Algorithms for Power Aware Testing of Nanometer Digital ICs

A THESIS

submitted by

A. SATYA TRINADH

for the award of the degree

of

DOCTOR OF PHILOSOPHY

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY HYDERABAD

Nov 2016

Declaration

I declare that this written submission represents my ideas in my own words, and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

A. Satya Trinadh

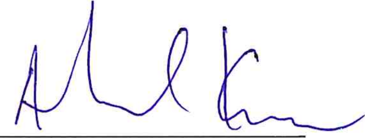
A. SATYA TRINADH

CS11P1001

Acad-m-14843/21/5/2018

Approval Sheet

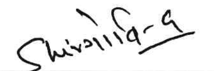
This thesis entitled **Algorithms for Power Aware Testing of Nanometer Digital ICs** by (A. Satya Trinadh) is approved for the degree of Doctor of Philosophy from IIT Hyderabad.



Prof. Anshul Kumar
IIT Delhi
Examiner

Dr. M V Panduranga Rao
IIT Hyderabad
Internal Examiner

Dr. Ch Sobhan Babu
IIT Hyderabad
Adviser/Guide



Dr. Shiv Govind Singh
IIT Hyderabad
Co-Adviser

Dr. V Chandrika Prakash
IIT Hyderabad
Chairman

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my research advisors Dr. Ch. Sobhan Babu and Dr. Shiv Govind Singh for their continuous advice and encouragement throughout my research.

I would like to thank my doctoral committee members Dr. M.V. Panduranga Rao, Dr. Naveen Sivadasan and Dr. Asudeb Dutta for providing constructive criticism and encouragement during my doctoral committee meetings.

My sincere thanks to Prof. V. Kamakoti from IIT Madras for teaching me Digital Systems Testing, proving access to facilities in Reconfigurable Intelligent Systems Engineering (RISE) lab and helping me during struggling phase of my research.

I am grateful to Seetal Potluri from RISE Lab, IIT Madras for his care and precious friendship during my stay at IIT Madras. I am thankful to him for consistently helping me throughout my research and for the support to make this thesis possible.

I would like to thank my friend Valeti Nageswara Rao who helped me during my research work.

I offer my sincere thanks to all my research colleagues at Software Systems Research Lab (SSRL) Lab for all the help and cooperation.

Finally, I take this opportunity to express the profound gratitude from my deep heart to my beloved parents, brothers, wife and kids for their love, continuous encouragement and support.

This thesis is heartily dedicated to my parents who took the lead to heaven before the completion of this work.

ABSTRACT

KEYWORDS: Digital system testing, At-speed testing, Capture power, Test Vector Ordering, X-filling

At-speed testing of deep-submicron digital very large scale integrated (VLSI) circuits has become mandatory to catch small delay defects. Now, due to continuous shrinking of complementary metal oxide semiconductor (CMOS) transistor feature size, power density grows geometrically with technology scaling. Additionally, power dissipation inside a digital circuit during the testing phase (for test vectors under all fault models (Potluri, 2015)) is several times higher than its power dissipation during the normal functional phase of operation. Due to this, the currents that flow in the power grid during the testing phase, are much higher than what the power grid is designed for (the functional phase of operation). As a result, during at-speed testing, the supply grid experiences unacceptable supply IR-drop, ultimately leading to delay failures during at-speed testing. Since these failures are specific to testing and do not occur during functional phase of operation of the chip, these failures are usually referred to false failures, and they reduce the yield of the chip, which is undesirable.

In nanometer regime, process parameter variations has become a major problem. Due to the variation in signalling delays caused by these variations, it is important to perform at-speed testing even for stuck faults, to reduce the test escapes (McCluskey and Tseng, 2000; Vorisek *et al.*, 2004). In this context, the problem of excessive peak power dissipation causing false failures, that was addressed previously in the context of at-speed transition fault testing (Saxena *et al.*, 2003; Devanathan *et al.*, 2007*a*,*b*,*c*), also becomes prominent in the context of at-speed testing of stuck faults (Maxwell *et al.*, 1996; McCluskey and Tseng, 2000; Vorisek *et al.*, 2004; Prabhu and Abraham, 2012; Potluri, 2015; Potluri *et al.*, 2015). It is well known that excessive supply IR-drop during at-speed testing can be kept under control by minimizing switching activity during testing (Saxena *et al.*, 2003). There is a rich collection of techniques proposed in the past for reduction of peak switching activity during at-speed testing of transition/delay faults

in both combinational and sequential circuits. As far as at-speed testing of stuck faults are concerned, while there were some techniques proposed in the past for combinational circuits (Girard *et al.*, 1998; Dabholkar *et al.*, 1998), there are no techniques concerning the same for sequential circuits. *This thesis addresses this open problem.* We propose algorithms for minimization of peak switching activity during at-speed testing of stuck faults in sequential digital circuits under the combinational state preservation scan (CSP-scan) architecture (Potluri, 2015; Potluri *et al.*, 2015). First, we show that, under this CSP-scan architecture, when the test set is completely specified, the peak switching activity during testing can be minimized by solving the Bottleneck Traveling Salesman Problem (BTSP). This mapping of *peak test switching activity minimization problem* to *BTSP* is novel, and proposed for the first time in the literature.

Usually, as circuit size increases, the percentage of don't cares in the test set increases. As a result, test vector ordering for any arbitrary filling of don't care bits is insufficient for producing effective reduction in switching activity during testing of large circuits. Since don't cares dominate the test sets for larger circuits, don't care filling plays a crucial role in reducing switching activity during testing. Taking this into consideration, we propose an algorithm, *XStat*, which is capable of performing test vector ordering while preserving don't care bits in the test vectors, following which, the don't cares are filled in an intelligent fashion for minimizing input switching activity, which effectively minimizes switching activity inside the circuit (Girard *et al.*, 1998). Through empirical validation on benchmark circuits, we show that *XStat* minimizes peak switching activity significantly, during testing.

