors are in random and hence it is necessary to rearrange the order of occurrence of test vectors so that the hamming distance between successive test vectors is minimum. In general the total switching power in the whole circuit is proportional to the hamming distance of input test vectors. Therefore the reordered test vector set with minimum hamming distance is used for testing the CUT to reduce the switching power. The problem of minimizing switching power is solved by graph theory using Hamiltonian path [3] technique. Graph G(V,E) is defined with V nodes and E edges. The problem is formulated by considering the test vector as node and hamming distance between them as edge cost of the graph. Here the Hamiltonian path is a path with all nodes and minimum total edge cost. Graph Theory based Reordering algorithm[12] is used to construct the Hamiltonian path, which is resultant reordered test vector set whose total hamming distance is minimum. Now the path developed by the algorithm is reordered test vector sequence which offers less number of transitions at the input which in turn results in reduced power dissipation in the circuit under test during testing [3].Heuristic approach is used in the algorithm to find more suboptimal sequences.

3.1.2 Total Hamming Distance(THD)

Total hamming distance is defined as Sum of hamming distance between successive test vectors in the sequence.

Let hamming distance d[ti, tj] be the total number of changes between i thth and j test vector. The Total Hamming Distance(THD) for the whole test vector set is calculated by the following relation.

$$THD = \sum_{i=1} d[t_i, t_{i+1}] \tag{3.1}$$

where n represents total number of test vectors in the whole set. The overall procedure to minimize the switching activity during testing is as follows.

a. Consider a digital circuit with p inputs and q outputs.

b. Generate all the test vectors to detect all the single stuck at faults[4] of the circuit. Let the number of test vectors be n.

c. Find the hamming distance between each and every test vector and load the same in array hd of size n x n. Let hd[i][j] be the array elements which gives hamming distance between i^{th} and j^{th} test vectors.

d. Apply reordering algorithm to find the reordered test vector sequence with minimum total hamming distance.

e. Perform fault simulation[4] with reordered test vector set which gives minimum number of transition and hence less power dissipation.

f. Since Heuristic based algorithm generates more sub-optional sequences, select the best sequence with least total switch activity.

The reordering algorithm used in step d is discussed in the next sub-section.

3.1.3 Reordering Algorithm:

The various parameters used in the algorithms are as follows: t1, t2,....tn be n test vectors with m bits each. T={1,2,.....k,...n } where k represents kth position in the vector set generated by ATPG.

R is a set to store ordered test vector sequence.

Q is a set to store T-R.

Step 1: Select a test vector x such that swa_init[x] is minimum in the array swa_init[ ]. Add x to set R.

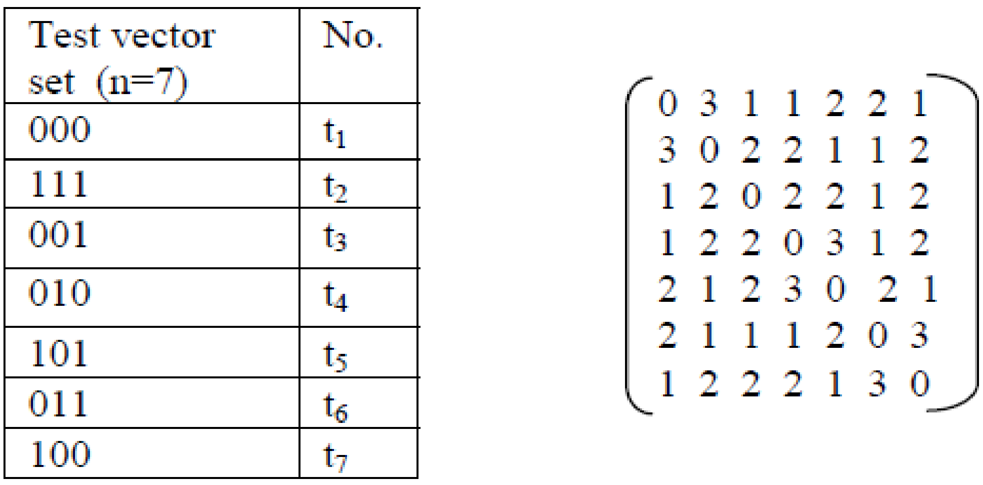
Step 2: Select a test vector y such that hd[x][y_{min}] is minimum in the array.

Step 3: Add y_{min} to R; Q←T-R; $x_{min} \leftarrow y_{min}$.

Step 4: From the array hd[x_{min}][j] when j varies as in Q, find y so that hd[x][y_{min}] is the smallest value. Go to step 3.

Step 5: In the step 4, if hd[x_{min}][j] has more than one smallest value, then such number of reordered sequence will be generated for every x_{min}. These sequences are called as sub-optimal sequences.

Finally the set R will have reordered test vector sequence with minimum hamming distance which results in minimum switching activity during testing.



Test vector set (n=7)	No.
000	t_1
111	t_2
001	t_3
010	t_4
101	t_5
011	t_6
100	t_7

$$\begin{pmatrix} 0 & 3 & 1 & 1 & 2 & 2 & 1 \\ 3 & 0 & 2 & 2 & 1 & 1 & 2 \\ 1 & 2 & 0 & 2 & 2 & 1 & 2 \\ 1 & 2 & 2 & 0 & 3 & 1 & 2 \\ 2 & 1 & 2 & 3 & 0 & 2 & 1 \\ 2 & 1 & 1 & 1 & 2 & 0 & 3 \\ 1 & 2 & 2 & 2 & 1 & 3 & 0 \end{pmatrix}$$

Figure 3.1: a.)Test Vectors for Full adder circuit b.) Switching matrix hd[n][n]

The above procedure is illustrated with simple full adder circuit with 3 inputs and 2 outputs. The test vector set that used to detect the entire single stuck at faults is given in Table I. The set consists of 7 vectors with total hamming distance as 15 . The vectors are represented by the order of occurrence for the sake of convenience. The hamming distance array hd[ ][ ] of order n X n is constructed. This is given as in fig 3.1. It is known that the number of bit changes between t1 and t2 test vectors in the table is three. Similarly between t2 and t3 test vectors is two. This is shown in the array as hd[1][2]=3 and hd[2][3]=2 respectively. The hamming distance array hd[ ][ ] is developed in this method.

On application of reordering algorithm to this matrix hd[ ] [ ], the reordered test vectors are generated with minimum hamming distance.

The solution and the THD are given as follows:

Unordered sequence : t1 t2 t3 t4 t5 t6- t7 THD : 15

Ordered sequence : t3 t1 t4 t6 t2 t5 t7 THD : 6

The total hamming distance for the resultant ordered sequence is 6 which shows that 60 % of total hamming distance is reduced when compared with that of unordered sequence. This reduces the switching activity and hence the power dissipation in the circuit.

3.2 Artificial Intelligence approach

3.2.1 Brief overview

Advancements in semiconductor fabrication technology has helped the design engineers to accommodate more number of transistors in a VLSI chip. With the proliferation of mobile battery-powered devices, reduction of power in the embedded VLSI chips has become an active area of research. During the last decade, power reduction techniques have been proposed at all levels of the design hierarchy - from system to device levels. For the development of complex, high performance, low power devices implemented in deep submicron technology, power management is a critical parameter and it cannot be ignored even during testing. With the increase in the density of the chips, the problem of testing has also increased manifold.

A related problem is to achieve power reduction during the actual testing of a chip [25]. Power consumption in test mode is considerably higher than the normal functional mode of a chip. The reason is that test patterns cause as many nodes switching as possible, while a power saving system mode only activates a few modules at a time. Thus, during testing switching activity in all the internal lines of a chip is often several times higher than during normal operation. Sometimes parallel testing is used in SoCs to reduce test application time, which results in excessive power dissipation. Again, successive functional input vectors applied to a given circuit during system mode have a significant correlation, while the correlation between consecutive test patterns can be very low. Usually, there is no definite correlation between the successive test patterns generated by an ATPG (for external testing) or by an LFSR (for BIST) for testing of a circuit. This can cause significantly larger switching activity in the circuit during testing than that during its normal operation. Abnormal power consumption during testing leads to adverse effects on the chip and the testing process such as, (a) this may give rise to severe hazards to the circuit reliability and lead to long or short-term malfunction, (b) it can cause chip destruction due to excessive heat in absence of proper heat dissipation mechanism, (c) it can increase the packaging and cooling costs, (d) it can cause the chip to falsely fail the test due to noise problems such as, IR and Ldi/ dt drops, which may be a source of yield loss and increase in production cost, (e) it may make it difficult to obtain a carefully tested bare die to be used in multichip modules (MCM) or what is called the Known Good Die problem (KGD) and (f) it can dramatically shorten the battery life when on-line testing is considered. For all these reasons, various techniques have been proposed to reduce the impact of high power consumption during test application. Low power dissipation during test application is becoming an equally important figure of merit in today's VLSI circuits design with BIST and is expected to become one of the major objectives in the near future.

3.2.2 Prior Work

A survey on low power testing of VLSI circuits has been given in [27]. There are several techniques for low power testing of VLSI circuits such as, test vector reordering, scan chain ordering, power-constrained test scheduling, use of multiple scan chains, low power test pattern generation, vector compaction, etc. Test vector reordering is a very well-known technique to reduce dynamic power dissipation during combinational circuit testing through switching activity minimization in the circuit. Test vector reordering is an essential task in testing VLSI systems because it affects from two perspectives: power consumption and correlation among data used for test data compression. The problem of test vector reordering can be mapped into finding Hamiltonian cycle in a complete weighted graph, which is known to be NP-hard. So, there is no polynomial time solvable algorithm for the problem. Therefore, it is essential to find a good heuristic-based solution for the problem. Existing approaches have used several heuristics for solving this problem. In [19], the problem of test vector reordering has been mapped into finding the Hamiltonian path in a fully-connected weighted graph which is similar to the traveling salesman problem (TSP). As there exists no polynomial time algorithm for TSP, approximation methods of solution have been used. Solutions for three cases have been given: (i) reordering for maximal or minimal activity, (ii) reordering of test vectors with a desired circuit activity across the VLSI chip while achieving a high coverage for stuck-at faults, and (iii) reordering for localized switching activities to maximize it in one part and minimize at other part of the circuit. In another work [30], proposed greedy algorithm has guaranteed decrease in power consumption without modifying the initial fault coverage. A second technique based on Simulated Annealing (SA) has been proposed in which the greedy solution is used as initial solution and it shows a considerable average power reduction during test application. Here, only Hamming distance between test vectors has been used to avoid simulation of the circuit and providing a solution (an Hamiltonian path of minimum cost in the Hamming distance graph) in a short computation time. It has been shown that there is a correlation between the Hamming distance and the transition activity. But, if signal transitions in the internal line be considered, then obviously optimal solution can be found.

Another work [28] has also considered the Hamming distance minimization between adjacent vectors to reduce the dynamic power dissipation during testing. In [42], reduction of power dissipation during test application has been studied both for scan designs and for combinational circuits tested using built-in self-test (BIST). They have shown that heuristics with good performance bounds can be derived for combinational circuits tested using BIST and a post ATPG phase has been proposed for reducing power dissipation during test application in full-scan circuits and for pure combinational circuits. They have shown that scan-latch ordering along with test-vector reordering can give considerable improvement in power dissipation and considerable savings can be obtained by repeating some of the test vectors. In [12], an evaluation of different heuristic approaches has been done in terms of execution time and quality. Here, it has been shown that the Multi-Fragment heuristic performs better than Christofides and Lin-Kernighan heuristics in terms of time. It also outperforms the Christofides heuristic in terms of quality and achieves performance very close to Lin-Kernighan. They recommended reordering algorithms to use the Multi-Fragment heuristic for near-minimal ordered sets of vectors that result in both reduced power consumption and enhanced data compression ratio. Recently, some works have formulated test vector reordering problem as TSP and Genetic Algorithm (GA) has been used to generate low power test patterns.

Chattapadhyay and Choudhary have shown proposed a GA-based formulation to solve the problem of generating a test pattern set such that it has high fault coverage and low power consumption has been proposed in [2]. They have shown a method of selecting a subset of test vectors generated by an ATPG tool to reduce power dissipation by sacri?cing a small amount of fault coverage. In [1], authors have studied two well known search methods (2-opt heuristic and a GA-based approach) with reduction in fault coverage. They have also combined those two methods for power reduction.

3.2.3 Motivation of the work and problem formulation

Motivation of the work

Most popular techniques for test power minimization orders the deterministic test patterns and several approaches have been followed for test vector reordering such as, ?nding minimum cost Hamiltonian path after mapping the problem into TSP instance, finding optimal solution by applying GA or SA. Although the dynamic power minimization problem by test vector reordering during VLSI testing is an old problem, here we have proposed a new approach for solving it using Artificial Intelligence (AI). This problem can again be viewed as finding optimal path from start to goal node in a search space by applying informed search methods of AI, where start node is the node when no test vector is selected and the goal node is the node when only one test vector is remaining for selection. A* search algorithm is a very well-known informed search method used in AI. It takes advantages of both ef?ciency of greedy search and optimality of uniform-cost search by simply summing the two evaluation functions. Thus, it is optimally ef?cient algorithm for finding optimal solution in an informed search space. This has motivated us to apply A* search technique for test vector reordering problem for dynamic power reduction during testing.

Problem formulation

Consider a test set for a combinational circuit is given by V = {v1,v2,...vk } with a predefined fault coverage, where |V | =k. Each test vector is formed by a fixed ordered set of bits b_j i.e., v_i =< b1,b2,...bl> where l =length of the test vectors or the number of primary inputs (PIs) of the circuit. Assume ? be the initial ordering of test vectors V . The problem of dynamic power minimization by test vector reordering is to compute an optimal vector ordering of V such that total dynamic power dissipation in the circuit during testing is minimized. The problem of reducing the peak power dissipation is not considered here. Only the average power reduction has been considered. Since, the power dissipation is directly proportional to switching activity, the problem can be restated as to ?nd out an optimal path or optimal ordering of vertices $\prod'$ from the search space of having all possible orderings of vectors V such that total switching activity in the circuit is minimized.

3.2.4 An A* based method for dynamic power minimization by test vector reordering

Basic underlying principle for A* Algorithm

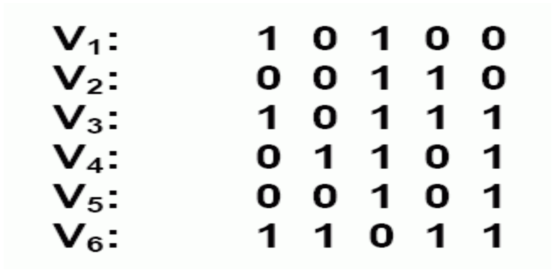
The A* algorithm combines features of uniform-cost search and pure heuristic search to efficiently compute optimal solutions. A* is a best-first search in which the cost associated with a node is given by the evaluation function f (n).

$$f(n) = g(n) + h(n) \tag{3.2}$$

where g(n) is the cost of the path from the initial state to node n, and h(n) is the heuristic estimate of the cost of a path from node n to a goal node. Thus, f (n) estimates the lowest total cost of any solution path going through node n. At each point, a node with lowest f -value is chosen for expansion. Ties among nodes of equal f -value is broken in favor of nodes with lower h-values. The algorithm terminates when a goal node is chosen for expansion. For a given node, the sum [current cost + heuristic value] is an estimation of the cost of reaching the ending node from the starting node, passing by the current one. This value is used to continuously choose the most promising path. In practice, the algorithm maintains two lists of nodes that are filled and modified during the search: an OPEN list and a CLOSED list. OPEN list is a priority queue, contains the tracks leading to nodes that can be explored in increasing order of the evaluation function f (n). Initially, there is only the starting node and at each step, the best node of OPEN list is taken out. Then, the best successor of this node (according to the heuristic) is added to the list as a new track. The CLOSED list stores the tracks leading to nodes that have already been explored.

3.2.5 Cost function g(n)

A complete weighted graph T (V,E,W) is constructed where V is the test set, E is the set of edges and



V_1:	1	0	1	0	0
V_2:	0	0	1	1	0
V_3:	1	0	1	1	1
V_4:	0	1	1	0	1
V_5:	0	0	1	0	1
V_6:	1	1	0	1	1

Figure 3.2: Test Vectors of C17

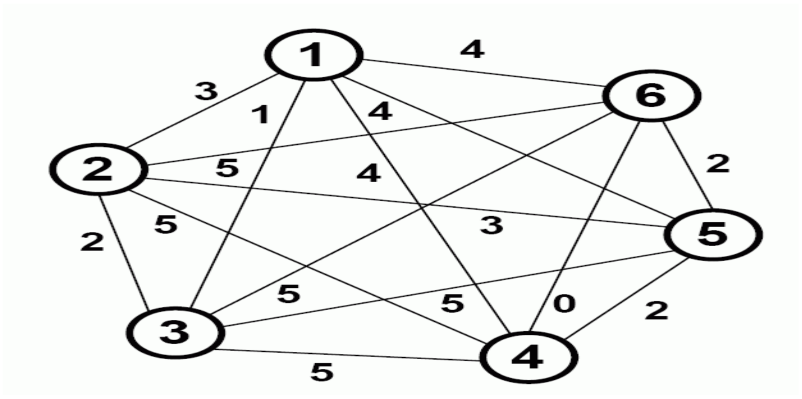


Figure 3.3: Transition graph of C17

W resents the set of weights associated with each edge indicat-ing the total number of signal transitions (i.e., total switch-ing activity) in the whole circuit considering internal lines for consecutive application of two adjacent test vectors. So, T represents a matrix of total transitions and it is called as Transition graph or T -graph. For example, Fig. 3.3 represents the T -graph for the test vectors shown in Fig. 3.2 for C17 benchmark circuit of ISCAS85. The cost function g(ni) of a successor ni of the node n is calculated by g(ni)= g(n)+ W(n,ni),where W -value is taken from the T -graph.