Although *XStat* is a very powerful heuristic for minimizing peak input-switching-activity, it will not guarantee optimality. To address this issue, we propose an algorithm that uses *Dynamic Programming* to calculate the lower bound for a given sequence of test vectors, and subsequently uses a *greedy strategy* for filling don't cares in this sequence to achieve this lower bound, thereby guaranteeing optimality. This algorithm, which we refer to as *DP-fill* in this thesis, provides the *globally optimal* solution for minimizing peak input-switching-activity and also is the best known in the literature for minimizing peak input-switching-activity during testing. The proof of optimality of *DP-fill* in minimizing peak input-switching-activity is also provided in this thesis.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS **i**

ABSTRACT **ii**

LIST OF TABLES **vii**

LIST OF FIGURES **viii**

ABBREVIATIONS **x**

NOTATIONS **xi**

1 Introduction **1**

1.1 Power issues during at-speed testing 1

1.2 Reduction of peak power during testing of combinational circuits . . 3

1.3 Reduction of peak power during testing of sequential circuits 4

2 Background **7**

2.1 Related work in low power testing 8

2.2 Motivation for at-speed stuck-at testing 9

2.3 Design for Testability 10

2.3.1 Enhanced Scan (ES) scheme 11

2.3.2 CSP-scan scheme 12

3 An Efficient Test Vector Ordering Algorithm for Minimizing Peak Switching Activity **15**

3.1 PITMP and BTSP 15

3.2 Mapping of PITMP vs BTSP 17

3.2.1 Mapping of PITMP to BTSP 18

3.2.2 Mapping of BTSP to PITMP 18

3.3 Algorithm for BTSP 19
3.4 Experimental Results 22
3.4.1 Experimental Setup 22
3.4.2 Results 23
3.5 Summary 33

4 An Efficient X-filling algorithm for Minimizing Peak Switching Activity 37

4.1 Peak Input Toggle Minimization Problem (PITMP) 38
4.2 Balanced X-Filling (B-Fill) Algorithm 38
4.2.1 Motivation 38
4.2.2 Existing X-Filling Techniques 39
4.2.3 Algorithm Details 40
4.3 Test Cube Ordering Algorithm 41
4.3.1 The Need for an Efficient Test Cube Order 43
4.3.2 The X-Based Ordering Algorithm 44
4.3.3 Effectiveness of *X-Based Ordering Algorithm* 44
4.4 Integrated Test Vector Ordering and X-filling Algorithm 47
4.5 Experimental Results 49
4.5.1 Experimental Setup 49
4.6 Summary 51

5 An Optimal X-Filling algorithm for Minimizing Peak Switching Activity 55

5.1 Peak Input Toggle Minimization Problem (PITMP) 55
5.2 Bottleneck Coloring Problem (BCP) 56
5.2.1 Problem Statement 56
5.2.2 Dynamic Programming Algorithm to compute Lower-Bound 57
5.2.3 Greedy Algorithm for *Bottleneck Coloring Problem* 57
5.2.4 Proof of correctness 58
5.3 Optimal X-Filling Algorithm 59
5.3.1 Motivation 59
5.3.2 Algorithm Details 59
5.4 Test Vector Ordering Algorithm 60

5.4.1 Motivation 60
5.4.2 Algorithm Details 60
5.4.3 Experimental Results 61
5.5 Bottleneck Minimization Algorithm 61
5.6 Experimental Results 62
5.6.1 Experimental Setup 62
5.6.2 Results 62
5.7 Local Search With Iterative 1-bit Neighbourhood 64
5.8 Summary 76

6 Conclusions **77**
6.1 Test vector ordering for fully specified test sets 77
6.2 Simultaneous test vector ordering and don't care filling 78
6.3 An optimal algorithm for peak input switching activity 79
6.4 Future Work 79

LIST OF TABLES

2.1 ITC'99 Benchmarks (X % : Average % of X-bits in test cubes) . . . 14

3.1 ITC'99 Benchmarks 28

3.2 Edge cost : Primary input toggles per vector pair 29

3.3 Edge cost : Circuit total toggles per vector pair 30

3.4 Edge cost : Circuit total power (in μW) per vector pair 31

3.5 Impact of test vector ordering for different cost functions considered 32

3.6 Peak power comparisons for different cost functions considered . . 34

3.7 Average power comparisons for different cost functions considered . 35

4.1 ITC'99 benchmarks (X % : Average % of X-bits in test cubes) . . . 37

4.2 Lookup table for X-filling 40

4.3 Peak input toggles : Tool-Ordering with different X-fillings 49

4.4 Peak input toggles : BTSP-Ordering followed by different X-fillings 50

4.5 Peak input toggles : X-Base-Ordering with different X-fillings . . . 51

4.6 Peak input toggles : Comparison of XStat-Method over existing methods 52

4.7 Peak circuit power : Comparison of XStat-Method over existing methods 53

4.8 Computation time in performing test vector ordering 54

5.1 ITC'99 benchmarks (X % : Average % of X-bits in test cubes) . . . 68

5.2 Peak input toggles : Tool-ordering with different X-fillings 69

5.3 Peak input toggles : BTSP-Ordering followed by different X-fillings 70

5.4 Peak input toggles : X-Base-Ordering with different X-fillings . . . 71

5.5 Peak input toggles : I-Ordering with different X-fillings 72

5.6 Peak input toggles : Comparison of DP-Method over existing methods 73

5.7 Peak circuit power : Comparison of DP-Method over existing methods 74

5.8 Additional savings using local search with 1-bit Neighbourhood . . 75

LIST OF FIGURES

1.1 Static IR-drop profile on a 100x100 power grid 3

2.1 Enhance Scan Flip-Flop proposed in (Dervisoglu and Stong, 1991) . 11

2.2 Combinational State Preservation (CSP) proposed in (Potluri, 2015) 12

2.3 Scan flip-flop that implements the CSP-scan scheme (Potluri, 2015) 13

2.4 Timing diagram for CSP-scan scheme (Potluri, 2015) 14

3.1 An example of edge-weighted undirected complete graph, G 16

3.2 Bottleneck Hamiltonian Cycles (BHCs) in G 17

3.3 Bottleneck Hamiltonian Cycle, Path in $G^{'}$ 18

3.4 BBSS and NN in $G^{'}$ 22

4.1 Motivation for Balanced-X-Filling (B-Fill) 40

4.2 Don't care distribution in test cubes 44

4.3 Computing X-base metric, X-base $(TC_1,\ TC_2)$ = 5 46

4.4 Gap between MAX-X-BASE and MIN-X-BASE for test cube ordering given by commercial tool 47

4.5 Gap between MAX-X-BASE and MIN-X-BASE for X-based test cube ordering 48

5.1 Motivation for Optimum-X-Filling (O-Fill) 60

5.2 Don't care-stretch analysis on `b19` 61

5.3 Bottleneck minimization algorithm iterations 62

5.4 Bottleneck minimization algorithm iterations 63

5.5 Local search technique with 1-bit neighbourhood 65

List of Algorithms

1 NNH Algorithm 20

2 BBSSP Algorithm 21

3 BTSP Algorithm 25

4 $Enhanced\ Lower\ Bound$ Algorithm 26

5 BTSPP Algorithm 27

6 Balanced X-Fill (B-Fill) Algorithm 42

7 X-Based Test Cube Ordering Algorithm 45

8 Bottleneck Minimization Algorithm 48

9 Algorithm for Computing Lower-Bound 57

10 Algorithm for assigning color to intervals 58

11 Optimal X-Filling Algorithm 66

12 Test Vector Ordering Algorithm 67

13 Bottleneck Minimization Algorithm 67

ABBREVIATIONS

IITH	Indian Institute of Technology, Hyderabad
VLSI	Very Large Scale Integration
DFT	Design For Testability
LOC	Launch On Capture
LOS	Launch On Shift
CSP	Combinational State Preservation
ES	Enhanced Scan
TVO	Test Vector Ordering
ATPG	Automatic Test Pattern Generation
PITMP	Peak Input Toggle Minimization Problem
BTSP	Bottleneck Traveling Salesman Problem
BTSPP	Bottleneck Traveling Salesman Path Problem
BHC	Bottleneck Hamiltonian Cycle
BBSS	Bottleneck Biconnected Spanning Subgraph
NNH	Nearest Neighbour Hood
LB	Lower Bound
PIT	Primary Input Toggles
CTT	Circuit Total Toggles
CTP	Circuit Total Power
DP	Dynamic Programming
SA	Simulated Annealing
ITC	International Test Conference
PI	Primary Input
PPI	Pseudo Primary Input
TV	Test Vector
TC	Test Cube

NOTATIONS

C	Circuit
X	Don't care
G	Graph
V	Vertices
E	Edges
P	Polynomial
NP	Non-deterministic Polynomial
μW	Micro Watt
$\%$	Percentage
O	Big-O
$\#$	Number of
π	Order
ϵ	Approximation ratio
$\lfloor , \rfloor$	Floor
$\lceil , \rceil$	Ceil
$\leftarrow$	Assignment
$/*, */$	Comment
$\emptyset$	Empty set
∞	Infinite
δ	Delta
$\forall$	For all
$\in$	in
$\sum$	sum

CHAPTER 1

Introduction

According to Dennard's scaling (Dennard *et al.*, 1974), power density should remain constant, even with increasing device densities. But exponential increase in sub-threshold leakage with threshold voltage scaling caused leakage power to dominate total power consumption (Borkar, 1999). Due to this, threshold voltage scaling and Dennard's scaling came to an end below $100nm$, causing power density to rise exponentially with successive technology generations. Today, aggravated power densities and hot spots have become one of the most important concerns in the nanoscale circuit design. Additionally power dissipation for test vectors is several times higher than that of functional vectors (Gerstendorfer and Wunderlich, 1999). Next, we shall see the issues with these elevated levels of power dissipation during testing.

1.1 Power issues during at-speed testing

The problems concerning test power are two fold. The *first problem* is one of high average test power, which increases thermal stress (Huang, 2007; Yao *et al.*, 2011) on the chip during testing, thereby decreasing its reliability (Saxena *et al.*, 2001; Girard, 2002). In worst cases, the chip can burn on the tester, thereby leading to destructive testing. *The second problem* is that of high peak power during testing. Since power grid is designed for functional vectors, the excessive power dissipation during test vector application can cause excessive IR-drop (Wen *et al.*, 2007; Devanathan *et al.*, 2007*b*), causing timing failures. Since such elevated power levels are not observed during regular operation, such timing failures are categorized as false failures. Since these failures don't occur during the chip's normal functional mode of operation, this problem is also popularly known as the *over testing* problem. This kind of over testing can drastically reduce the fabrication yield, ultimately causing a huge financial loss for the semiconductor manufacturer. Now, at-speed scan based testing is crucial to catch small delay defects that occur during the fabrication of high performance digital chips (Ahmed

et al., 2006*a*; Yilmaz *et al.*, 2008*b*,*a*; Peng *et al.*, 2010; Goel *et al.*, 2010; Yilmaz *et al.*, 2010, 2011; Tehranipoor *et al.*, 2011; Peng *et al.*, 2013; Bao *et al.*, 2013*a*,*b*). These small delay defects can manifest themselves as delay faults, transition faults or stuck-at faults (Chakraborty and Agrawal, 1995*a*,*b*). Since launch to capture clock cycle is very small during at-speed testing, capture is performed when dynamic IR-drop is very high. This causes excessive gate delays on the critical path (Saxena *et al.*, 2003), thus making the over testing problem even more pronounced during at-speed testing. Hence, peak power reduction during at-speed testing is an important problem in the broad area of VLSI testing.

Launch-Off-Capture (LOC), Launch-Off-Shift (LOS) and Enhanced Scan (ES) are the available design-for-testability (DFT) schemes in the literature for the purpose of at-speed testing. Taking physical design overheads and limitations into account, LOC and LOS are the two prevalently used schemes for this purpose. LOS achieves higher fault coverage while consuming lesser test time over LOC scheme, but dissipates higher peak power during the capture phase of the at-speed test (Wu *et al.*, 2011). This excessive peak power in LOS scheme, leads to high *IR-drop* on the power grid, more than what the power grid is designed to handle. This excessive *IR-drop* on the power grid, during capture phase of LOS scheme leads to false delay failures, thereby leading to significant yield reduction that is unwarranted.