3.2.6 Computing lower bound of switching activity: Heuristic function h(n))

In the A*-based algorithm, lower bound of switching activity among the remaining test vectors is taken as the heuristic function h(n) for a node n. Lower bound of signal transition is computed by bit-wise scanning the test vectors to be selected to reach the Goal node. If for ith bit of all the test vectors are 1 or 0, then the lower bound of switching activity for ith bit is 0; otherwise, it is taken as 1. So, the maximum value of the lower bound is equal to the length of the test vectors, l.

Theorem 1. The lower bound of switching activity for the remaining test vectors is given by the sum of the lower bounds for all the bits.

An heuristic h is called an admissible heuristic, if it never overestimates the cost to reach the goal. The above lower bound of signal transition is computed using a procedure f ind lower bound() and it never overestimates the actual cost of reaching the Goal node. Thus, it is an admissible heuristic. Some examples of calculating h() value has been shown below for the test vectors shown in fig 3.2

v2 : 00110

v3 : 10111

v4 : 01101

v5 : 00101

v6 : 11011

....

Lb(bitwise) 11111 =5

a.) For the remaining five test vector ¡v2,v3,v4,v5,v6¿ the lower bound h()=5

v3 : 10111

v4 : 01101

v5 : 00101

v6 : 11011

....

Lb(bitwise) 11110 =4

b.) For the remaining four test vector ¡v3,v4,v5,v6¿ the lower bound h()=5

v4 : 01101

v5 : 00101

....

Lb(bitwise) 01000 =1

c.) For the remaining two test vector ¡v4,v5¿ the lower bound h()=1

3.2.7 The A* based algorithm:AITVR

.... Algorithm 1 AITVR() ....

```
1. Find the T matrix for the test set V
2. G(Source)=0;
3. H(source)=0;
4. F(source)=0;
5. Put Source in OPEN; found=False
6. while OPEN not empty and not(Found) do
7. Select a node n from OPEN with minimum f -value
8. Remove n from OPEN and put n in CLOSED
9. if n is a Goal node then
10. Found = True
11. else
12. Get the list of test vectors selected so far from Source to the node n. Let, this is $V_s$'
14. Get the list ($V_s$') of test vectors not selected so far from Source to the node n
15. Expand n (* consider each successor test vector to the next level in $V_s$' *)
16. for each immediate successor $n_i$ of n do
17. g(ni) = g(n)+W(n,ni)
18. Find $V_s$' for $n_i$ i.e., $V_s$' ($n_i$)
```

```
19. $h(n_i)$ = find lower bound($V_s$' ($n_i$))
20. f ($n_i$) = $g(n_i)+h(n_i)$
21. Put $n_i$ in OPEN
22. Direct backward pointer from $n_i$ to n
23. end for
24. end if
25. end while
26. Find the remaining test vector $v_{rem}$ to be selected; $v_{last}$ =last test vector selected in the Goal node n
27. Find $SW_{last}$ =W($v_{last}$ ,$v_{rem}$)
28. Calculate total switching activity SW = f(n)+$SW_{last}$
29. Output SW and solution path $\prod'$ (* sequence of test vectors selected * ) by tracing back pointers
.............................................................................................
```

3.2.8 Time complexity of AITVR

Complexity of A*-based algorithm is obtained from its empirical performance. However, time complexities can be estimated for the heuristic computation, and node expansion. The worst-case complexity for computing g() for each node is $O(n^2)$. If l be the length of the test vectors, then the time complexity for calculating h()-value for each node expanded is O(ln), in worst case. Thus, a single node generation in AITVR requires time $O(n^2)+O(ln)$.

3.2.9 Empirical Observation

The proposed algorithm AITVR was implemented in MATLAB on an Windows platform with an Intel Pentium IV proces- sor of 1 GHz clock speed and 1 GB RAM. For evaluation of the proposed algorithm, S27 benchmark circuits were considered. First

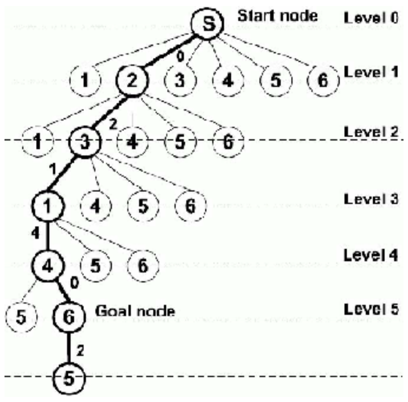


Figure 3.4: Finding optimal path from informed search space

test patterns were generated by the Mentor Graphics. DFT Advisor and Fast Scan are used from Mentor Graphics to generate patterns.

Mentor Graphics ATPG tool with 100 % fault coverage and those were taken as initial ordering of the test set. In this work, we have reduced dynamic power consumption during test application without losing stuck-at fault coverage. A simulation-based technique has been used to obtain the actual switching activity at the internal nodes of a circuit using unit-delay model and the total Weighted Transition Matrix was formed. The characteristics of the test patterns obtained from TetraMax are tabulated in Table 1. Experimental result shows that the dynamic power of a given combinational circuit can be reduced by about 56.52% after applying the test vectors in the order given by AITVR over the test vectors order given by Mentor Graphics, where as Hamming distance method provides 21.97% reduction in switching activity on a average for all the benchmarks.

The proposed algorithms finds the optimal ordering of test vectors from the search space as shown by the path {2,3,1,4,6,5} from Start node to Goal node in Fig. 3.4.

3.3 WTM based Reordering of Combine Test Vector and Output Response

3.3.1 Problem formulation

Consider a test set for a combinational circuit is given by T = {t1,t2, ..tk} and its output response is given by O={o1,o2,.ok} with a predefined fault coverage, where | T | = k. Each test vector is formed by a fixed ordered set of bits bj i.e., ti ={ b1,b2, ...bl }, where l =length of the test vectors or the number of primary inputs (PIs) of the circuit. Assume $\prod$ be the initial ordering of test vectors T.

The problem of scan in scan out power minimization by test vector reordering is to compute an optimal vector ordering of T such that total scan in scan out power dissipation in the circuit during testing is minimized. The problem of reducing the peak power dissipation is not considered here. Only the average power reduction has been considered. Since, the power dissipation is directly proportional to switching activity, the problem can be restated as to find out an optimal path or optimal ordering of vertices from the search space of having all possible orderings of vectors V such that total switching activity in the circuit is minimized.

3.3.2 Cost Function g(n)

A complete weighted graph X(T,O,E,W) is constructed where T is the test set, O is the output response, E is the set of edges.

Weighted Transition Matrix(WTM) is used here to reorder the test patterns. Here different test patterns passed through the output response and hence the switching will occur and from that WTM is made as shown in figure 1 c. The corresponding equation is also shown below.

$$WTM = \sum_{j=1}^{n} \sum_{j=1}^{n} t(j,i) XORO(j,i) * (n-i). \tag{3.3}$$

t1: 010	O1: 100
t2: 001	O2: 110
t3: 110	O3: 100
t4: 111	O4: 111
t5: 010	O5: 000
t6: 010	O6: 011
t7: 100	O7: 000
t8: 100	O8: 111

Figure 3.5: a.) Test Vector b.)Output Response

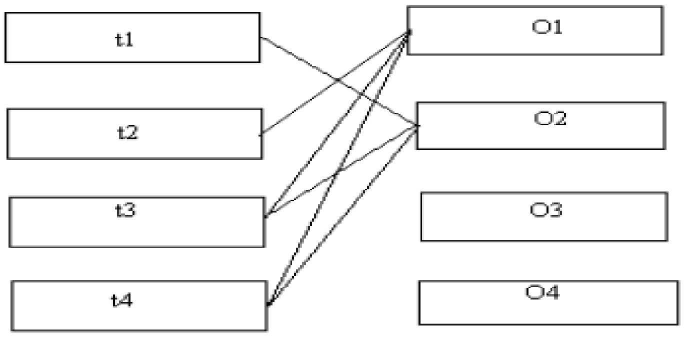


Figure 3.6: Test Vector passing through Output Response

WTM =

$$\begin{pmatrix} 0 & 2 & 3 & 2 & 4 & 4 & 4 & 4 \\ 5 & 0 & 2 & 1 & 5 & 5 & 1 & 3 \\ 4 & 2 & 0 & 2 & 4 & 4 & 4 & 1 \\ 6 & 4 & 1 & 0 & 6 & 1 & 2 & 2 \\ 3 & 1 & 4 & 3 & 0 & 3 & 5 & 5 \\ 5 & 3 & 2 & 1 & 5 & 0 & 3 & 3 \\ 3 & 1 & 4 & 3 & 3 & 3 & 0 & 5 \\ 6 & 4 & 1 & 0 & 6 & 6 & 2 & 0 \end{pmatrix}$$

and W represents the set of weights associated with each edge indicating the total number of signal transitions (i.e., total switching activity) in the whole circuit considering internal lines for consecutive application of two adjacent test vectors. So, WTM (Weighted transition matrix) represents a matrix of total transitions and it is called as Transition graph or T-graph. For example, Fig. 3.6 represents the T-graph for the test vectors shown in Fig. 3.5(a),(b) for S27 benchmark circuit of ISCAS85. The cost function g(ni) of a successor ni of the node n is calculated by g(ni) = g(n)+W(n,ni), where W-value is taken from the T-graph.

The proposed algorithms finds the optimal ordering of test vectors from the search space as shown by the path < 5,2,7,1,3,8,4,6 > from Start node to Goal node in Fig. 3.7.

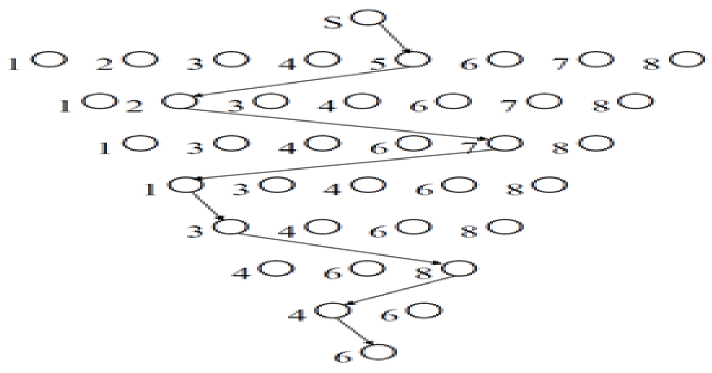


Figure 3.7: Finding optimal path in informed search space

3.3.3 Empirical observation

The proposed algorithm AITVR was implemented in MATLAB on an Windows platform with an Intel Pentium IV proces- sor of 1 GHz clock speed and 1 GB RAM. For evaluation of the proposed algorithm, S27 benchmark circuits were considered. First test patterns were generated by the Mentor Graphics. DFT Advisor and Fast Scan are used from Mentor Graphics to generate patterns.

Mentor Graphics ATPG tool with 100% fault coverage and those were taken as initial ordering of the test set. In this work, we have reduced dynamic power consumption during test application without losing stuck-at fault coverage. A simulation-based technique has been used to obtain the actual switching activity at the internal nodes of a circuit using unit-delay model and the total Weighted Transition Matrix was formed.

Experimental result shows that the dynamic power of a given combinational circuit can be reduced by about 22.11% after applying the test vectors in the order given by AITVR over the test vectors order given by Mentor Graphics, where as Hamming distance method provides 17.39% reduction in switching activity on a average for all the benchmarks. These are shown in table I.

Table I: Comparison of Switching activity in Hamming and AI

Sr. No	ISCAS circuit	No.of Test Patterns	Hamming	% reduction	Artificial Intelligence	% reduction
1	S27	23	19	17.39	10	56.52
2	S298	1595	1631	0	1328	16.73
3	S344	1475	1517	0	1244	15.66
4	S349	1602	1615	0	1312	18.10
5	S382	3951	4008	0	3595	9.01
6	S386	402	389	3.23	250	37.8
7	S400	4124	4139	0	3439	16.61
8	S420	3239	3241	0	2642	18.43
9	S444	4218	4187	0	3585	15.00
10	S510	754	697	7.55	536	28.9
11	S526	4523	4510	0.2	3969	12.24
12	S641	3181	3185	0	2674	15.93
13	S713	2590	2626	0	2165	16.40
14	S820	857	794	7.3	616	28.12
15	S832	886	815	8.01	653	26.29
					Avg	22.11

Chapter 4

Bit Filling Algorithm

4.1 MT Fill Algorithm

The different techniques to reduce the dynamic power are: test vector ordering [29] [30][39][42], scan-cell ordering [35][15][42], low power ATPG [36] etc. The don't care bits of the ATPG generated test patterns are often filled judiciously to reduce dynamic power. Another way to reduce the dynamic power is to guard the circuit against the scan-transitions. As it has been pointed out in [29][30][38], the transitions occurring due to shifting-in the test patterns and shifting-out the responses is a major source of dynamic power consumption, as it introduces a large number of ripples in the input to the circuit[30].The don't care bits (X's) of test patterns are filled with specific values (adjacent, 0 or 1) so as to minimize the occurrence of such transitions and thus the dynamic power during testing. Compared to other solutions, such a filling technique has the advantage to be applicable after the end of the design process and does not require any modifications to the circuit. Classical non-random filling heuristics are discussed in [24]. These are MT-filling (Minimum Transition filling), 0-filling, and 1-filling. For example, consider the test pattern 0X1X1XX0XX0XX. If we apply each of the three nonrandom filling heuristics, the resulting pattern will be:

o 0011111000000 with MT-filling.

o 0010100000000 with 0-filling.

o 0111111011011 with 1-filling.

Here in MT fill algorithm on either side of don't care if there is '1' than don't care is filled up with '1'otherwise '0'. Whereas in '0' fill all don't cares are filled up with '0's. and in 1-fill all don't cares are filled up with '1's. As it can be seen that with MT Fill only two switchings are there whereas with 0 fill and 1 fill there are four and five switchings respectively. So MT Fill is the optimum method to reduce power.
Estimation of switching activity with artificial intelligence(AI) approach and compression with selective Huffman approach for various benchmark circuits are shown in table II and III respectively.

4.2 Frequency Directed Bit filling Algorithm

For the statistical codes, test data is divided into equal size blocks of B bits. To improve the test data compression, the no. of distinct blocks in a given test set should be reduced and frequency of occurrence for each distinct block should be increased. For coding process, for each distinct block, the corresponding frequency of occurrence is calculated. The Hamming distance of block B_1 with highest frequency of occurrence will be calculated from the B_2 with the second highest frequency. The Hamming distance is 1 if the bits on the same position of two blocks are opposite, i.e. "1" and "0".
The Hamming distance between two blocks is summation of such bits with opposite values. The Hamming distance between 10X1 and 010X is 2 as its first and second bits have opposite values. If the Hamming distance between B_1 and B_2 is more than 0, the Hamming distance with next block with descending order of frequency will be calculated. Two blocks for which the Hamming distance is 0, will be merged and a new block M_1 will come into existence. The next block in the sequence will be than compared with merged block M_1.This process is repeated until further merging is not possible. The process is repeated with the next highest frequently occurring

still unmerged block. The merging has increased the number of specified bits. Still there is a chance that few bits are unspecified. Such bits are replaced with zeroes. Let's understand the concept with one example

Table I: Test data set for frequency directed bit filling

X	0	1	1	1	X	X	X	X	0	1	0	0	1	X	X
1	1	0	1	1	1	X	X	0	0	0	0	1	0	X	X
1	1	0	1	X	1	X	X	1	1	0	X	1	1	X	X
0	1	0	1	1	0	X	X	X	0	1	X	1	0	X	X

Consider the test data set with total 62 bits shown in table I. Here the bits size b =4. To make the last block of size b, at the end of test set two don?t care bits are appended.

Here the unique vectors are {10XX, 11XX, 1101, 01XX, X011, 1XXX, 0000, X010, X1XX, 110X, 0101, X01X} with the corresponding frequencies {3, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1}. Starting with B1: 10XX. The Hamming distance of B1 from B2, B3, B4 is 1, 1, 2 respectively but with B5, it is 0. So B1 will be merged with B5. 10XX and X011 will make a merged block 1011 and frequency of this merged block is sum of the individual block i.e. 4. This merged block M1 will be further compared with B6 to B12. B6 and B12 will be merged with M1. After one cycle of merging the merged block 1101 has frequency 6. The next cycle of merging will start with B2 as it is still unmerged. The same process will continue with all unmerged blocks. For given example, the merged symbols are {1011, 1101, 0101, 0000, X010} with corresponding frequencies 6, 6, 2, 1, 1}. The last merged symbol X010 still contains a dont care bit which will be replaced by 0 and the merged symbol will be 0010.

4.2.1 For Optimization of Compression and Power Both

For filling the bits such that it improves power and compression both together, we will have to find out the locations of bits which incorporate to compression heavily. The same way the locations of the bits which incorporate to scan power heavily are also necessary to know. Considering to test data compression, all the bits in test vectors have equal weight. But for scan power, it is well known that the bit location in scan-in and scan-out vectors is weighted in term of contribution to power.

For that, lets explore this concept.

Lets consider a sequence of scan-in and scan out like this.

1. A scan-in vector t1 is inserted to scan register which cause WTMload number of transitions to occur. t1 scan-in vector generated the O1 output response. Here n is number of bits in each test vector.

2. During scan-out of O1 from scan register, WTMunload transitions are occurred.

3. Now suppose that, the next scan-in vector selected is t3

4. So while O1 is being scanned out, t3 is being scanned-in. The WTM$_{Load}$ for the loading operation is given for t1 pattern containing n bits by equation

$$WTM_{LOADt1} = \sum_{i=1}^{n} (t1(i) XOR t1(i+1)) * (n-i) \tag{4.1}$$

Similarly for the unloading operation of O1, WTMUnload can be calculated as given in equation

$$WTM_{unloadoi} = \sum_{i=1}^{n} (O1(i) XOR O1(i+1)) * i \tag{4.2}$$

The sequence of unloading O1 scan-out pattern is in parallel with loading of t3 scan-in pattern and so on.