This thesis proposes efficient solutions for minimizing peak switching activity during testing, to keep *IR-drop* under control during the same. In static mode, the *IR-drop* increases as the nodes on the power grid get farther from the supply voltage source as shown in Figure 1.1. This figure shows the static supply voltage map for a 100x100 grid with voltage sources at all the nodes on the periphery. The power grid is a rectangular mesh network with each node in the network having current sink of 1μA, simulated using SPICE. Although in practice, all the periphery nodes will not have supply voltage sources, this map is shown to illustrate the idea that the IR-drop increases as the nodes go farther from the supply pins. As already explained during at-speed testing *IR-drop* is strongly correlated to the toggle rates inside the digital circuit during the testing process (Saxena *et al.*, 2003). We thus focus on minimizing peak switching activity as a means to keep *IR-drop* under control during the testing process.

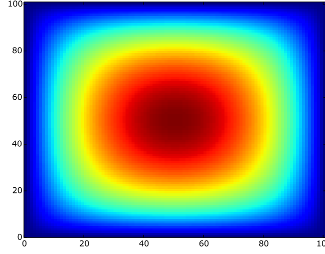


Figure 1.1: Static IR-drop profile on a 100x100 power grid

There has been work in the past for minimizing peak switching activity during testing of combinational circuits (Dabholkar *et al.*, 1998; Girard *et al.*, 1998). However, most of the current high-performance designs are highly pipelined and hence are inherently sequential in nature. These sequential circuits use scan based methodology for the purpose of testing. Due to the disturbance caused in the combinational logic in the scan-shift phase, the peak switching activity reduction techniques proposed in the past for combinational circuits, are not directly applicable to sequential circuits. However, it should be noted that under the CSP-scan architecture (Potluri, 2015; Potluri *et al.*, 2015), many of the algorithms for peak switching activity reduction during testing of combinational circuits, can be applied in a straight-forward manner to sequential circuits. To motivate, we next discuss the techniques proposed in the past for peak switching activity reduction during testing of combinational circuits.

1.2 Reduction of peak power during testing of combinational circuits

Power dissipation in digital Complementary Metal Oxide Semiconductor (CMOS) circuits has two components, namely *static power* and *dynamic power*. Among the two, dynamic power is the major source of power dissipation while the circuit is in operation. Typically, a major portion of a circuit is kept active during testing. This is done to ensure that the total test time spent in testing a chip is reduced i.e., higher fault sampling per test vector. Therefore, dynamic power is the major contributor to power dissipation during testing of a digital chip. The dynamic power dissipation occurs at a node when it switches from $0 \rightarrow 1$ or $1 \rightarrow 0$, and is directly proportional to the toggle count inside the

combinational circuit. Additionally, since interconnect dimensions does not scale the same way as transistor dimensions, interconnect contributes majorly to dynamic power dissipation in today's nanometer CMOS circuits (Magen *et al.*, 2004; Qiu *et al.*, 2008; Potluri *et al.*, 2012). The problems with interconnect scaling further aggravate the timing/power issues during testing, so much, so that there were even techniques proposed on how to perform test vector selection based on interconnect and layout considerations (Yilmaz *et al.*, 2008*a*, 2010). The interconnect scaling asserts itself in another way, on the supply routing interconnects of the power-grid. The inductive effects on the power-grid that were negligible in previous technologies begin to manifest and dominate the *IR-drop* on the power-grid in the sub-100nm technologies (Pant, 2008; Pant *et al.*, 2010). Thus, the increased levels of dynamic power dissipation inside the circuit, produces heavy currents to traverse along the power-grid, creating dynamic inductive drops, which further aggravate the *supply IR-drop* during testing, that was discussed previously.

In (Girard *et al.*, 1998; Dabholkar *et al.*, 1998; Dabholkar and Chakravarty, 1994; Kavousianos *et al.*, 2004; Kurian *et al.*, 2009; Kumar *et al.*, 2010), it was shown that average switching activity during testing of combinational circuits can be reduced by ordering the test vectors as an instance of the Hamiltonian path problem, which is NP-hard. However, this mapping is restricted for minimization of average switching activity, and currently there is no mapping available for minimization of peak switching activity during testing through test vector ordering. For the first time in literature, this thesis proposes a theoretical mapping for peak test switching activity minimization through test vector ordering. The provided mapping is also extended to sequential circuits, which is described in detail in the next section.

1.3 Reduction of peak power during testing of sequential circuits

Today, the scan architecture (Williams and Angell, 1973; Eichelberger, 1974; Eichelberger and Williams, 1977) is used as the de-facto standard for testing sequential circuits. This scheme converts a sequential circuit to a combinational circuit, for the pur-

pose of generating test vectors under the single-stuck-fault (SSF) model. As a result, the rich literature available for test generation (Funatsu *et al.*, 1975; Liaw *et al.*, 1980; Abramovici *et al.*, 1994; Malaiya and Narayanaswamy, 1983; Savir and McAnney, 1988; Schulz *et al.*, 1988; Glover and Mercer, 1988; Reddy *et al.*, 1992; McCluskey and Tseng, 2000; Lin *et al.*, 2001; Liu, 2004; Venkataraman *et al.*, 2004; Ahmed *et al.*, 2006*b*; Miyase and Kajihara, 2006; Bao *et al.*, 2013*a*) and fault simulation (Abramovici *et al.*, 1983; Waicukauski *et al.*, 1986; Takahashi *et al.*, 2006; Chakraborty and Agrawal, 1995*b*; Singh *et al.*, 2006; Bosio *et al.*, 2010) for combinational circuits, can be reused for sequential circuits. In the deep sub-micron CMOS technologies, at-speed testing is necessary to detect small delay defects. Enhanced Scan (ES), Launch on Capture (LOC) and Launch on Shift (LOS) are the currently existing techniques for at-speed testing (Liu, 2004).