So total WTM between t1-t3 which is denoted as WTM1-3 is calculated as below:

$$WTM_{1-3=} WTM_{UNLOADo1} + WTM_{LOADt3} + (t3(1) XOR O1(n)) * i \tag{4.3}$$

In equation the last term considers the mismatch between the last bit of scan-out O1 vector and first bit of t3 scan-in vector. If these two bits have different values, it will add total n transitions.

Now as we are discussing, the bit filling methods, we will consider WTM load and WTMi-j for given test vector. If {B1, B2,. . . . . . .Bn} are the bits of vector t1, as shown above from B1 to Bn, the contribution of bits to scan-in power increases linearly. But the power contribution of bit n has one more term in overall equation and it is depending on last scanout response. So from above equation, it can be inferred that for test data compression, all the bits in test vector has equal weight but for scan power the bits carry the location wise increasing weight. The last bit has an added weight because of last output response also.

4.2.2 Zone wise Bit Filling

Let us divided the test vector {B1, B2,. . . . . . .Bn} in to three different zones.

Zone I : B1, B2,. . . . . . .Bk

Here the value of k is user defined. How to decide the value of K is described later on. It is clear that 1<k<n. In the proposed adaptive bit filling method, zone-I is bit filled using algorithm- I discussed above.

Zone II : Bk+1, Bk+2,. Bn-1

Zone – II is bit filled with minimum transition fill.

Zone III : Bn

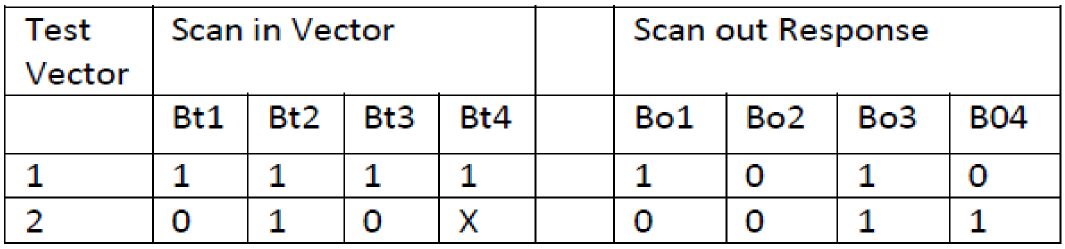
Bit n in zone-III will be filled after considering all reordering options. The following example clears the zone-III bit filling. Here scan-out response 1010 is being out while scan-in vector 010X is being in.

Case-I : X is filled with opposite bit of the last bit of output response.

In this case, the transitions are as shown below in fig. 4.2.

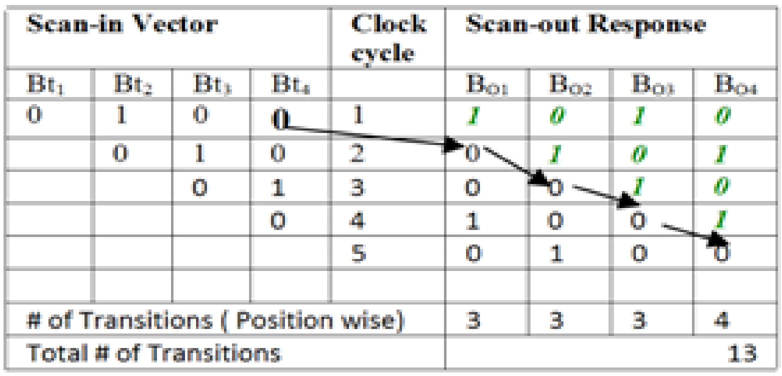
Case-II: X is filled with same bit the as last bit of output response.

In this case, the transitions are as shown below in fig. 4.3:



Test Vector	Scan in Vector					Scan out Response			
	Bt1	Bt2	Bt3	Bt4		Bo1	Bo2	Bo3	B04
1	1	1	1	1		1	0	1	0
2	0	1	0	X		0	0	1	1

Figure 4.1: Test Vector Sequence



Scan-in Vector				Clock cycle	Scan-out Response			
Bt_1	Bt_2	Bt_3	Bt_4		B_{O1}	B_{O2}	B_{O3}	B_{O4}
0	1	0	0	1	*1*	*0*	*1*	*0*
	0	1	0	2	0	*1*	*0*	*1*
		0	1	3	0	0	*1*	*0*
			0	4	1	0	0	*1*
				5	0	1	0	0
# of Transitions (Position wise)					3	3	3	4
Total # of Transitions								13

Figure 4.2: Transitions in case of Opposite Bit Filling

As it is made clear from above discussion, the bit filling of bit Bn will be done during test vector reordering, the following algorithm II is proposed for filling of remaining bits i.e. bits from B1 to Bn-1 .

Algorithm – II

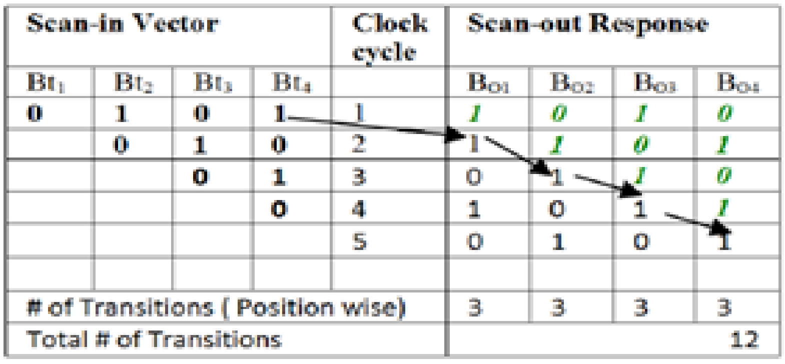
1. Lets take test data set T = [m, n] where m is number of test vectors and n is the number of bits in each test vector. This test data set is partially filled test set generated by ATPG.

2. Enter the adaptive partition index k

3. For all bits from Bi, k+1 to Bi, n-1 , apply MT fill algorithm for bit filling. Call this new matrix as Tmt

4. Divide the T into symbols of size s where n/s is an integer u.

5. The total number of symbols will be STotal.



Scan-in Vector				Clock cycle	Scan-out Response			
Bt_1	Bt_2	Bt_3	Bt_4		B_{O1}	B_{O2}	B_{O3}	B_{O4}
0	1	0	1	1	*1*	*0*	*1*	*0*
	0	1	0	2	1	*1*	*0*	*1*
		0	1	3	0	1	*1*	*0*
			0	4	1	0	1	*1*
				5	0	1	0	1
# of Transitions (Position wise)					3	3	3	3
Total # of Transitions								12

Figure 4.3: Transitions in case of Same Bit Filling

6. Separate out the fully specified symbols with

partially specified symbols.

7. Arrange the fully specified vectors in descending

order of their frequency of occurrence. Name it Sfully.

8. Arrange the partially specified vectors in descending order of their frequency of occurrence. Name it Spartially.

9. For each element Spi of Spartially (where n=i = 1),

follow the steps 9 to 12.

10. Find the bit location wise Hamming distance of Spi from first element Sf1of Sfully

.

11. If Hamming distance is zero, merge the Sp1 with Sf1. The frequency of Sf1will be now increased by

frequency of Sp1.

12. If Hamming distance is not zero, compare it with the next element of Sp1. and continue upto last element.

13. Still if the matching fully specified symbol is not

found, apply MT filling for that symbol and add it as

a new element to array of Sf1.

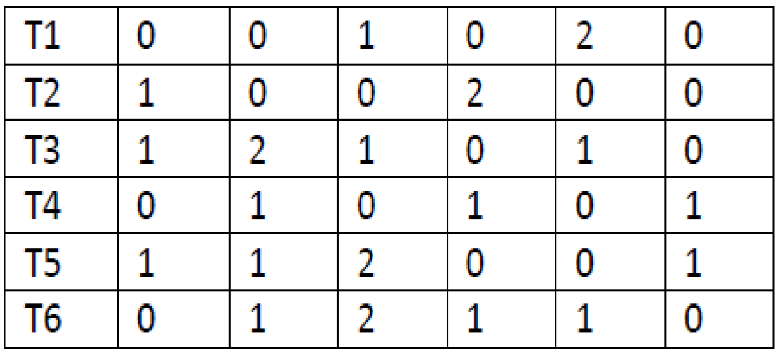
14. With this, The Spartially is merged in to Sfully.

15. Now as in Tmt, the symbols from 1 to +(k/s)+ in each row is still not processed

for bit filling. Symbols from +(k/s)+ to (u-1) in each row are fully specified (because of MT filling) and uth symbol in each row is still not to be processed (it is kept for reordering).
16. So all the partially specified symbols within the range 1 to +(k/s)+ in each row will be replaced with their corresponding fixed symbols.
17. The test set matrix T may now consist the don?t care
bits in last column only. Let?s call this matrix as T-1

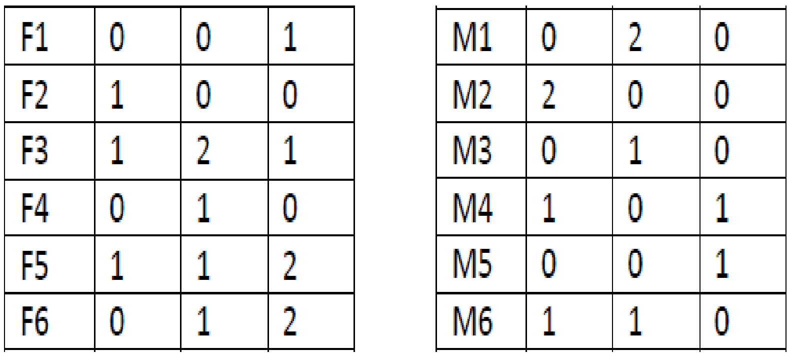
4.3 Mixed approach : Frequency directed + MT Fill

In the mixed approach the test vector is equally divided in two parts. In one part frequency directed approach is to be applied and in another part MT Fill algorithm is to be applied. After applying MT Fill to one part that data is to be mapped in another part of test vector . Now make a list of groups and apply Compression algorithm to that data and find percentage compression. For example fig 4.4 Shows the list of test vectors. Since the length of the test vector is 6 we are equally dividing in a segment of three bits each. That is shown in fig 4.5.a and 4.5.b. After MT Filling in fig 4.6 map it to fig 4.5 a. as per the ascending order as shown . Now make a list of groups with probability as in fig 4.6 and apply compression algorithm on that data and find out percentage compression.
Again repeat the procedure of section 4.2.1 to generate WTM matrix. Estimation of switching activity with artificial intelligence(AI) approach and compression with selective Huffman appproach for various benchmark circuits are shown in table II and III respectively.



T1	0	0	1	0	2	0
T2	1	0	0	2	0	0
T3	1	2	1	0	1	0
T4	0	1	0	1	0	1
T5	1	1	2	0	0	1
T6	0	1	2	1	1	0

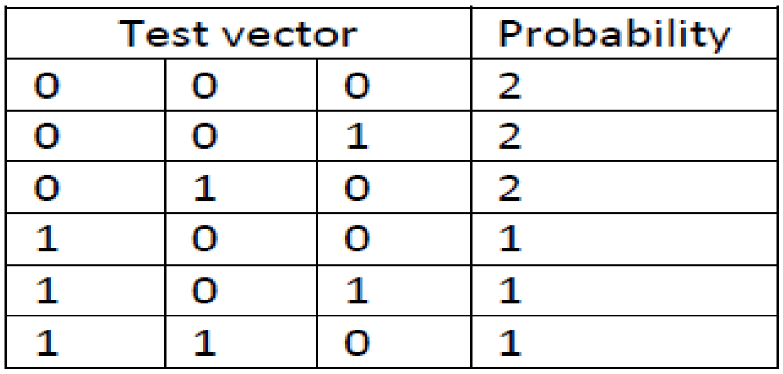
Figure 4.4: Test Vectors



F1	0	0	1
F2	1	0	0
F3	1	2	1
F4	0	1	0
F5	1	1	2
F6	0	1	2

M1	0	2	0
M2	2	0	0
M3	0	1	0
M4	1	0	1
M5	0	0	1
M6	1	1	0

Figure 4.5: Part of data for a.)Frequency Directed b.MT Fill)



Test vector			Probability
0	0	0	2
0	0	1	2
0	1	0	2
1	0	0	1
1	0	1	1
1	1	0	1

Figure 4.6: MT Filled data is mapped in above fig.

Table II: Comparison of Switching activity in different ISCAS Circuits

ISCAS Circuits	**Symbole Size**	**SWITCHING**		
		AI with Freq.Directed	**AI with Freq.Directed + MT Fill**	**AI with MT Fill**
S298	2	1420	1160	1082
	7	1273	1106	1082
S344	3	1241	1015	924
	5	1224	1024	924
S382	3	3402	3248	3233
	7	3409	3292	3233
S386	2	190	175	167
	3	189	170	167
S400	3	3252	3051	3000
	7	3427	3170	3000
S510	2	527	520	505
	3	527	520	505
S526	3	3895	3668	3579
	7	3772	3656	3579
S820	5	616	609	604
S832	5	686	673	630
S349	3	1404	1270	1213
	5	1322	1219	1213
S344	3	3703	3600	3582
	7	3613	3588	3582
S420	2	2342	2221	2142
	4	2385	2164	2142
	8	2261	2247	2142

Table III: Comparison of Compression in different ISCAS Circuits

ISCAS Circuits	**Symbole Size**	**Copmression**		
		Comp with FD	**Comp with FD + MT Fill**	**Comp with MT Fill**
S298	2	17.98	15.51	13.26
	7	23.64	22.66	20.68
S344	3	23.07	22.44	21.53
	5	22.56	20.15	18.97
S382	3	21.57	20.58	20.30
	7	23.11	21.09	18.62
S386	2	35.38	35.08	33.39
	3	47.07	46.78	44.44
S400	3	21.93	21.43	20.43
	7	25.18	22.81	19.80
S510	2	29.60	29.10	28.60
	3	20.64	20.40	20.14
S526	3	22.38	21.54	21.44
	7	25.00	23.69	21.28
S820	5	17.25	17.06	16.70
S832	5	16.18	16.00	15.63
S349	3	22.14	20.47	19.76
	5	17.38	15.95	15.71
S344	3	19.18	18.77	18.63
	7	18.63	18.36	17.00
S420	2	34.16	33.15	32.75
	4	21.76	20.64	20.58
	8	27.26	26.18	24.67

Chapter 5

Compression Algorithm

Test compression involves compressing the amount of test data (both stimulus and response) that must be stored on automatic test equipment (ATE) for testing with a deterministic (automatic test pattern generation [ATPG]-generated) test set. This is done by adding some additional on-chip hardware before the scan chains to decompress the test stimulus coming from the ATE and after the scan chains to compress the response going to the ATE. This differs from built-in self-test (BIST) and hybrid BIST in that the test vectors that are applied to the circuit under test (CUT) are exactly the same as the test vectors in the original deterministic (ATPG-generated) test set (no additional pseudo-random vectors are applied). Test compression can provide a 10x or even 100x reduction in the amount of test data stored on the ATE. This greatly reduces ATE memory requirements and even more importantly reduces test time because less data has to be transferred across the limited bandwidth between the ATE and the chip. Moreover, test compression methodologies are easy to adopt in industry because they are compatible with the conventional design rules and test generation flows used for scan testing.

Automatic test equipment (ATE) has limited speed, memory, and I/O channels. The test data bandwidth between the tester and the chip, as illustrated in Figure 5.1, is relatively low and generally is a bottleneck with regard to how fast a chip can be tested . The chip cannot be tested any faster than the amount of time required to transfer the test data which is equal to:

$$\frac{AmountofTestDataonTester}{(NumberofTesterChannels)(TesterClockRate)}$$

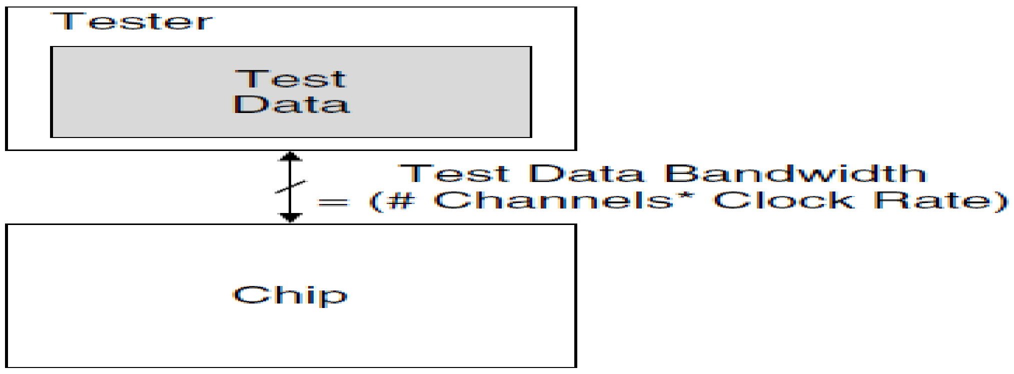


Figure 5.1: Block diagram illustrating test data bandwidth

The idea in test compression is to compress the amount of test data (both stimulus and response) that is stored on the tester. This provides two advantages. The first is that it reduces the amount of tester memory that is required. The second and more important advantage is that it reduces test time because less test data has to be transferred across the low bandwidth link between the tester and the chip. Test compression is achieved by adding some additional on-chip hardware before the scan chains to decompress the test stimulus coming from the tester and after the scan chains to compact the response going to the tester. This is illustrated in Figure 5.2. This extra on-chip hardware allows the test data to be stored on the tester in a compressed form.

Test data is inherently highly compressible. Test vectors have many unspecified bits that are not assigned values during ATPG (i.e., they are dont cares that can be filled with any value with no impact on the fault coverage). In fact, typically only 1 to 5% of the bits have specified (care) values, and even the specified values tend to be highly correlated due to the fact that faults are structurally related in the circuit. Consequently, lossless compression techniques can be used to significantly reduce the amount of test stimulus data that must be stored on the tester. The on-chip decompressor expands the compressed test stimulus back into the original test vectors (matching in all the care bits) as they are shifted into the scan chains. Output response is even more compressible than test stimulus because lossy compression (also known as compaction) can be used.