In the presence of path delays that are comparable to the clock interval, delayed signal transitions or timing hazards influence the detection of defects. Due to the these variations in signalling delays, it is important to perform at-speed testing even for stuck faults, to reduce the test escapes (McCluskey and Tseng, 2000; Vorisek *et al.*, 2004). It was shown in the past that under the ES architecture, stuck-at vectors can be reused for testing for transition faults (Liu, 2004), with *improvement* in transition fault coverage. But, the implementation of enhanced scan architecture is costly, due to the requirement of multiple clocks (Glover and Mercer, 1988; Dervisoglu and Stong, 1991), which is not feasible in today's designs where routing a single clock, is itself a formidable challenge. In addition to that, test vector ordering is ineffective for reducing peak test power in sequential circuits in standard LOS, LOC and enhanced scan architectures (Potluri, 2015). To address this issue, recently, CSP-scan architecture (Potluri *et al.*, 2015) was proposed, which uses principles of asynchronous circuit design (Sparso and Furber, 2001), to preserve the state of the combinational logic both during scan-shift and capture cycles, thus making test vector ordering effective in reducing peak test power during at-speed testing of stuck faults as well as transition faults in sequential circuits. We assume that this architecture is in place, and propose efficient algorithms for test vector ordering and don't care filling. The following are the contributions of this thesis:

1. We show that given a fully specified test set, optimal test vector ordering problem under the CSP-scan architecture, maps to the Bottleneck Traveling Salesman

Problem (BTSP) problem, which is NP-hard. We solve the optimal test vector ordering problem for all of the ITC circuits by using an efficient BTSP heuristic. Interestingly, the solution obtained in each of the benchmark circuits is globally optimal. The mapping, algorithm, experimentation results and the verification for global optimality of the solutions obtained is given in chapter 3.

2. The test sets are dominated by don't care bits for large circuits, making don't care filling very important for minimizing test power. This increases the hardness of the peak power minimization engine. Keeping this in mind, we propose an efficient heuristic ($XStat$) for test vector ordering and don't care filling in an integrated fashion, that produces solutions which reduce test power significantly, while taking very little time in arriving at the solutions. The details of the proposed heuristic and experimentation results are explained in chapter 4.

3. While $XStat$ is an efficient heuristic for reducing input switching activity, thereby reducing circuit switching activity, it does not guarantee optimality. To address this issue, we show that given a test vector order, don't cares can be filled in an optimal way using *dynamic programming* so as to minimize input switching activity. The details of this algorithm, its proof of optimality and its improvements over $XStat$ are explained in detail in chapter 5.

Under CSP-scan architecture, it is sufficient to validate the proposed algorithms for stuck fault vectors as the transition fault vectors as well as delay fault vectors can be derived from the stuck fault vectors using the technique proposed in (Liu, 2004). Thus, the algorithms proposed in this thesis are generic, in the sense that they are applicable to at-speed testing of faults under all of the aforementioned fault models. The rest of the thesis is organized into 5 chapters. The next chapter gives a background of the low power testing research area and the different techniques proposed in the past to address the low power testing problem. This chapter explains the techniques proposed in the past, at different levels of the VLSI flow, and sets the stage for explaining our contributions. Chapters 3, 4 and 5 explain our contributions. Chapter 6 concludes this thesis.

CHAPTER 2

Background

With technology scaling, the process complexity has increased exponentially. This huge increase in the process complexity, also led to a proportionate increase in manufacturing defect rates. Additionally, the thrust for high-speed devices has made designers focus on high-speed designs. In these high speed designs, the number of gates between two pipelines stages has reduced drastically. As a result, these defects often manifest themselves as small delay defects (Ahmed *et al.*, 2006*b*; Yilmaz *et al.*, 2008*b*,*a*; Goel *et al.*, 2010; Yilmaz *et al.*, 2010, 2011; Bao *et al.*, 2013*b*) in these high-speed designs. In the presence of path delays that are comparable to the clock interval, delayed signal transitions or timing hazards influence the detection of defects. Due to the these variations in signalling delays, it is important to perform at-speed testing even for stuck faults, to reduce the test escapes (McCluskey and Tseng, 2000; Vorisek *et al.*, 2004). However, with increase in test speed, peak power dissipation during at-speed stuck-at testing also increases proportionately. This thesis addresses this problem of peak power minimization during at-speed stuck-at testing.

This chapter is divided into three sections, and this sets the background necessary to understand the proposed algorithms for low power at-speed stuck-at testing. The first section explains the related work in the broad area of low power testing. The second section motivates why at-speed stuck-at testing is important, with an example. The third section explains the design-for-testability (DFT) architecture, in the presence of which, the proposed algorithms are effective in reducing the peak power dissipation during at-speed stuck-at testing. Next, we begin with the first section on prior work related to low power testing.

2.1 Related work in low power testing

There have been several techniques proposed in the past for minimizing peak test power. These techniques can be broadly categorized into circuit level (Gerstendorfer and Wunderlich, 1999; Parimi and Sun, 2004; Bhunia *et al.*, 2005*a*; Devanathan *et al.*, 2007*c*), gate level (Girard *et al.*, 1999; Lee *et al.*, 2000; Almukhaizim and Sinanoglu, 2008; Lin and Rajski, 2008) and system level (Girard *et al.*, 1998; Dabholkar *et al.*, 1998; Sankaralingam *et al.*, 2000; Sankaralingam and Touba, 2002; Devanathan *et al.*, 2007*b*; Yao *et al.*, 2011) techniques.