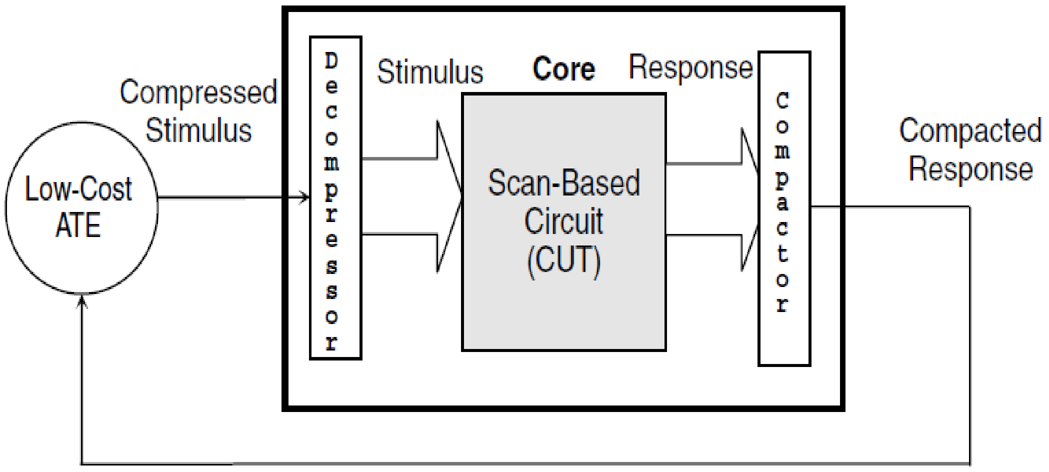


Figure 5.2: Architecture for test compression

5.1 Test Stimulus Compression

A test cube is defined as a deterministic test vector in which the bits that are not assigned values by the ATPG procedure are left as dont cares (Xs). Normally, ATPG procedures perform random fill, in which all the Xs in the test cubes are filled randomly with 1s and 0s to create fully specified test vectors; however, for test stimulus compression, random fill is not performed during ATPG so the resulting test set consists of incompletely specified test cubes. The Xs make the test cubes much easier to compress than fully specified test vectors. As mentioned earlier, test stimulus compression should be an information lossless procedure with respect to the specified (care) bits in order to preserve the fault coverage of the original test cubes. After decompression, the resulting test patterns shifted into the scan chains should match the original test cubes in all the specified (care) bits. Many schemes for compressing test cubes have been proposed. They can be broadly classified into the three categories shown below; these schemes are described in detail in the following subsections (shown in parentheses):

1. **Code-based schemes** These schemes use data compression codes to encode the test cubes.

2. **Linear-decompression-based schemes** These schemes decompress the data using only linear operations (e.g., linear feedback shift registers [LFSRs] and exclusive-OR [XOR] networks).

3. **Broadcast-scan-based schemes** These schemes are based on broadcasting the same value to multiple scan chains.

5.2 Code-Based Schemes

One approach for test compression is to use data compression codes to encode the test cubes. Data compression codes partition the original data into symbols, and then each symbol is replaced with a codeword to form the compressed data. The decompression is performed by having a decoder that simply converts each codeword into the corresponding symbol.

Data compression codes can be classified into four categories, depending on whether the symbols have a fixed size (i.e., each symbol contains exactly n bits) or a variable size (i.e., different symbols have different numbers of bits) and whether the codewords have a fixed or variable size.

1. **Fixed-to-fixed Dictionary code**
2. **Fixed-to-variable Huffman code**
3. **Variable-to-fixed Run-length code**
4. **Variable-to-variable Golomb code**

5.2.1 Dictionary Code (Fixed-to-Fixed)

In fixed-to-fixed coding, the original test cubes are partitioned into n-bit blocks to form the symbols. These symbols are then encoded with codewords that each have b bits. In order to get compression, b must be less than n. One can view each symbol as an entry in a dictionary and each codeword as an index into the dictionary that points to the corresponding symbol. There are 2^n possible symbols and 2^b possible codewords, so not all possible symbols can be in the dictionary. If $S_{dictionary}$ is the set of symbols that are in dictionary and S_{data} is the set of symbols that occur in the original data, then if $S_{data} \subseteq S_{dictionary}$, it is a complete dictionary; otherwise, it is a partial dictionary. Compression can be achieved with a complete dictionary provided that the number of distinct symbols that occur in the original data $|S_{data}|$ is much less than 2^n, the number of all possible symbols.

A test compression scheme that uses a complete dictionary was described in Figure 5.3. There are n scan chains,and the test cubes are partitioned into n-bit symbols such that each scan slice corresponds to a symbol. Each scan slice is comprised of the n-bits that are loaded into the scan chains in each clock cycle as illustrated in Figure 5.3. The Xs in the test cubes are filled so as to minimize the number of distinct symbols (i.e., $|S_{data}|$). The size of each codeword is b bits, where $b = \lceil \log_2 |Sdata \rceil$. Note that, with this scheme, b channels from the tester can be used to load n scan chains. Normally, b channels from the tester can only load b scan chains. By having more scan chains, the length of each scan chain becomes shorter, thus reducing the number of clock cycles required to load each scan vector and therefore reducing the test time. This is a good illustration of how test compression reduces not only tester storage but also test time.

A drawback of using a complete dictionary is that the size of the dictionary can become very large, resulting in too much overhead for the decompressor. In , a partial dictionary coding scheme was proposed in which the size of the dictionary is selected by the user based on how much area the user wants to allocate for the

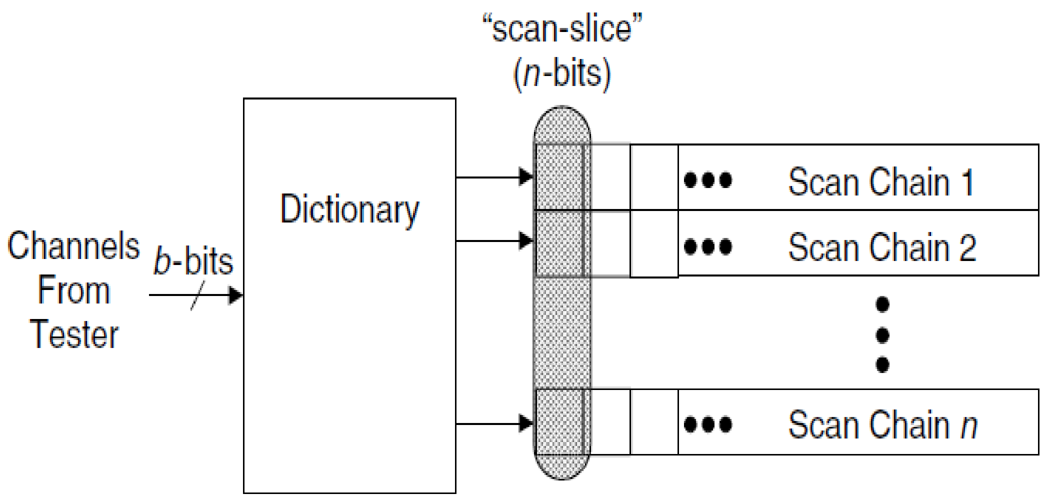
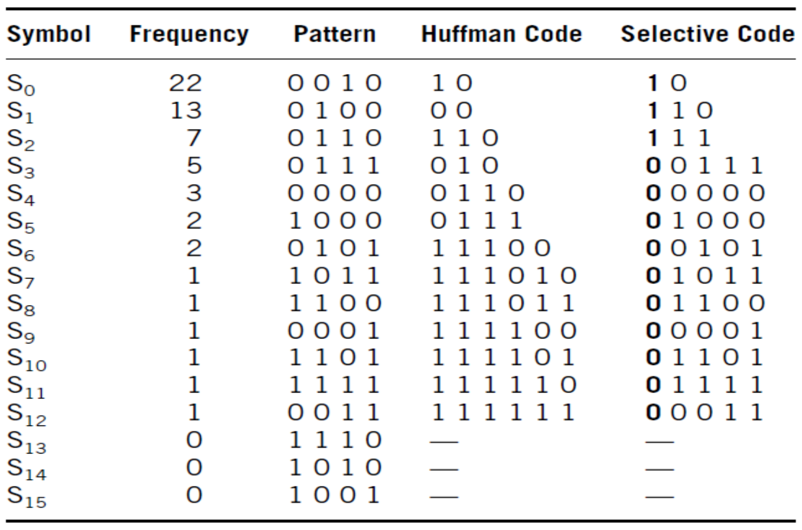


Figure 5.3: Test compression using a complete dictionary

decompressor. If the size of the dictionary is 2^b, then the 2^b symbols that occur most frequently in the test cubes are placed in the dictionary. For any symbol that is not in the dictionary, the symbol is left unencoded and the dictionary is bypassed. An extra bit is added to each codeword to indicate whether or not to use the dictionary.

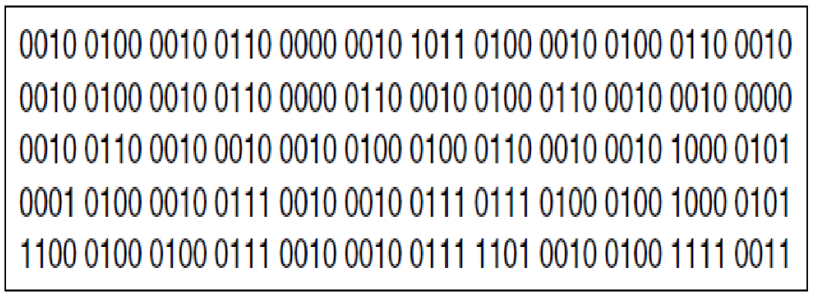
5.2.2 Huffman Code (Fixed-to-Variable)

In fixed-to-variable coding, the original test cubes are partitioned into n-bit blocks to form the symbols. These symbols are then encoded using variable-length codewords. One form of fixed-to-variable coding is statistical coding, where the idea is to calculate the frequency of occurrence of the different symbols in the original test cubes and make the codewords that occur most frequently have fewer bits and those that occur least frequently more bits. This minimizes the average length of a codeword. A Huffman code is an optimal statistical code that is proven to provide the shortest average codeword length among all uniquely decodable fixed-to-variable length codes. A Huffman code is obtained by constructing a Test Compression Huffman tree. The path from the root to each leaf in the Huffman tree gives the codeword for the binary string corresponding to the leaf. An example of constructing a Huffman code can be seen in figure 5.4 and Figures 5.5. Figure 5.5 shows the frequency of occurrence of each of the possible symbols. The example shown in Figure 5.5 has a total of 60 4-bit symbols. Figure 5.6 shows the Huffman tree for this frequency distribution, and the corresponding codewords are shown in figure 5.4.



Symbol	Frequency	Pattern	Huffman Code	Selective Code
S_0	22	0 0 1 0	1 0	**1** 0
S_1	13	0 1 0 0	0 0	**1** 1 0
S_2	7	0 1 1 0	1 1 0	**1** 1 1
S_3	5	0 1 1 1	0 1 0	**0** 0 1 1 1
S_4	3	0 0 0 0	0 1 1 0	**0** 0 0 0 0
S_5	2	1 0 0 0	0 1 1 1	**0** 1 0 0 0
S_6	2	0 1 0 1	1 1 1 0 0	**0** 0 1 0 1
S_7	1	1 0 1 1	1 1 1 0 1 0	**0** 1 0 1 1
S_8	1	1 1 0 0	1 1 1 0 1 1	**0** 1 1 0 0
S_9	1	0 0 0 1	1 1 1 1 0 0	**0** 0 0 0 1
S_{10}	1	1 1 0 1	1 1 1 1 0 1	**0** 1 1 0 1
S_{11}	1	1 1 1 1	1 1 1 1 1 0	**0** 1 1 1 1
S_{12}	1	0 0 1 1	1 1 1 1 1 1	**0** 0 0 1 1
S_{13}	0	1 1 1 0	—	—
S_{14}	0	1 0 1 0	—	—
S_{15}	0	1 0 0 1	—	—

Figure 5.4: Statistical Coding based on symbole frequencies



```
0010 0100 0010 0110 0000 0010 1011 0100 0010 0100 0110 0010
0010 0100 0010 0110 0000 0110 0010 0100 0110 0010 0010 0000
0010 0110 0010 0010 0010 0100 0100 0110 0010 0010 1000 0101
0001 0100 0010 0111 0010 0010 0111 0111 0100 0100 1000 0101
1100 0100 0100 0111 0010 0010 0111 1101 0010 0100 1111 0011
```

Figure 5.5: Example of test set divided into 4 bit block

The test cubes are partitioned into symbols and then the Xs in the test cubes are filled to maximally skew the frequency of occurrence of the symbols. A selective Huffman code in which only the k most frequently occurring symbols are encoded is used. The reason for this is that using a full Huffman code that encodes all n-bit symbols requires a decoder with 2^n-1 states. By only selectively encoding the k most frequently occurring symbols, the decoder requires only n+k states. It was

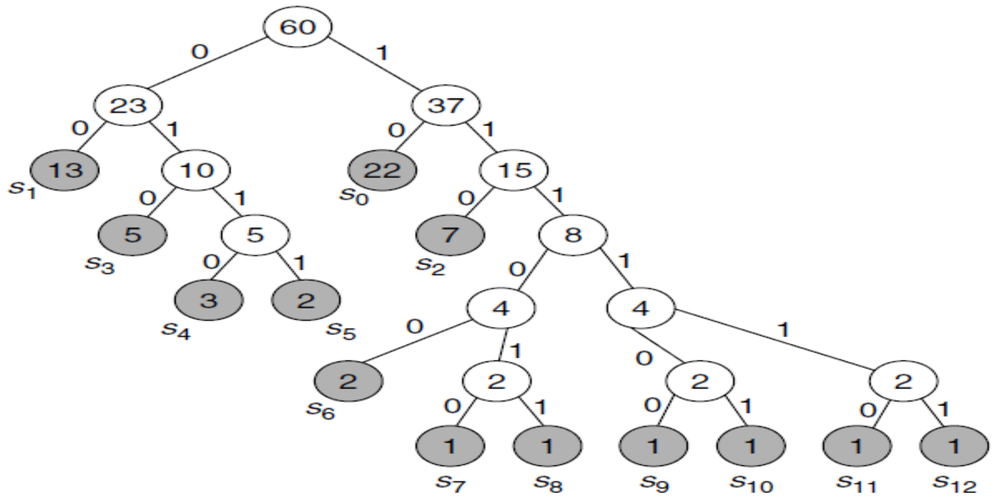


Figure 5.6: Huffman Tree

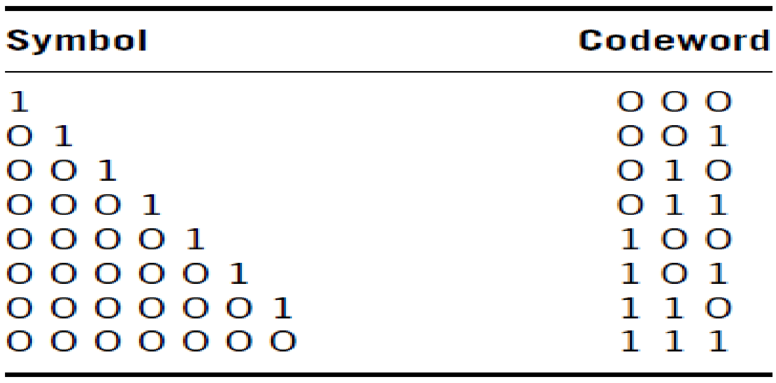
shown that a selective Huffman code achieves only slightly less compression than a full Huffman code for the same symbol size while using a much smaller decoder. Because the decoder size grows only linearly with selective Huffman encoding, it is possible to use a much larger symbol size, which significantly improves the effectiveness of the code thereby achieving much more overall compression.

In selective Huffman coding, an extra bit is added at the beginning of each codeword to indicate whether or not it is coded. As an example, consider selective Huffman coding for the test set shown in Figure 5.5, where only the three most frequency occurring symbols are encoded (i.e., k = 3). A Huffman tree is built only for the three most frequently occurring symbols. The codewords are then constructed as shown in figure 5.4. The first bit of the codewords for the three most frequently occurring symbols is 1 to indicate that they are coded (and hence must pass through the decoder). The first bit of the rest of the codewords is a 0 to indicate that they are not coded (i.e., the remainder of the codeword is simply the unencoded symbol itself).

A method for improving the compression with a statistical code by modifying the test cubes without losing fault coverage is also described. The goal is to modify the specified bits in the test cubes in a way that maximally skews the frequency distribution.

5.2.3 Run-Length Code (Variable-to-Fixed)

In variable-to-fixed coding, the original test cubes are partitioned into variablelength symbols, and the codewords are each b-bits long. In run-length coding, one particular variable-to-fixed coding scheme, the symbols consist of runs of consecutive 0s or 1s. An example of a 3-bit run-length code for runs of 0s is given in figure 5.7. Each codeword is 3 bits long and encodes different length runs of 0s. As an example, the sequence 001 0001 01 0000001 1 000001 can be encoded into 010 011 001 110 000 101, which is a reduction from 23 bits to 18 bits. For very long runs of 0s (longer than 7), codeword 111 can be used repeatedly as needed. Note that only data with an unbalanced number of 0s and 1s can be efficiently compressed by a run-length code.



Symbol	Codeword
1	0 0 0
0 1	0 0 1
0 0 1	0 1 0
0 0 0 1	0 1 1
0 0 0 0 1	1 0 0
0 0 0 0 0 1	1 0 1
0 0 0 0 0 0 1	1 1 0
0 0 0 0 0 0 0	1 1 1

Figure 5.7: 3 Bit runlength code

Test compression based on a run-length code was proposed using a cyclical scan architecture as shown in Figure 5.8. The cyclical scan architecture XORs the data currently being shifted in with the previous test vector. Thus, instead of applying the original test set, TD=$\{t_1,t_2,t_3.........t_n\}$ a difference vector set, Tdiff = t_1, $t_1 \oplus t_2$, $t_2 \oplus t_3$, ..., $t_{(n-1)} \oplus t_n$ is applied instead. The advantage of this is that the test vectors can be ordered so similar test vectors come after each other so the difference vectors have many 0's. This enhances the effectiveness of run-length coding.