Circuit level techniques include supply gating (Bhunia *et al.*, 2005*a*), scan flip-flop redesign (Gerstendorfer and Wunderlich, 1999; Parimi and Sun, 2004; Xu and Singh, 2007; Ganesan and Khatri, 2008; Mishra *et al.*, 2010), supply voltage scaling (Devanathan *et al.*, 2007*c*) and circuit partitioning (Girard *et al.*, 1999; Almukhaizim and Sinanoglu, 2008). Gate level techniques include clock gating (Lee *et al.*, 2000), scan cell output gating (Lin and Rajski, 2008), and low power scan chain synthesis (Lee *et al.*, 2000; Bonhomme *et al.*, 2002; Bhattacharya, 2003; Bonhomme *et al.*, 2004). System level techniques include low power test vector generation (Devanathan *et al.*, 2007*b*), test compaction (Sankaralingam *et al.*, 2000; Sankaralingam and Touba, 2002), power aware test scheduling (Yao *et al.*, 2011), test vector ordering (Girard *et al.*, 1998; Dabholkar *et al.*, 1998) and X-filling (Devanathan *et al.*, 2007*b*).

Among these various possibilities, one should choose such a test strategy that minimizes peak power dissipation during testing and at the same time introduces very minimal area, timing and power overheads on the design in its normal functional mode of operation. Thus, system level techniques are most attractive as such techniques do not modify the design at all. This thesis focuses on such system level techniques for minimizing the peak power during at-speed testing of sequential circuits. Low power test vector generation is attractive as it reduces test power without modifying the design. However, due to the hard nature of the test generation and test set compaction problems, adding further constraints would increase the effort of the automatic test vector generation (ATPG) engine, thereby increasing the design cycle of the product. As this is not attractive, we focus on system level techniques that reduce test power significantly, with

little increase in design time. In particular we focus on Test cube ordering and don't care filling. For combinational circuits, capture power is dependent on application of a pair of test vectors- the previous test vector followed by the current test vector. In (Girard *et al.*, 1998) it was shown how test vector ordering for average capture power minimization problem maps to the well known Least Cost Hamiltonian Path Problem which is NP-Hard. In the same paper, a 2-approximation algorithm for TSP was used to achieve reasonably good solutions. In this thesis for both combinational and sequential circuits, it was shown how test vector ordering for peak capture power minimization maps to Bottleneck Hamiltonian Path Problem, which is also NP-Hard. Further details of our contributions will be explained in furture chapters of this thesis.

2.2 Motivation for at-speed stuck-at testing

The real defect is a short or an open between two nodes inside a gate. However, a defect can manifest itself as a stuck-at 0 or stuck-at 1 at the output of a gate. Apart from a defect manifesting itself as stuck-at 0 or stuck-at 1 at the output of a gate, it also changes the delay of the gate. Sometimes, a defect changes the truth table of a gate, which may not be exactly stuck-at 0 or stuck-at 1 behavior. However, they will be usually be caught by the stuck-at tests (McCluskey and Tseng, 2000). In fact, it was shown practically using manufacturing data, that at-speed stuck-at testing can greatly reduce the test escapes (McCluskey and Tseng, 2000; Vorisek *et al.*, 2004). *This motivates the need for at-speed application of stuck-at tests, to reduce the number of test escapes.*. This is especially true in today's chips which are fabricated in deep-submicron technologies, that contain many small delay defects (Ahmed *et al.*, 2006*b*; Yilmaz *et al.*, 2008*b*,*a*; Goel *et al.*, 2010; Yilmaz *et al.*, 2010, 2011; Bao *et al.*, 2013*b*). Now, during at-speed stuck-at testing, if peak power is high, then voltage drops on the power grid is also high, thereby causing excessive delays on gates, leading to the following two scenarios:

1. **good chip:** the response maybe delayed, and since we are capturing at-speed, we observe faulty response, and discard the chip, although it works well in the functional mode of operation (when the excessive delay on gates won't occur); or

2. **defective chip:** the effect of stuck fault maybe masked by an excessive delay of a gate caused by high peak power, which is another type of test escape (Chakraborty and Agrawal, 1995*a*,*b*). This fault can be caught using slow-speed stuck-at testing. However, this additional phase of slow-speed stuck-at testing, as the name indicates, is slow, and hence adds significantly to the test time in modern system-on-chips (SoCs), which are very complex. Now, if we reduce the peak power during the at-speed stuck-at testing, such delay effects on stuck-at testing (Chakraborty and Agrawal, 1995*a*,*b*) can be avoided, thereby reducing the test escapes during at-speed stuck-at testing, and hence an additional phase of slow-speed stuck-at testing can be avoided.

Thus, the advantages of minimizing peak power dissipation during at-speed testing are two-fold:

1. we can avoid a good chip being categorized as defective, which is the problem of *false negatives*, that impacts the yield of a product and a loss to the manufacturer; and more importantly
2. we can avoid a defective chip being categorized as good, which is the problem of *false positives*, that impacts the trust of the customers on the manufacturer, which leads to customer/business loss to manufacturer, finally ending in a financial loss to the manufacturer.

This motivates the need to minimize peak power dissipation during at-speed stuck-at testing. Having motivated this, next we shall see the design for testability techniques existing in the literature, for at-speed testing and the appropriate technique amongst them for the problem under consideration.

2.3 Design for Testability

We focus on ordering the test vectors and selectively filling the don't care (X) bits in the test cubes to minimize peak test power, under the CSP-scan scheme. Before understanding the CSP-scan scheme, it will be useful to understand enhanced scan, the physical design and other limitations posed by this scheme and how the CSP-scan addresses these challenges, yet preserves the properties of enhanced scan. So, next we shall briefly discuss about enhanced scan.