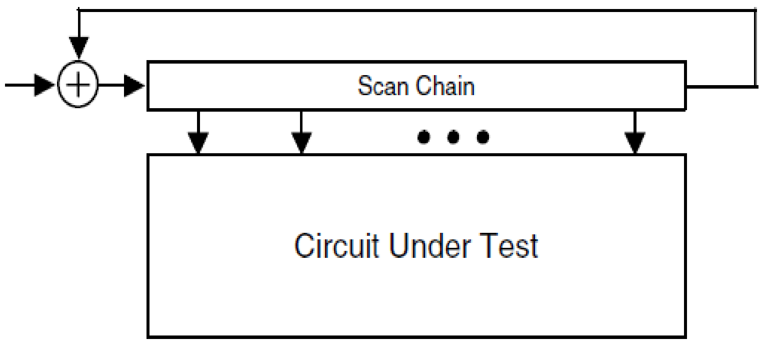
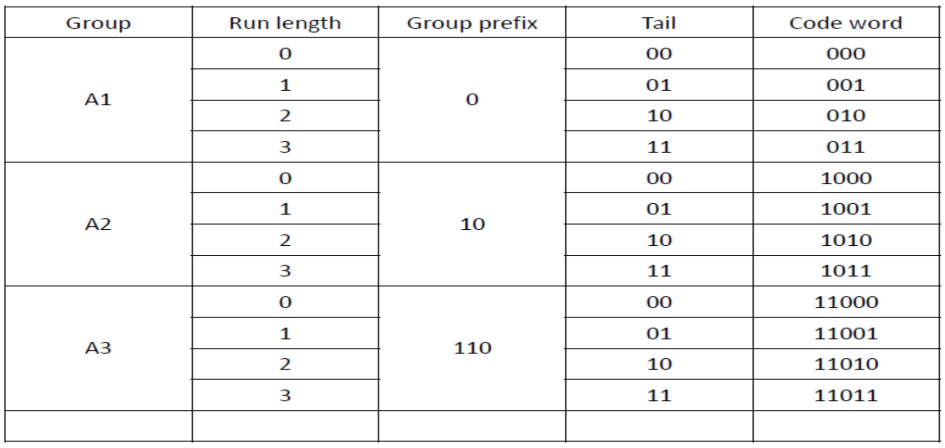


Figure 5.8: Cyclical scan architecture for applying difference vectors

5.2.4 Golomb Code (Variable-to-Variable)

In variable-to-variable coding, both the symbols and codewords have variable length. A Golomb code [Chandra 2001] is a variable-to-variable code that evolved from the run-length code. To construct a Golomb code, a specific parameter m, called the group size (usually a power of 2), is first chosen. All the run-lengths are divided into groups of size m denoted by A_1,A_2,A_3...... The set of run-lengths {0,1,2,.....m-1} form the first group A1, the set of run-lengths {m,m+1,m+2.....2m} form the second group A_2, and so on. Each codeword of a Golomb code consists of two parts: a group prefix and a tail. A run-length L that belongs to group Ak is assigned a group prefix (k-1) of ones followed by a zero. The tail is an index of the run-length in a group. If m is chosen to be a power of 2 (i.e., m = 2N for some integer N), then each group contains 2^N members, and a $\log_2$m-bit-long sequence (tail) can uniquely identify each member within the group. Figure 5.9 shows an example of Golomb code in which each group contains four run-lengths. In the original test sequence TD, the run-length of the 0s before the first 1 is 2. Based on figure 5.9, the sequence 001 is encoded as 010, in which the 0 is the prefix and 10 is the tail. Similar procedures are repeated until all the run-lengths are processed. It can be found that the length of the test sequence can be reduced from 43 to 32. In a Golomb code, each group contains the same number of run-lengths and thus may still be inefficient in some cases.

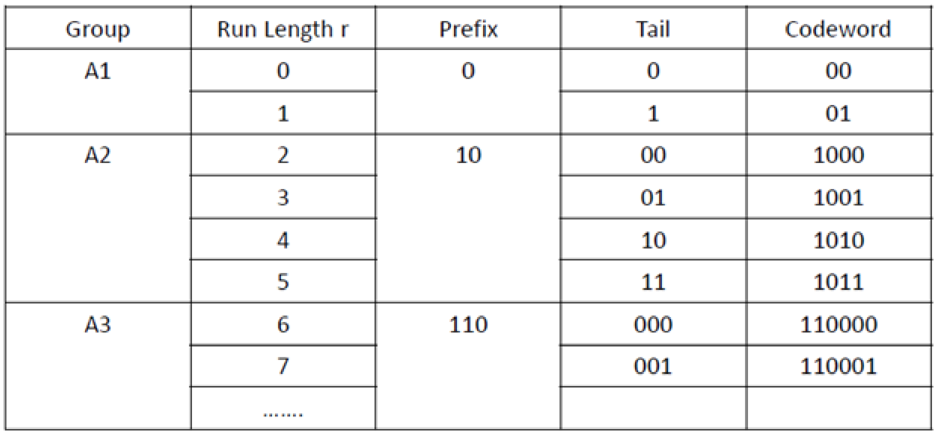


Group	Run length	Group prefix	Tail	Code word
A1	0	0	00	000
	1		01	001
	2		10	010
	3		11	011
A2	0	10	00	1000
	1		01	1001
	2		10	1010
	3		11	1011
A3	0	110	00	11000
	1		01	11001
	2		10	11010
	3		11	11011

Figure 5.9: GOLOMB Code

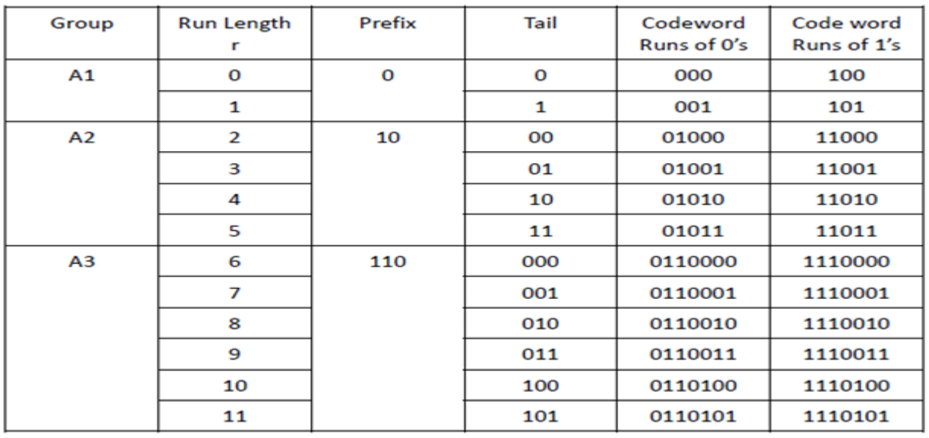
In 2001, Chandra and Chakrabarty proposed a new scheme based on the observation that the frequency of runs of 0s with run length less than 20 is high and even within the range of 0 to 20, the frequency of runs of length l decreases rapidly with increasing l. So test data compression can be more efficient if the runs of 0s with shorter run length are mapped to shorter codewords. So further optimization can be achieved using frequency-directed run-length (FDR) codes shown in figure 5.10. The FDR is similar to Golomb code but the difference is the variable group size. The size of the ith group is equal to 2^i i.e. that group contains 2^i members. Each codeword consists of two parts-a group prefix and a tail. The group prefix is used to identify the group to which the run belongs and the tail is used to identify the members within the group.

The FDR code is very efficient for compressing data that has few 1s and long runs of 0s but inefficient for data streams that are composed of both runs of 0s and runs of 1s. Generally test vectors contain 0s and 1s in group i.e. there will be a run of 1s followed by run of 0s and vice versa. Maleh and Abaji proposed an extension of FDR (EFDR) shown in figure 5.11. Here the run of 0s followed by bit '1' and run of 1s followed by bit '0' are coded same way as FDR but adding an extra bit at beginning of FDR codeword. If that extra bit of codeword is 0, the run is of 0 type and if the first bit is 1 then the run type is of '1'.



Group	Run Length r	Prefix	Tail	Codeword
A1	0	0	0	00
	1		1	01
A2	2	10	00	1000
	3		01	1001
	4		10	1010
	5		11	1011
A3	6	110	000	110000
	7		001	110001
	....			

Figure 5.10: FDR Code

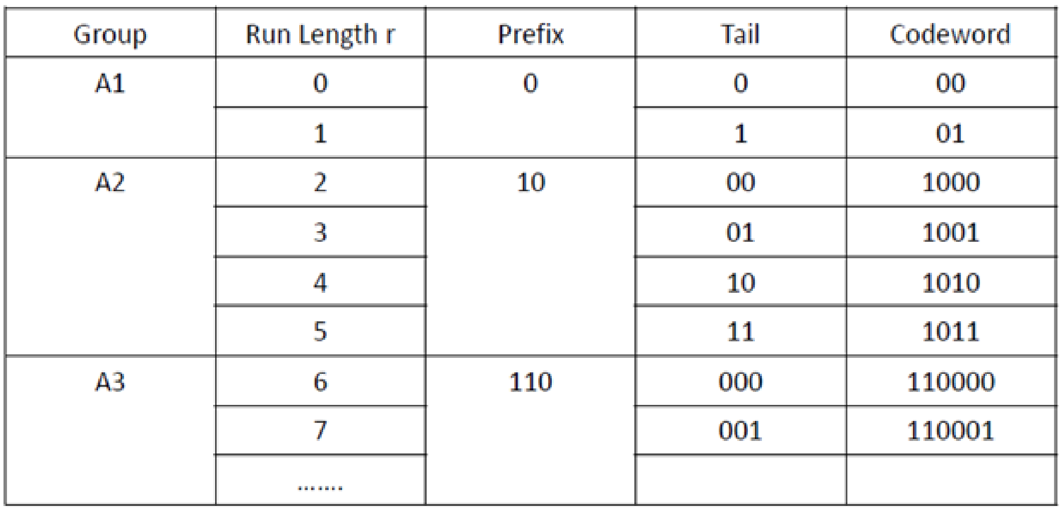


Group	Run Length r	Prefix	Tail	Codeword Runs of 0's	Code word Runs of 1's
A1	0	0	0	000	100
	1		1	001	101
A2	2	10	00	01000	11000
	3		01	01001	11001
	4		10	01010	11010
	5		11	01011	11011
A3	6	110	000	0110000	1110000
	7		001	0110001	1110001
	8		010	0110010	1110010
	9		011	0110011	1110011
	10		100	0110100	1110100
	11		101	0110101	1110101

Figure 5.11: EFDR Code

Generally, the test set T is composed of alternative runs of zeros and run of ones. In Alternative Run Length coding style, instead of adding the extra bit to each run length code-word, only one bit will be added at beginning to indicate the type of first run length and then successive run length is automatically alternate type. So all run lengths are coded with normal FDR but at beginning, one bit will be added to indicate the first run type as shown in figure 5.12.

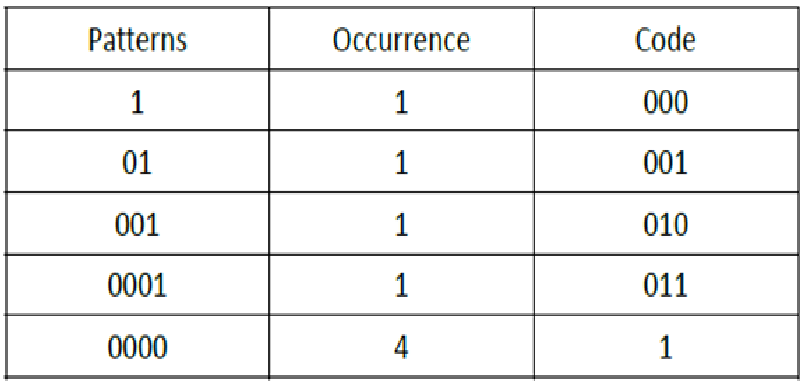
An evolution in Alternate Run Length based FDR is Shifted Alternate Runlength based FDR. Based on the fact that, in alternate FDR, there will be no run-length of 0 sizes. So code word for run length size 0 is not needed now. This code word is assigned to run length size 1 and so on each code word is shifted to one position higher. This helps in achieving higher compression compared to Alternate FDR.



Group	Run Length r	Prefix	Tail	Codeword
A1	0	0	0	00
	1		1	01
A2	2	10	00	1000
	3		01	1001
	4		10	1010
	5		11	1011
A3	6	110	000	110000
	7		001	110001
	....			

Figure 5.12: SAFDR Code

The FDR code is composed from two parts, a prefix and a tail. In order to decompress a FDR code, the on-chip decoder has to identify these two parts. Because the code is not dependent on a group size as Golomb codes, the decoder has to detect the length of the prefix in order to decode the tail. So, the FDR code requires a more complicated decoder with higher area overhead. Therefore, regardless of the good compression ratios the area overhead of FDR is a disadvantage. So a mix of Huffman and FDR is proposed which in stead of only patterns of fixed-length uses patterns of variable-length as input to the Huffman algorithm (VIHC). Here the compression ratio is retained because of FDR and the area overhead is reduced using selective Huffman Coding as shown in figure 5.13.



Patterns	Occurrence	Code
1	1	000
01	1	001
001	1	010
0001	1	011
0000	4	1

Figure 5.13: VIHC Code

Chapter 6

Error Resilience

Error-resilience is the capability of a test data stream [which is transferred from an automatic test equipment (ATE) to the device under test (DUT)] to tolerate errors. These errors may occur in an ATE, either in the electronics components of the loadboard or in the high-speed serial communication links. Initially, it is shown that the combined effect of such errors and test data compression can seriously degrade the test quality (as measured by coverage) of the compressed data streams. The effects of errors on compressed streams are analyzed and various test data compression approaches are evaluated.

6.1 Brief Overview

THE electronics industry has been highly affected by the widespread use of third-party ready-to-use intellectual property (IP) cores [1] for assembling complex ICs. Recently, the integration of multiple cores has resulted in the manufacturing of system-on-a-chip (SoC) on the same silicon die. A complete test set for an SoC includes test sets for each core (often provided by the core vendor). The aggregate volume of these test sets can be very large, in most cases well in the range of a few gigabits. The application of such large volume of test data requires significant storage facilities on an ATE and may take a long time to execute. Large storage translates into higher equipment cost, while long test application time impacts the ATE throughput [devices under test (DUTs) tested per unit time], thus increasing the overall cost of the test. To address these problems, test data compression through so-called resource partitioning has been introduced. By compressing the initial test sets, a smaller volume of data is loaded on the ATE, reducing the cost of storage. Additionally, the compressed test data is transferred to the DUT where it is decompressed and applied. As a smaller volume of data is transferred, then test application time can be reduced too. Traditionally, a reduction in data volume has been accomplished through vector compaction [2]; however, compression can be applied to already compacted data to further reduce its volume.

6.1.1 Previous Work done

Current literature has addressed compression mostly with respect to its efficiency in reducing data volume (commonly quantified by the compression ratio), test application time, and overhead complexity (often for the on-DUT decompression circuitry). In [3], run-length coding has been proposed for compression. This has been extended by Golomb and frequency- directed-run length (FDR) coding techniques in which variable-length codewords are used for runs of 0 [4], [5]. In [6], Huffman coding has been applied to fixed length symbols. The application of Huffman coding to variable-length symbols has been studied in the so-called variable-input-huffman coding (VIHC) [7]. To avoid the exponential complexity of a Huffman decoder with an increasing length of symbols, [8] has introduced selective Huffman coding. For manufacturing testing of VLSI systems such as SoC, the transfer of compressed data between the ATE and the DUT must be also considered. In today's ATE architectures, the high volume and the gigahertz speed of communication may result in the likely occurrence of errors in data transmission. These errors can be modeled by the so-called bit-flips which affect the test data stream. [9] has statistically evaluated the error resilience of Huffman coding to achieve a higher level of error resilience against bit-flips. This has been further expanded in [11] by analyzing the possible outcomes of a bit-flip on compressed test streams and evaluating the loss of coverage as a measure of error resilience for different compression techniques.

6.1.2 Basic Principles

Why Bit-Flips

Various scenarios can lead to bit-flips in the operation of the head of an ATE. The presence of a transmission line between the ATE pins and the DUT may be affected by noise in the so-called loadboard, leading to errors due to bit-flips. Moreover, the generation of test inputs as pin waveforms to the DUT, can also introduce an additional source of bit-flips (for example a ground bounce at the DUT receiver connected to the loadboard can result in bit-flips if the voltage levels are close to the margins such as for 3.3 or 1.5 V DUTs). In general, once the ATE test program and loadboard are fully debugged, the occurrence of bit-flips can be reduced. However, as testing requires very high speed (tens of millions ofATE cycles over hundreds of pins), fewbit-flips must be considered to be unavoidable in ATE environments [13].
Bit-flips can negatively affect the testing process over its entire spectrum: while developing test programs, bit-flips increase costs associated with debugging and development, while reducing productivity. During manufacturing test, a bit-flip can change data such that the coverage of the erroneous test set (as affected by bit-flips) can be reduced.

Experimental Framework

For experimental evaluation of bit-flip effects and related phenomena, the MINTEST dynamically compacted test-cubes for the full-scan versions of different benchmark circuits are considered [2]. The location of the bit-flips is assumed to be uniformly distributed over the entire bit-stream of a test set. This spreads bit-flips evenly and permits to evaluate the results under the most stringent conditions. Bit-flip counts of 1, 2, 5, and 10 are used in experiments. Seven compression techniques (Golomb, Huffman, Selective Huffman, FDR, EFDR, SAFDR and VIHC) are used to compress the test vectors of each circuit.

Bit-Flips With Differentiation

In previous test data compression approaches, data is often preprocessed by vector reordering and differentiation. Differentiation can further decrease error-resilience. The difference vectorsD1,D2Dn are defined as D1=V1, D2=V2 XOR V1, Dn= Vn XOR V (n-1) For decompression, the original vectors are restored from the difference vectors as V1=D1, V2=D2 XOR V1, ..Vn=Dn XOR D(n-1) XOR.D2 XOR D1. By substitution, ; therefore, if a bit-flip occurs in Di, then the process of restoring all subsequent vectors (i.e. from the ith to the nth) will be affected.

To evaluate these phenomena, for each benchmark circuit, bit-flips are injected into the reordered and differentiated test data stream. The reduction in coverage is the measure by which the effect of bit-flips on differentiation is evaluated.

Bit-Flips Without Compression

The effect of a bit-flip is significant when test data compression is required. On average, bits in a compressed bit-stream carry more information compared with the initial uncompressed stream because compression techniques rely on removing redundancy. Consequently, on average, a bit-flip in a compressed bit stream can destroy significantly more information. The coverage loss of uncompressed test streams due to bit-flips represent a lower bound on the error-resilience. To properly quantify this phenomena, bit-flips are injected into the uncompressed test data for each benchmark circuit, the erroneous test data is then fault simulated. Table II shows the fault coverage results. The average coverage loss is only 0.01%-0.03% for 1 to 10 bit-flips.

Table I: Bit Flip without Compression

Benchmark Circuit	**Fault Coverage**				**Fault coverage In %**
	1 - Bit flip	**2 - Bit flip**	**5 - Bit flip**	**10 - Bit flip**	
S27	98.00	96.00	96.00	95.50	98.00
S298	97.64	97.64	97.64	97.64	97.64
S344	99.28	99.28	99.28	99.00	99.28
S349	99.28	99.28	99.28	99.28	99.28
S382	97.94	97.94	97.94	97.94	97.94
S386	99.55	99.55	99.55	99.55	99.70
S400	97.59	97.59	97.59	97.59	97.59
S420	99.68	99.36	99.36	99.36	99.68
S444	98.22	98.22	98.22	98.22	98.22
S510	99.62	99.62	99.62	99.62	99.85
S526	97.95	97.95	97.95	97.95	97.95
S641	99.22	99.22	99.22	99.22	99.22
S713	99.23	99.23	99.13	99.13	99.23
S820	99.14	98.92	98.92	98.92	99.14
S832	97.17	97.17	97.17	97.17	98.39
S838	99.67	99.67	99.67	99.67	99.84

Bit-Flips With Compression

Consider a compressed test data with a compression ratio of 95%, i.e., a 20 reduction in data volume. On average, each bit in the compressed bit stream is expanded into 20 bits of the initial uncompressed stream. If a bit-flip occurs in the compressed bit stream, then we can expect 20 bits to be affected in the uncompressed stream. These bits can be scattered over the uncompressed stream, thus, affecting multiple vectors and seriously degrading the coverage of the decompressed test set. To quantify the decrease in error-resilience when compression is employed, bit-flips are injected into the compressed test stream of each circuit. Seven compression techniques (Golomb, Huffman, Selective Huffman, FDR, EFDR, SAFDR and VIHC) are used to compress the test vectors of each circuit. Tables II -VII show the average coverage as representative of error-resilience of different compression techniques. In all tables, a column corresponds to a specific bit-flip count; for example, "1b" means only 1 bit-flip has been injected. Huffman coding results in a much lower coverage loss due to bit-flips.

Table II: Bit Flip after Compression for Huffman Coding

Benchmark circuit	**Fault Coverage**				**Max. Fault coverage (%)**
	1b	**2b**	**5b**	**10b**	
S27	98.00	95.00	97.00	95.00	98.00
S298	97.47	97.64	97.64	97.30	97.64
S344	99.00	99.00	99.00	99.00	99.28
S349	98.70	98.70	98.70	98.55	99.28
S382	97.81	97.81	97.81	97.81	97.94
S386	99.39	99.39	98.79	98.79	99.70
S400	97.34	97.34	97.34	97.34	97.59
S420	99.68	99.68	99.20	99.20	99.68
S444	97.33	97.33	97.33	97.33	98.22
S510	99.39	99.39	99.39	99.08	99.85
S526	97.52	97.52	97.52	97.52	97.95
S641	99.13	99.13	99.13	98.83	99.22
S713	99.23	99.23	99.04	98.85	99.23
S820	98.92	98.92	98.92	98.86	99.14
S832	98.34	98.34	98.34	98.28	98.39

Table III: Bit Flip after Compression for Selective Huffman Coding

Benchmark circuit	**Fault Coverage**				**Max. Fault coverage (%)**
	1b	**2b**	**5b**	**10b**	
S27	95.00	95.00	90.00	81.00	98.00
S298	97.30	97.30	97.30	97.30	97.64
S344	99.00	99.00	99.00	99.00	99.28
S349	98.70	98.70	98.70	98.70	99.28
S382	97.81	97.81	97.81	97.69	97.94
S386	99.39	99.39	99.39	99.20	99.70
S400	97.34	97.34	97.34	97.34	97.59
S420	99.68	99.68	99.20	99.20	99.68
S444	97.33	97.33	97.33	97.33	98.22
S510	99.39	99.39	99.08	98.69	99.85
S526	97.52	97.52	97.52	97.52	97.95
S641	99.13	99.13	98.93	98.93	99.22
S713	99.13	99.23	99.04	99.04	99.23
S820	98.92	99.14	98.86	98.86	99.14
S832	98.34	98.34	98.28	98.28	98.39

Table IV: Bit Flip after Compression for FDR Coding

Benchmark circuit	**Fault Coverage**				**Max. Fault coverage (%)**
	1b	**2b**	**5b**	**10b**	
S27	95.00	98.00	97.00	97.00	98.00
S298	97.64	97.64	97.64	97.30	97.64
S344	99.28	99.00	99.00	99.00	99.28
S349	99.28	99.28	98.99	98.99	99.28
S382	97.94	97.94	97.94	97.81	97.94
S386	99.39	99.39	99.39	99.79	99.70
S400	97.59	97.59	97.59	97.34	97.59
S420	99.68	99.68	99.20	99.68	99.68
S444	98.22	98.22	97.71	97.71	98.22
S510	99.39	99.39	99.39	99.39	99.85
S526	97.95	97.95	97.95	97.74	97.95
S641	99.22	99.22	99.22	99.13	99.22
S713	99.23	99.23	99.23	99.23	99.23
S820	98.92	99.14	98.92	98.80	99.14
S832	98.34	98.34	98.34	98.17	98.39

Table V: Bit Flip after Compression for EFDR Coding

Benchmark circuit	**Fault Coverage**				**Max. Fault coverage (%)**
	1b	**2b**	**5b**	**10b**	
S27	98.00	98.00	90.00	85.00	98.00
S298	97.64	97.30	97.30	97.30	97.64
S344	99.28	99.00	99.00	99.00	99.28
S349	98.70	98.70	98.55	98.55	99.28
S382	97.94	97.81	97.81	97.81	97.94
S386	99.09	99.70	98.79	98.18	99.70
S400	97.59	97.34	97.34	97.34	97.59
S420	99.68	99.68	99.20	99.20	99.68
S444	98.22	97.33	97.33	97.33	98.22
S510	99.85	99.39	99.08	98.85	99.85
S526	97.95	97.52	97.52	97.52	97.95
S641	99.13	99.13	99.13	99.13	99.22
S713	99.23	99.23	99.23	99.23	99.23
S820	99.14	98.92	99.03	98.69	99.14
S832	98.34	98.34	98.28	98.11	98.39

Table VI: Bit Flip after Compression for SAFDR Coding

Benchmark circuit	**Fault Coverage**				**Max. Fault coverage (%)**
	1b	**2b**	**5b**	**10b**	
S27	98.00	90.00	86.00	85.00	98.00
S298	97.64	97.30	97.30	97.30	97.64
S344	99.28	99.00	99.00	99.00	99.28
S349	99.28	98.70	98.55	98.55	99.28
S382	97.94	97.81	97.81	97.81	97.94
S386	99.39	99.39	98.79	98.33	99.70
S400	97.59	97.34	97.34	97.34	97.59
S420	99.68	99.68	99.20	99.20	99.68
S444	98.22	97.33	97.33	97.33	98.22
S510	99.39	99.39	99.08	98.69	99.85
S526	97.95	97.52	97.52	97.52	97.95
S641	99.13	99.13	99.13	99.13	99.22
S713	99.23	99.23	99.23	99.23	99.23
S820	99.14	98.92	98.97	98.69	99.14
S832	98.34	98.34	98.17	98.11	98.39

Table VII: Bit Flip after Compression for GOLOMB Coding

Benchmark circuit	**Fault Coverage**				**Max. Fault coverage (%)**
	1b	**2b**	**5b**	**10b**	
S27	95.00	90.00	67.00	65.00	98.00
S298	97.30	97.30	97.30	96.96	97.64
S344	99.00	99.00	99.00	99.00	99.28
S349	98.70	98.70	98.70	98.55	99.28
S382	97.81	97.81	97.81	97.81	97.94
S386	99.39	99.39	98.79	98.79	99.70
S400	97.34	97.34	97.34	97.34	97.59
S420	99.68	99.68	99.20	99.20	99.68
S444	97.33	97.33	97.33	97.33	98.22
S510	99.39	99.39	99.08	98.62	99.85
S526	97.52	97.52	97.74	97.52	97.95
S641	99.13	99.13	99.13	99.13	99.22
S713	99.23	99.23	99.23	99.23	99.23
S820	98.92	98.92	99.09	98.69	99.14
S832	98.34	98.34	98.17	97.95	98.39

Table VIII: Bit Flip after Compression for VIHC Coding

Benchmark circuit	**Fault Coverage**				**Max. Fault coverage (%)**
	1b	**2b**	**5b**	**10b**	
S27	98.00	64.00	67.00	85.00	98.00
S298	97.30	97.30	97.30	97.30	97.64
S344	99.00	99.00	99.00	99.00	99.28
S349	98.41	98.70	98.70	98.55	99.28
S382	97.81	97.81	97.81	97.81	97.94
S386	99.39	99.70	99.39	98.79	99.70
S400	97.34	97.34	97.47	97.47	97.59
S420	99.68	99.68	99.20	99.20	99.68
S444	97.33	97.33	97.33	97.33	98.22
S510	99.39	99.39	99.08	98.85	99.85
S526	97.52	97.52	97.52	97.52	97.95
S641	99.13	99.13	99.13	99.13	99.22
S713	99.23	99.23	99.23	99.23	99.23
S820	99.14	98.92	98.86	98.80	99.14
S832	98.34	98.34	98.28	98.17	98.39

6.2 Analysis

Consider the test data for a combinational (or full-scan sequential) circuit given by V test vectors, each vector is assumed to be m bits long. A compression technique often partitions the uncompressed test data into a set of allowed symbols $((s_i \mid 1) \leq I \leq$ S) where S is the total number of different symbols. A coding technique maps each symbol s_i to a corresponding codeword c_i The compressed bit-stream can be represented by an ordering Π of codewords, ($C_\Pi(1)$, $C_\Pi(2)$,.. $C_\Pi(N)$) , where N is the total number of symbols/codewords in the test data. This section analyzes the possible outcomes of bit-flips on the compressed test streams.

6.2.1 Propagation and Shifts

A bit-flip can change the compressed sequence in a complex manner; this is a function of different parameters such as compression ratio, number of symbols, length of symbols/codewords, and the ordering . However, a bit-flip always affects a codeword in one of the following manners.

o The affected codeword is broken into [k $\geq$ 1] valid codewords.

o The affected codeword is broken into k $\geq$ 0 valid codewords and a so-called "dangling suffix" which is not a valid codeword.

Consider the set of Huffman codewords given by 0,10, 110, 111. A bit-flip affecting codeword 0 results into a zero valid codeword and a dangling suffix of 1 which is not a valid codeword. A bit-flip affecting the first bit of codeword 10, results into 00, i.e., two valid codewords. A bit-flip at the second bit of codeword 111, results into 101, i.e., one valid codeword and a dangling suffix of 1 which is not valid.

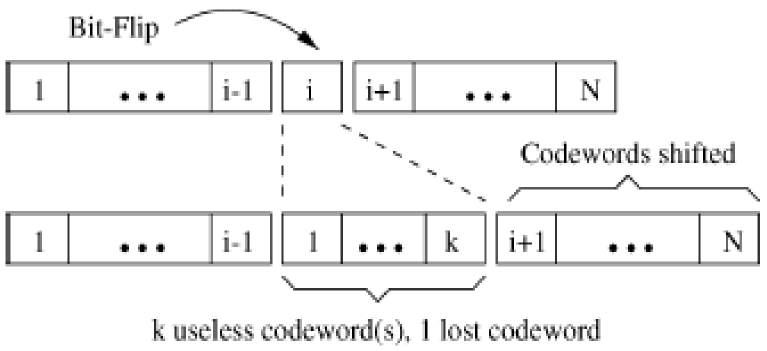


Figure 6.1: Conceptual view of the shift effect due to a bit-flip

The significant feature of the first case is that if a codeword C_{Π}(i) in the compressed bit stream is affected by a bit-flip, then all codewords C_{Π}(i+1) to C_{Π}(N) occurring next, will be unaffected. The only effect of the bit-flip is to have k additional codewords between C_{Π}(i-1) and C_{Π}(i+1) as a result of C_{Π}(i) being broken. In this case, the correct sequence of codewords starting from C_{Π}(i+1) is shifted by k codewords. This is shown in Fig. 6.1.

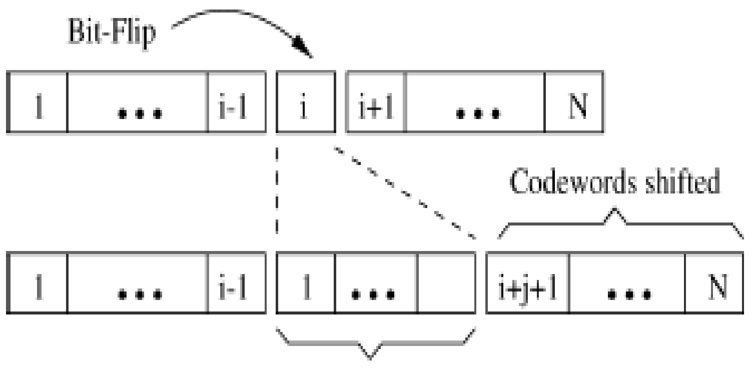


Figure 6.2: Conceptual view of the propagation effect due to a bit-flip.

The significant feature of the second case is that if a codeword C_{Π}(i) is affected by a bit-flip, the dangling suffix will form a valid codeword with a portion of C_{Π}(i+1). This effectively propagates the bit-flip to C_{Π}(i+1). If C_{Π}(i+1) is also left with a dangling suffix, the suffix will form a valid codeword with a portion of C_{Π}(i+2) and so on. If eventually C_{Π}(i+j) does not have a dangling suffix, error propagation will stop and C_{Π}(i+j+1) to C_{Π}(N) will be unaffected. However, the entire compressed stream from C_{Π}(i) to C_{Π}(i+j) is corrupted and, therefore, made useless. This is shown in Fig. 6.2.

A simple comparison between these two cases highlights the relationship between shift and propagation due to a bit-flip. In the first case, a codeword C_{Π}(i) is lost and the remaining codewords are shifted by at least one codeword (since k ¿ 1). In the second case, j+1 codewords (from C_{Π}(i) to C_{Π}(i+j)) are lost, and the remaining codewords are shifted by at least one codeword. Propagation and shift only differ in the number of lost (corrupted) codewords. A shift is a special case of propagation in which only one codeword is lost, i.e j=0. Propagation always comes with shifts.

An important observation is that a coding technique with constant length codewords will never have a propagation with j ¿ 0 independently of the number of bit-flips or their location. This occurs because a bit-flip can never break a constant length codeword into more than one valid codewords and a dangling suffix is never encountered. This means that run-length codes will have at most one lost (corrupt) codeword and a possible shift effect.

6.2.2 Synchronization Loss

Define the length of a propagation effect by the number of codewords which are lost, e.g., in the above discussion the length was j+1. On average the number of lost bits at decompression is j+1 times the average codeword length i.e. (j+1)lc$_{AVG}$ For the special case of shifts, the number of lost bits is lc$_{AVG}$. This lost segment of test stream is one contributing factor to coverage loss and a reduction in error-resilience. Another contributing factor is due to the Synchronization Loss. After decompression, I t is important that the unaffected sequence of bits (represented by codewords from C$_{\Pi}$(i+j+1) to C$_{\Pi}$(N)) appears exactly in the same location within the uncompressed test stream as it would if there was no bit-flip. In the presence of bit-flips, this sequence is preceded by a corrupt sequence of bits (represented by a number of corrupt codewords which have replaced the lost codewords from C$_{\Pi}$(i) to C$_{\Pi}$(i+j)). If the number of such corrupt bits is different from the number of otherwise correct bits in the uncompressed stream, then there is a loss of synchronization, i.e., the upcoming unaffected sequence of bits will be misplaced. Synchronization loss acts as if the unaffected sequence of bits is a random set of bits.

6.3 Improving Error-Resilience

The previous section highlights the importance of error resilience and the extend by which bit-flips can negatively impact test data quality as measured by fault coverage. This section deals with methodologies to increase error-resilience of the compressed test streams by traditional techniques.

6.3.1 Traditional Techniques

An approach which addresses the negative effects of bit-flips, is based on the use of error-correcting-codes (ECC); for example, the data downloaded from the ATE workstation to the test head is often coded using additional bits such asHamming code, parity or cyclic-redundancy-codes (CRC).

Chapter 7

Hamming Code based technique

In telecommunication, a Hamming code is a linear error-correcting code named after its inventor, Richard Hamming. Hamming codes can detect up to two simultaneous bit errors, and correct single-bit errors; thus, reliable communication is possible when the Hamming distance between the transmitted and received bit patterns is less than or equal to one. By contrast, the simple parity code cannot correct errors, and can only detect an odd number of errors.