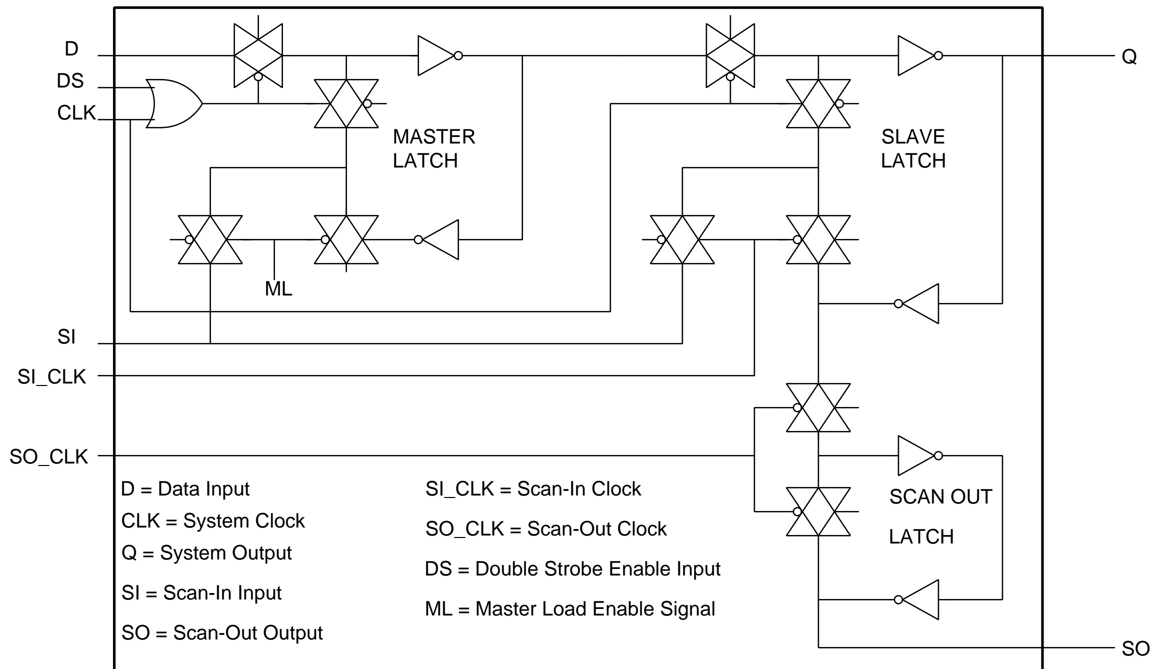


Figure 2.1: Enhance Scan Flip-Flop proposed in (Dervisoglu and Stong, 1991)

2.3.1 Enhanced Scan (ES) scheme

Originally, enhanced scan architecture was proposed in (Dervisoglu and Stong, 1991) for arbitrary two-vector application for at-speed testing of sequential circuits. The circuit schematic of this enhanced scan flip-flop is shown in Figure 2.1. From this schematic, it is clear that to implement this scheme, multiple other clocks (SI_CLK, SO_CLK) are required apart from the system clock (CLK). In today's highly complex chips, routing a single clock itself poses several key challenges like clock-skew, common-path pessimism removal etc. Keeping this in mind, it is beyond question to accept such an implementation, which needs system level routing of more than one clock signal.

Several new implementations of enhanced scan scheme were proposed recently (Datta *et al.*, 2004; Bhunia *et al.*, 2005*b*), to avoid the multiple-clock routing problem and minimize the physical design overhead. However, all of these techniques are meant for arbitrary two-vector application, in which (1) At first, the first vector is scanned in, (2) following which, the first vector is launched into the combinational logic; (3) then, second vector is scanned in, (4) following which, the second vector is launched into the combinational logic; and finally (5) the response is captured.

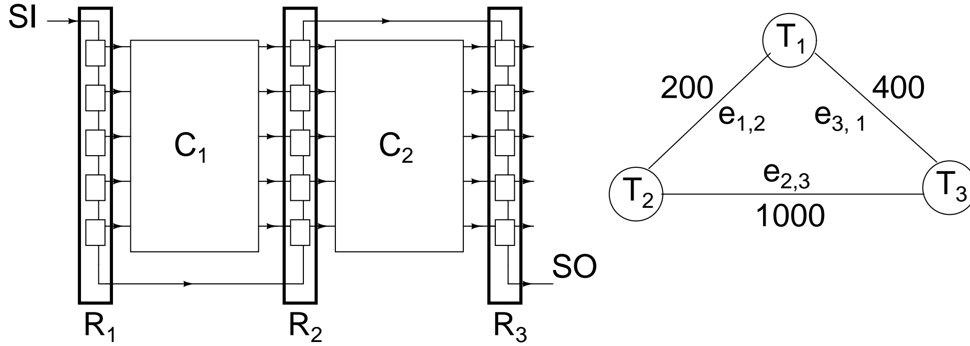


Figure 2.2: Combinational State Preservation (CSP) proposed in (Potluri, 2015)

Hence, it should be clear that the ES scheme is not suitable for test vector ordering, where, after launching each test vector into the combinational logic, the response is also captured. The response thereby captured also disturbs the state of the combinational logic. Thus, the ES scheme preserves the state of combinational logic only during scan-shift and is unable to preserve the state of the combinational logic during the capture cycle. Recently, to address this issue, the combinational state preservation (CSP) scan scheme is proposed, that preserves the state of combinational logic during both scan-shift and capture cycles. The next section explains the CSP-scan scheme in detail.

2.3.2 CSP-scan scheme

The CSP-scan architecture is proposed in (Potluri, 2015) for the purpose of preserving combinational logic states during scan-shift as well as capture phases of LOS based at-speed scan testing. Figure 2.2 shows how the combinational logic states are so preserved that the sequential circuit can practically be treated as combinational circuit, and we can perform test vector ordering for minimizing peak switching activity.

The scan flip-flop that implements the CSP-scan scheme is shown in Figure 2.3. The timing diagram corresponding to the CSP-scan scheme is shown in Figure 2.4. It can be seen that the SE_{latch} is low only during launch, and is high both during scan-shift and capture cycles, thus ensuring combinational state preservation between successive test vectors. It should be noted that, satisfaction of CSP makes test vector ordering effective in reducing peak power during LOS based at-speed testing of sequential circuits (Potluri, 2015).