In mathematical terms, Hamming codes are a class of binary linear codes. For each integer $m \geq 2$ there is a code with m parity bits and 2m - m - 1 data bits. The parity-check matrix of a Hamming code is constructed by listing all columns of length m that are pairwise independent. Hamming codes are an example of perfect codes, codes that exactly match the theoretical upper bound on the number of distinct code words for a given number of bits and ability to correct errors.

Because of the simplicity of Hamming codes, they are widely used in computer memory (RAM). In particular, a single-error-correcting and double-error-detecting variant commonly referred to as SECDED.

If more error-correcting bits are included with a message, and if those bits can be arranged such that different incorrect bits produce different error results, then bad bits could be identified. In a 7-bit message, there are seven possible single bit errors, so three error control bits could potentially specify not only that an error occurred but also which bit caused the error.

Hamming studied the existing coding schemes, including two-of-five, and generalized their concepts. To start with, he developed a nomenclature to describe the system, including the number of data bits and error-correction bits in a block. For instance, parity includes a single bit for any data word, so assuming ASCII words with 7-bits, Hamming described this as an (8,7) code, with eight bits in total, of which 7 are data. The repetition example would be (3,1), following the same logic. The code rate is the second number divided by the first, for our repetition example, 1/3.

Hamming also noticed the problems with flipping two or more bits, and described this as the "distance" (it is now called the Hamming distance, after him). Parity has a distance of 2, as any two bit flips will be invisible. The (3,1) repetition has a distance of 3, as three bits need to be flipped in the same triple to obtain another code word with no visible errors. A (4,1) repetition (each bit is repeated four times) has a distance of 4, so flipping two bits can be detected, but not corrected. When three bits flip in the same group there can be situations where the code corrects towards the wrong code word.

7.0.2 Generating Algorithm

General algorithm

The following general algorithm generates a single-error correcting (SEC) code for any number of bits.

1. Number the bits starting from 1: bit 1, 2, 3, 4, 5, etc.

2. Write the bit numbers in binary. 1, 10, 11, 100, 101, etc.

3. All bit positions that are powers of two (have only one 1 bit in the binary form of their position) are parity bits.

4. All other bit positions, with two or more 1 bits in the binary form of their position, are data bits.

5. Each data bit is included in a unique set of 2 or more parity bits, as determined by the binary form of its bit position.

a. Parity bit 1 covers all bit positions which have the least significant bit set: bit 1 (the parity bit itself), 3, 5, 7, 9, etc.

b. Parity bit 2 covers all bit positions which have the second least significant bit set: bit 2 (the parity bit itself), 3, 6, 7, 10, 11, etc.

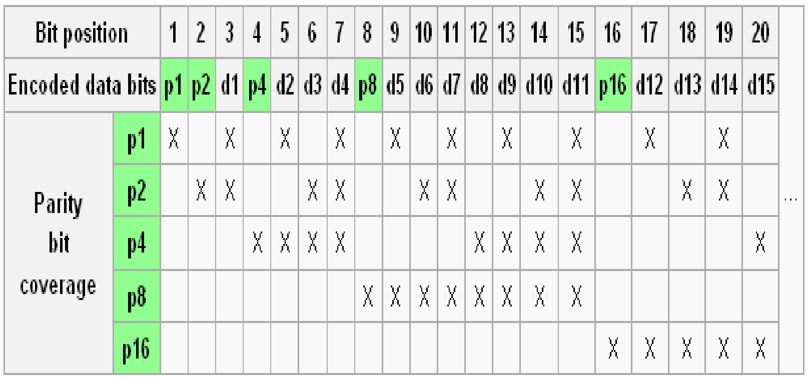
c. Parity bit 4 covers all bit positions which have the third least significant bit set: bits 47, 1215, 2023, etc.

d. Parity bit 8 covers all bit positions which have the fourth least significant bit set: bits 815, 2431, 4047, etc.

e. In general each parity bit covers all bits where the binary AND of the parity position and the bit position is non-zero.

The form of the parity is irrelevant. Even parity is simpler from the perspective of theoretical mathematics, but there is no difference in practice.

This general rule can be shown visually:



Bit position		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
Encoded data bits		p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8	d9	d10	d11	p16	d12	d13	d14	d15	
Parity bit coverage	p1	X		X		X		X		X		X		X		X		X		X		
	p2		X	X			X	X			X	X			X	X			X	X		...
	p4				X	X	X	X					X	X	X	X					X	
	p8								X	X	X	X	X	X	X	X						
	p16																X	X	X	X	X	

Figure 7.1: Hamming code example

Shown are only 20 encoded bits (5 parity, 15 data) but the pattern continues indefinitely. The key thing about Hamming Codes that can be seen from visual inspection is that any given bit is included in a unique set of parity bits. To check for errors, check all of the parity bits. The pattern of errors, called the error syndrome, identifies the bit in error. If all parity bits are correct, there is no error. Otherwise, the sum of the positions of the erroneous parity bits identifies the erroneous bit. For example, if the parity bits in positions 1, 2 and 8 indicate an error, then bit 1+2+8=11 is in error. If only one parity bit indicates an error, the parity bit itself is in error.

If, in addition, an overall parity bit (bit 0) is included, the code can detect (but not correct) any two-bit error, making a SECDED code. The overall parity indicates whether the total number of errors is even or odd. If the basic Hamming code detects an error, but the overall parity says that there are an even number of errors, an uncorrectable 2-bit error has occurred.

C1 = D1 xor D2 xor D4

C1 = 0 xor 1 xor 1

C1 = 0

C2 = D1 xor D3 xor D4

C2 = 0 xor 1 xor 1

C2 = 0

C3 = D2 xor D3 xor D4

C3 = 1 xor 1 xor 1

C3 = 1

So our Hamming code of 0111 becomes: 0001111

Now similarly to encoding, we must xor all bits whose row corresponds to the check bit in question. The difference is, this time we must also xor in the check bit. The result should be zero. If not, an error has occurred in transmission.

A1 = C1 xor D1 xor D2 xor D4

A1 = 0 xor 0 xor 1 xor 0

A1 = 1

A2 = C2 xor D1 xor D3 xor D4

A2 = 1 xor 0 xor 0 xor 0

A2 = 1

A3 = C3 xor D2 xor D3 xor D4

A3 = 1 xor 1 xor 0 xor 0

A3 = 0

We can see that all three A's are not equal to zero. This lets us know that there is an error in the code. It also tells us where exactly the error is. If you read the errors in reverse order, you will have the binary location of the error. Looking at this error, we can see there is an error in location three. We simply flip the bit, and we get the Huffman code: 1100

So the main advantage of Huffman code is that it removes single bit flip error so no loss of fault coverage but the biggest disadvantage is that it takes more area as shown in table I to VII.

Table I: Area Overhead with Huffman Code

Bench Mark Circuit	**Area of Bench mark cirduit**	**Area of Hamming Decoder**	**Area Overhead**
S27	21	383	1823
S298	134	4760	3552
S344	158	4514	2856
S349	155	4599	2967
S382	106	8214	7749
S386	124	4558	3675
S400	191	8369	4381
S420	156	9249	4847
S444	190	8106	4266
S510	232	4763	2053
S526	226	8992	3978

Table II: Area Overhead with selective Huffman Code

BenchMark Circuit	**Area of Benchmark circuit**	**Area of Hamming Decoder**	**Area Overhead**
S27	21	588	2800
S298	134	4939	3685
S344	158	4650	2943
S349	155	4856	3132
S382	106	9054	8541
S386	124	4530	3653
S400	191	9185	4808
S420	156	17258	11062
S444	190	8994	4733
S510	232	4718	2033
S526	226	9282	4107
S820	323	8464	2620
S832	333	8676	2605

Table III: Area Overhead with FDR Code

Benchmark Circuit	**Area of Benchmark circuit**	**Area of Hamming Decoder**	**Area Overhead**
S27	21	789	3757
S298	134	8484	6331
S344	158	4987	3156
S349	155	8488	5476
S382	106	8881	8378
S386	124	4463	3599
S400	191	9504	4975
S420	156	17348	11120
S444	190	8881	4674
S510	232	4998	2154
S526	226	17149	7588
S641	200	9292	4646
S713	201	8330	4144
S820	323	8180	2532
S832	333	8708	2615

Table IV: Area Overhead with EFDR Code

Benchmark Circuit	**Area of Benchmark circuit**	**Area of Hamming Decoder**	**Area Overhead**
S27	21	833	3966
S298	134	8197	6117
S344	158	8048	5093
S349	155	8291	5349
S382	106	17524	16532
S386	124	4463	3599
S400	191	18351	9607
S420	156	17918	11485
S444	190	18003	9475
S510	232	8251	3556
S526	226	17938	7937
S641	200	17515	8757
S713	201	9485	4718
S820	323	9223	2855
S832	333	9234	2772

Table V: Area Overhead with SAFDR Code

Benchmark Circuit	**Area of Benchmark circuit**	**Area of Hamming Decoder**	**Area Overhead**
S27	21	838	3990
S298	134	8056	6011
S344	158	7987	5055
S349	155	8156	5261
S382	106	17397	16412
S386	124	4463	3599
S400	191	17906	9374
S420	156	17544	11246
S444	190	17397	9156
S510	232	8587	3701
S526	226	17399	7698
S641	200	17527	8763
S713	201	9280	4616
S820	323	9100	2817
S832	333	9096	2731

Table VI: Area Overhead with GOLOMB Code

Benchmark Circuit	**Area of Benchmark circuit**	**Area of Hamming Decoder**	**Area Overhead**
S27	21	964	4590
S298	134	8658	6461
S344	158	7960	5037
S349	155	7885	5087
S382	106	17030	16066
S386	124	4463	3599
S400	191	17795	9316
S420	156	17721	11359
S444	190	17030	8963
S510	232	8639	3723
S526	226	17814	7882
S641	200	17084	8542
S713	201	9140	4547
S820	323	9060	2804
S832	333	9042	2715

Table VII: Area Overhead with VIHC Code

Benchmark Circuit	**Area of Benchmark circuit**	**Area of Hamming Decoder**	**Area Overhead**
S27	21	659	3138
S298	134	4871	3635
S344	158	4504	2850
S349	155	4610	2974
S382	106	8820	8320
S386	124	4463	3599
S400	191	9258	4847
S420	156	8968	5748
S444	190	8193	4312
S510	232	4828	2081
S526	226	9073	4014
S641	200	8826	4413
S713	201	7779	3870
S820	323	8409	2603
S832	333	8348	2506

Chapter 8

Cyclic redundancy check (CRC)

A cyclic redundancy check (CRC) is an error-detecting code designed to detect accidental changes to raw computer data, and is commonly used in digital networks and storage devices such as hard disk drives. A CRC-enabled device calculates a short, fixed-length binary sequence, known as the check value or improperly the CRC, for each block of data to be sent or stored and appends it to the data, forming a codeword. When a codeword is received or read, the device either compares its check value with one freshly calculated from the data block, or equivalently, performs a CRC on the whole codeword and compares the resulting check value with an expected residue constant. If the check values do not match, then the block contains a data error and the device may take corrective action such as rereading or requesting the block be sent again, otherwise the data is assumed to be error-free (though, with some small probability, it may contain undetected errors; this is the fundamental nature of error-checking).[1]

CRCs are so called because the check (data verification) code is a redundancy (it adds zero information to the message) and the algorithm is based on cyclic codes. CRCs are popular because they are simple to implement in binary hardware, are easy to analyze mathematically, and are particularly good at detecting common errors caused by noise in transmission channels.

CRCs are based on the theory of cyclic error-correcting codes. The use of systematic cyclic codes, which encode messages by adding a fixed-length check value, for the purpose of error detection in communication networks was first proposed by W. Wesley Peterson in 1961.[2] Cyclic codes are not only simple to implement but have the benefit of being particularly well suited for the detection of burst errors, contiguous sequences of erroneous data symbols in messages. This is important because burst errors are common transmission errors in many communication channels, including magnetic and optical storage devices. Typically, an n-bit CRC, applied to a data block of arbitrary length, will detect any single error burst not longer than n bits, and will detect a fraction 1-2-n of all longer error bursts.

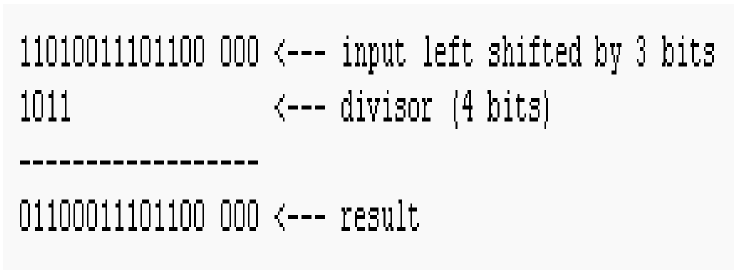
8.0.3 Computation of CRC

To compute an n-bit binary CRC, line the bits representing the input in a row, and position the (n+1)-bit pattern representing the CRC's divisor (called a "polynomial") underneath the left-hand end of the row.

Start with the message to be encoded:

11010011101100

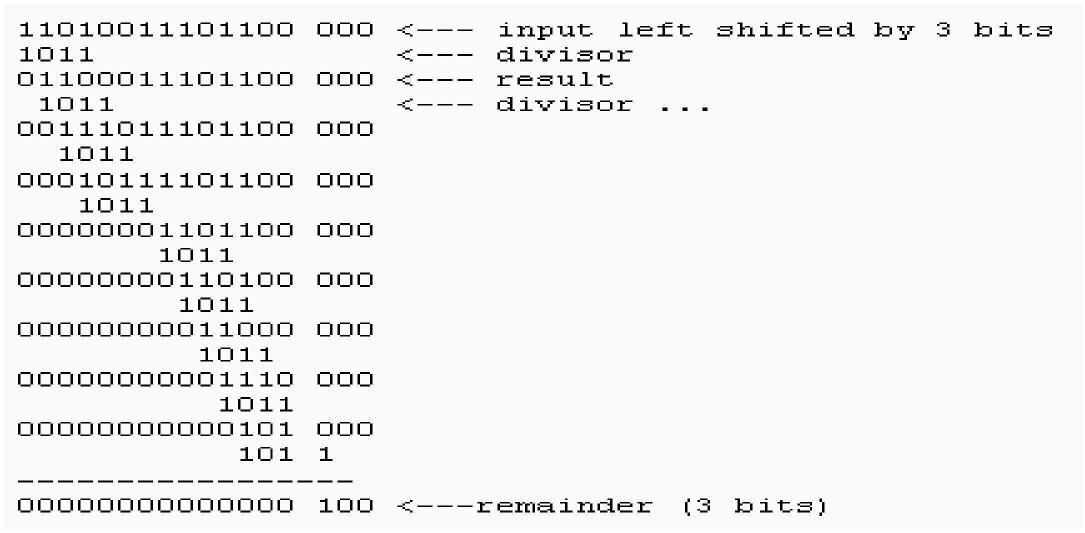
This is first padded with zeroes corresponding to the bit length n of the CRC. Here is the first calculation for computing a 3-bit CRC:



```
11010011101100 000 <--- input left shifted by 3 bits
1011               <--- divisor (4 bits)
------------------
01100011101100 000 <--- result
```

Figure 8.1: Padded bits with CRC

If the input bit above the leftmost divisor bit is 0, do nothing and move the divisor to the right by one bit. If the input bit above the leftmost divisor bit is 1, the divisor is XORed into the input (in other words, the input bit above each 1-bit in the divisor is toggled). The divisor is then shifted one bit to the right, and the process is repeated until the divisor reaches the right-hand end of the input row. Here is the entire calculation:

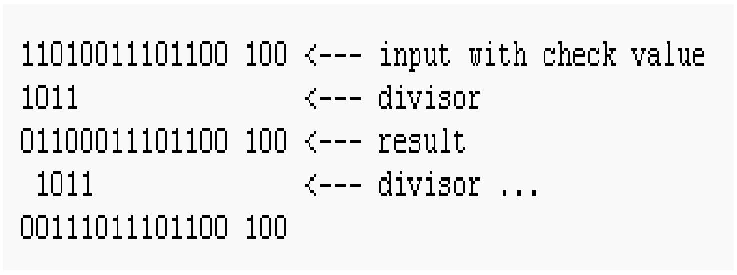


```
11010011101100 000 <--- input left shifted by 3 bits
1011               <--- divisor
01100011101100 000 <--- result
 1011              <--- divisor ...
00111011101100 000
  1011
00010111101100 000
   1011
00000001101100 000
       1011
00000000110100 000
        1011
00000000011000 000
         1011
00000000001110 000
          1011
00000000000101 000
           101 1
-----------------
00000000000000 100 <---remainder (3 bits)
```

Figure 8.2: Computation if CRC

Since the leftmost divisor bit zeroed every input bit it touched, when this process ends the only bits in the input row that can be nonzero are the n bits at the right-hand end of the row. These n bits are the remainder of the division step, and will also be the value of the CRC function (unless the chosen CRC specification calls for some postprocessing).

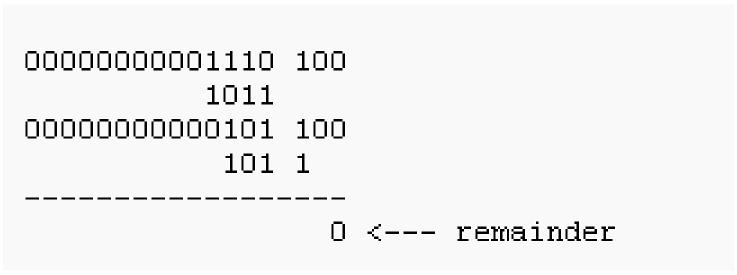
The validity of a received message can easily be verified by performing the above calculation again, this time with the check value added instead of zeroes. The remainder should equal zero if there are no detectable errors.



```
11010011101100 100 <--- input with check value
1011               <--- divisor
01100011101100 100 <--- result
 1011              <--- divisor ...
00111011101100 100
```

Figure 8.3: Rules for divisor

and so on until:



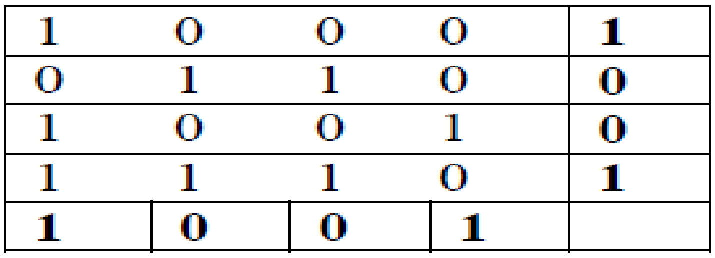
```
00000000001110 100
          1011
00000000000101 100
           101 1
------------------
                 0 <--- remainder
```

Figure 8.4: Correctness of CRC Check

Chapter 9

Two dimensional Parity check Code

In this method the dataword is organized in table as shown in figure 9.1. The data to be sent are arranged in separate rows 4 bit each. For each row and column 1 parity check bit is calculated as shown in figure 9.1 last row and last column . The whole table is then sent to the receiver which finds syndrome for each row and each column as shown in figure 9.2. If there is parity missmatch then it can be said that intersection of that row and column gives error.



1	0	0	0	1
0	1	1	0	0
1	0	0	1	0
1	1	1	0	1
1	0	0	1	

Figure 9.1: Two dimensional Parity check Matrix

The biggest disadvantage of such traditional technique is its increased extra bit count as compared to its original compressed bits. Bit overhead comparision is shown in table I to VIII.

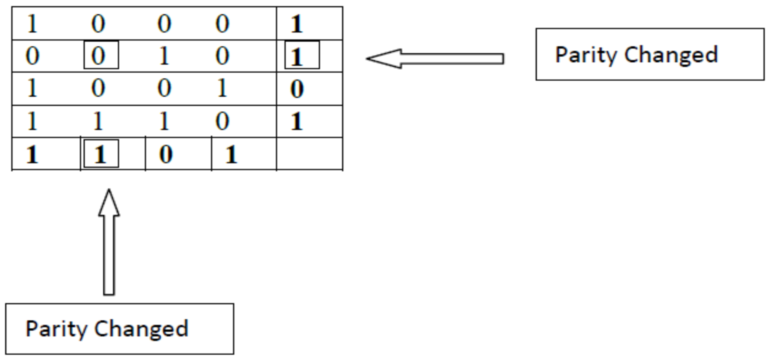


Figure 9.2: Effect of error on Two dimensional Parity check Matrix

Table I: Bits Overhead with Huffman Code

Benchmark circuit	**compressed bits Original**	**Compressed Bits with Hamming**	**Compressed Bits with Parity Check**	**Compressed Bits with CRC**
S27	18	35	27	39
S298	406	714	514	718
S344	390	686	494	690
S349	420	735	529	739
S382	714	1253	899	1257
S386	300	525	379	529
S400	798	1400	998	1404
S420	928	1624	1164	1628
S444	735	1288	785	1292
S510	402	707	503	711
S526	833	1463	1042	1467

Table II: Bits Overhead with Selective Huffman Code

Benchmark circuit	**compressed bits Original**	**Compressed Bits with Hamming**	**Compressed Bits with Parity Check**	**Compressed Bits with CRC**
S27	23	42	34	46
S298	501	882	634	886
S344	440	770	554	774
S349	478	840	604	844
S382	812	1421	1019	1425
S386	390	686	494	690
S400	917	1610	1151	1614
S420	1136	1988	1424	1992
S444	837	1470	1054	1474
S510	444	777	559	781
S526	930	1631	1169	1635
S820	562	987	709	991
S832	566	994	714	998
S838	4400	7700	5504	7704

Table III: Bits Overhead with FDR Code

Benchmark circuit	**compressed bits Original**	**Compressed Bits with Hamming**	**Compressed Bits with Parity Check**	**Compressed Bits with CRC**
S27	32	56	44	60
S298	522	917	659	923
S344	496	868	624	872
S349	534	938	674	942
S382	892	1561	1119	1565
S400	1010	1771	1269	1775
S420	1190	2086	1492	2090
S444	890	1561	1119	1565
S510	504	882	634	886
S526	1060	1855	1329	1859
S641	912	1596	1144	1600
S713	756	1323	949	1327
S820	694	1218	874	1222
S832	666	1169	839	1173

Table IV: Bits Overhead with EFDR Code

Benchmark circuit	**compressed bits Original**	**Compressed Bits with Hamming**	**Compressed Bits with Parity Check**	**Compressed Bits with CRC**
S27	39	70	54	74
S298	691	1211	869	1215
S344	701	1232	884	1236
S349	737	1295	929	1299
S382	1250	2191	1569	2195
S400	1630	2856	2044	2860
S420	1386	2429	1739	2433
S444	1338	2345	1679	2349
S510	684	1197	859	1201
S526	1434	2513	1799	2517
S641	1265	2219	1586	2223
S713	1011	1771	1269	1775
S820	918	1610	1154	1614
S832	934	1638	1174	1642

Table V: Bits Overhead with SAFDR Code

Benchmark circuit	**compressed bits Original**	**Compressed Bits with Hamming**	**Compressed Bits with Parity Check**	**Compressed Bits with CRC**
S27	37	70	54	74
S298	601	1057	759	1061
S344	609	1071	769	1075
S349	649	1141	819	1145
S382	1095	1918	1375	1922
S400	1197	2471	1504	2475
S420	1409	2100	1769	2104
S444	1129	1981	1419	1985
S510	573	1008	724	1012
S526	1239	2170	1554	2174
S641	1115	1953	1399	1957
S713	911	1596	1144	1600
S820	809	1421	1019	1425
S832	795	1393	999	1397

Table VI: Bits Overhead with GOLOMB Code

Benchmark circuit	**compressed bits Original**	**Compressed Bits with Hamming**	**Compressed Bits with Parity Check**	**Compressed Bits with CRC**
S27	37	70	54	74
S298	623	1092	784	1096
S344	602	1057	759	1061
S349	647	1134	814	1138
S382	1050	1841	1319	1845
S400	1425	2499	1789	2503
S420	1211	2121	1519	2125
S444	1049	1841	1319	1845
S510	618	1085	779	1089
S526	1275	2233	1599	2237
S641	1080	1890	1354	1894
S713	865	1519	1089	1523
S820	820	1435	1029	1439
S832	819	1435	1029	1439

Table VII: Bits Overhead with VIHC Code

Benchmark circuit	**compressed bits Original**	**Compressed Bits with Hamming**	**Compressed Bits with Parity Check**	**Compressed Bits with CRC**
S27	25	49	39	53
S298	407	714	514	718
S344	391	686	494	690
S349	420	735	529	739
S382	715	1253	899	1257
S400	930	1631	1169	1635
S420	799	1400	1004	1404
S444	736	1288	924	1292
S510	402	707	509	711
S526	841	1477	1059	1481
S641	714	1253	899	1257
S713	629	1106	794	1110
S820	546	959	689	963
S832	550	966	694	970

Chapter 10

Conclusion and Future Scope

10.1 Conclusion

In this dissertation, we proposed Artificial Intelligence approach for Test vector reordering for Low Power Testing ,then this approach is enhanced for scan in scan out power reduction by using WTM based reordering then finally one new approach was proposed that is combination of frequency directed and MT fill algorithm.

From the results it can be seen that average pecentage reduction in switching activity is 22.11 % with Artificial Intelligence approach.

Also it is concluded that MT Fill algorithm gives low power but low compression, Frequency directed approach gives high power and high compression, and Mixed approach gives moderate power and moderate compression.

Final part has presented the importance of error-resilience for reliable compression of test data in manufacturing testing of VLSI chips and systems. It has been evaluated that bit-flips in uncompressed test streams can negligibly impact fault coverage, while application of test data compression can seriously degrade test quality when such bit-flips occur. An analysis of the errors caused by bit-flips is presented to indicate how bit-flips can lead to change in test data stream and coverage loss.

Traditional techniques such as Hamming code based algorithm have been proposed to improve error-resilience by padding extra bits with the fixed length of code. The goal is to make the transmission error free without any loss of fault coverage. The biggest disadvantage with this technique is the area overhead and Bits overhead.

10.2 Future Scope

In future new techniques will be developed to achieve improved error resilience with less area overhead and less bit overhead.

References

[1] D. Whitley A. Sokolov, A. Sanyal and Y. Malaiya. Dynamic power minimization during combinational circuit testing as a traveling salesman problem. *Proc. of the IEEE Congress on Evolutionary Computation*, vol. 2, pp. 10881095, 2005.

[2] Nourani M Ahmed N, Tehranipour M.H. Low power pattern generation for bist architecture. *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems*, Vol.2, pp. 689-692, 2004.

[3] Sheng Zhang Bhargab B. Bhattacharya, Sharad C. Seth. Low- energy bist design for scan-based logic circuits. *16th International Conference on VLSI Design*, pp. 546 551, 2003.

[4] Mahmoodi H Ghosh D Roy K Bhunia, S. Power reduction in test-per-scan bist with supply gating and efficient scan partitioning. *Sixth International Symposium on Quality of Electronic Design*, pp. 453 458, 2005.

[5] Zhao-lin Li-Xin-chun Wu Rui Ke Bin Zhou, Yi-zheng Ye. A new low power test pattern generator using a variable-length ringcounter. *IEEE conference on Quality of Electronic Design*, pp.248 252, 2009.

[6] Landrault C-Pravossoudovitch Bonhomme Y, Girard P. Test power: a big issue in large soc designs. *Proceedings of The first IEEE workshop on Electronic Design, Test and Applications*, pp. 447 449, 2002.

[7] S. Chattopadhyay and N. Choudhary. Genetic algorithm based approach for low power combinational circuit testing. *Proc. of the VLSID*, pp. 552559, 2003.

[8] Xiang D-Yin B Chen Z, Feng J. Scan chain configuration based x-filling for low power and high quality testing. *IET Journal on Computers and Digital Techniques*, Vol. 4 pp. 1 - 13.

[9] M. Ravikumar C.P. ElShoukry, M. Tehranipoor. Partial gating optimization for power reduction during test application. *Proceedings of 14th Asian Test Symposium*, pp 242. - 247, 2005.

[10] Li gang Hou; Xiao-hong Peng; Wu-chen Wu. A low power dynamic pseudo random bit generator for test pattern generation. *Proceedings of 9th International*

Conference on Solid-State and Integrated-Circuit Technology, pp. 2079 2082, 2008.

[11] Lavanya P. Subashini-Umamageswaran M Gunavathi K, Paramasivam K. A novel bist tpg for testing of vlsi circuits. *Proceedings of First International Conference on Industrial and Information Systems*, pp. 109 114, 2006.

[12] H. Hashempour and F. Lombardi. Evaluation of heuristic techniques for test vector ordering. *Proc. of the GLSVLSI*, pp. 9699, 2004.

[13] M.; Atoofian E.; Navabi-Z.; Afzali-Kusha A. Hatami, S.; Alisafaee. A low-power scan-path architecture. *IEEE International Symposium on Circuits and Systems*, Vol. 5, pp. 5278 5281, 2005.

[14] Nicolici N. Ho Fai Ko. Automated scan chain division for reducing shift and capture power during broadside at-speed test. *IEEE Transactions on Computer-Aided design of Integrated Circuits and Systems*, Vol. 27,pp. 2092 - 2097, 2008.

[15] T. Huang and K. Lee. An input control technique for power reduction in scan circuits during test application. *Proc. Asian Test Symposium*, pp. 315-320, 1999.

[16] Wunderlich H.-J-Maeding N-Leenstra J Imhof M.E, Zoellin C.G. Scan test planning for power reduction. *44th ACM/IEEE conference on Design Automation*, pp. 521 526, 2007.

[17] Qiang Xu Jing-Ling Yang. State-sensitive x-filling scheme for scan capture power reduction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol.27 , pp. 1338 1343, 2008.

[18] Oktem L Kilic H. Low-power test pattern generator design for bist via non-uniform cellular automata. *IEEE VLSI-TSA International Symposium on VLSI Design, Automation and Test*, pp. 212 215, 2005.

[19] Chattopadhyay S Kundu S. Efficient don't care filling for power reduction during testing. *IEEE International Conference on Advances in Recent Technologies in Communication and Computing*, pp. 319 - 323, 2009.

[20] Li J.C.-M Lin H.-T. Simultaneous capture and shift power reduction test pattern generator for scan testing. *IET Journal on Computers and Digital Techniques*, Vol. 2, pp.132 141, 2008.

[21] Liu C-Chun-Lin Y Lu S, Lu Y. Low-power built-in self-test techniques for embedded srams. *VLSI Design archive*, Volume 2007, pp.1 6, April 2007.

[22] T.K. Chattopadhyay Maiti. Dont care filling for power minimization in vlsi circuit testing. *IEEE International Symposium on Circuits and Systems*, pp. 2637 2640, 2008.

[23] Li M Min-Hao C. Jump scan: a dft technique for low power testing. *Proceedings 23rd IEEE VLSI Test Symposium*, pp. 277 - 282, 2005.

[24] S. Pravossoudovitch-C. Landrault N. Badereddine, P. Girard and A. Virazel. Minimizing peak power consumption duringscan testing: Test pattern modification with x filling heuristics. *International Conference on Design and Test of Integrated Systems in Nanoscale Technology*, pp 359-364, September 2006.

[25] N. Nicola and B. M. Al-Hashimi. *Power-Constrained Testing of VLSI Circuits.* Kluwer Academic, 2003.

[26] Ahmed N Nourani M, Tehranipoor M. Low-transition test pattern generation for bist-based applications. *IEEE Transactions on Computers*, Vol.57, pp. 303 315.

[27] Girard P. Survey of low-power testing of vlsi circuits. *IEEE Journal on Design and and Test of Computers*, vol. 19,pp. 80 - 90, 2002.

[28] H. Neto-J. Monteiro P. Flores, J. Costa and J. Marques-Silva. Assignment and reordering of incompletely specified pattern sequences targeting minimum power dissipation. *Proc. of the VLSID*, pp. 3741, 1999.

[29] J.Neto J.Monteiro P. Flores, J. Costa and J. Silva. Assignment and reordering of incompletely specified pattern sequences targeting minimum power dissipation. *Proc. IEEE Intl. Conf. on VLSI*, pp. 37-41, 1999.

[30] S. Pravossoudovitch P. Girard, C. Landrault and D. Severac. Reducing power consumption during test application by test vector ordering. *Proc. of the ISCAS*, pp. 296299, 1998.

[31] Sathishkumar P Paramasivam K, Gunavathi K. Algorithm for low power combinational circuit testing. *Proceedings of IEEE Region 10 Conference*, Vol. 4, pp. 336 - 339, 2004.

[32] Kundu. S Becker. B Polian.I, Czutro. A. Power droop testing. *International IEEE conference on Computer Design*, pp. 243 - 250, 2006.

[33] Dong X Qiang X, Dianwei H. Pattern-directed circuit virtual partitioning for test power reduction. *Proceedings of IEEE International Test Conference*, pp.1 10, 2007.

[34] Pal A Roy S, Sen Gupta I. Artificial intelligence approach to test vector reordering for dynamic power reduction during vlsi testing. *Proceedings of IEEE Region 10 Conference*, pp.1- 6, 2008.

[35] M. Hosseinabady A. Afzali-Kusha S. Sharifi, J. Jaffari and Z. Navabi. Scan based structure with reduced static and dynamic power consumption. *Journal of Low Power Electronics, American Scientific Publishers*, vol. 2, No. 3, pp. 477-487, 2006.

[36] Feb Seongmoon Wang, Sandeep k.Gupta. Atpg for heat dissipation minimization during test application. *IEEE Trans. On computer*, Vol: 47, pp 256-262, 1998.

[37] Orailoglu A Sinanoglu, O. Aggressive test power reduction through teststimuli transformation. *Proceedings 21st IEEE International Conference on Computer Design*, pp. 542 - 547, 2003.

[38] M.Hosseinabady A.Afzali-Kusha S.Sharifi, J.Jaffari and Z. Navabi. Simultaneous reduction of dynamic and static power in scan structures. *Proceedings of the conference on Design, Automation and Test*, Vol.2 ,pp. 846 - 851, 2005.

[39] S.Wang and S. K. Gupta. Atpg for heat dissipation minimization during test application. *IEEE Trans. on Computers*, pp. 256-262, February 1998.

[40] Shu-Min Li K Sying-Jyan Wang, Yan-Ting Chen. Low capture power test generation for launch-off-capture transition test based on don't-care filling. *IEEE International Symposium on Circuits and Systems*, pp. 3683 - 3686, 2007.

[41] Li K.S.-M. Sying-Jyan Wang; Kuo-Lin Fu. Low peak power atpg for n-detection test. *IEEE International Symposium on Circuits and Systems*, pp. 1993 - 1996, 2009.

[42] I. Pomeranz V. Dabholkar, S. Chakravarty and S. Reddy. Techniques for minimizing power dissipation in scan and combinational circuits during test application. *IEEE TCAD*, pp. 13251333, 1998.

[43] Bei Cao; Liyi Xiao; Yongsheng Wang. A low power deterministic test pattern generator for bist based on cellular automata. *4th IEEE International Symposium on Electronic Design, Test and Applications*, pp.266 269, 2008.

[44] Seongmoon Wang. A bist tpg for low power dissipation and high fault coverage. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol.15, pp. 777 789, 2007.

[45] Abramovici. M Xiaoming Y. Sequential circuit atpg using combinational algorithms. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1294 1310, 2005.

[46] Yong Lee Sungho Kang Youbean Kim, Myung-Hoon Yang. A new low power test pattern generator using a transition monitoring window based on bist architecture. *Proceedings of 14th Asian Test Symposium*, pp. 230 235, 2005.

[47] Andre I Yuejian W. Low power soc memory bist. *21st IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems*, pp.197-205, 2006.

References