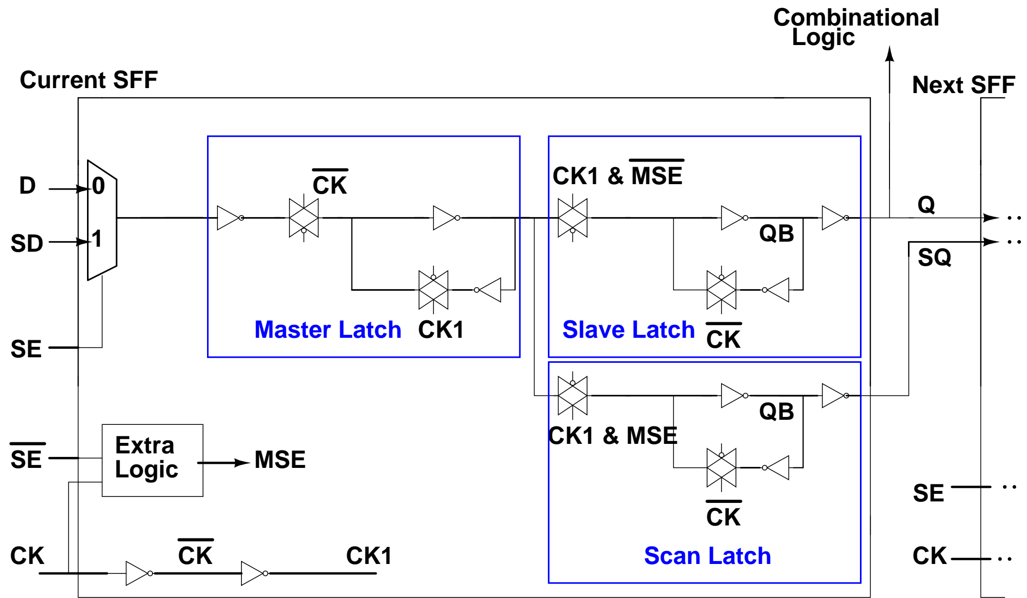


Figure 2.3: Scan flip-flop that implements the CSP-scan scheme (Potluri, 2015)

In this thesis, we focus on peak switching activity minimization during at-speed stuck-at testing. We assume that CSP-scan architecture is already in place and propose algorithms for the same. Additionally, the ATPG tool will give us the option to identify the don't care bits that can be replaced with 0 or 1, without loss in fault coverage (Miyase and Kajihara, 2006). Interestingly, the percentage of don't care bits is 67.8% on an average in the ITC circuits shown in Table 2.1. Since the majority of the bits in these sequential circuits are don't cares, don't care filling plays a major role in minimizing peak power during at-speed testing of sequential circuits. This thesis addresses the test vector ordering problem, the simultaneous vector ordering and don't care filling heuristic, and an optimal algorithm for don't care filling for a given test vector ordering. Chapters 3, 4 and 5 discuss these contributions in elaborate detail. The next chapter describes our first contribution.

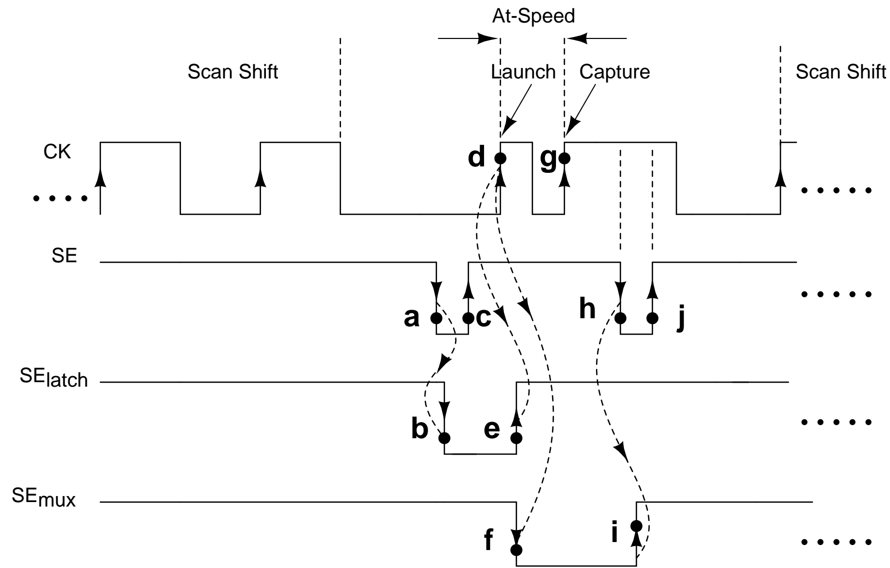


Figure 2.4: Timing diagram for CSP-scan scheme (Potluri, 2015)

Table 2.1: ITC'99 Benchmarks (X % : Average % of X-bits in test cubes)

Benchmark	# PIs	# Gates	# Test Cubes	X %
b04	77	615	67	64.4
b05	35	608	69	36.8
b06	5	60	16	12.5
b07	50	431	46	58.6
b08	30	196	38	60.4
b10	28	217	43	58.7
b11	38	574	83	64.1
b12	126	1.6K	100	76.9
b13	53	596	36	65.4
b14	275	5.4K	511	77.9
b15	485	8.7K	405	87.8
b17	1452	27.99K	618	89.9
b18	3357	75.8K	666	86.9
b19	6666	146.5K	953	89.8
b20	522	9.4K	476	75.3
b21	522	9.4K	479	73.2
b22	767	13.4K	435	74.1

CHAPTER 3

An Efficient Test Vector Ordering Algorithm for Minimizing Peak Switching Activity

As already explained in the previous chapter, under the CSP-scan architecture, the state of the combinational logic is preserved between application of successive test vectors. As a result, test vector ordering influences the peak switching activity during testing. In this chapter, we show that given a fully specified test set, the problem of optimal test vector ordering under the CSP-scan architecture (Potluri *et al.*, 2015), maps to the Bottleneck Traveling Salesman Problem (BTSP), which is NP-hard. We solve the test vector ordering problem by using an efficient BTSP heuristic (Larusic *et al.*, 2012). Interestingly, the solution obtained for all the benchmark circuits, is globally optimal.

Next, we define the Peak Input Toggle Minimization Problem (PITMP) and Bottleneck Traveling Salesman Problem (BTSP) respectively, and how one maps to the other. Section 3.3 explains the BTSP heuristic and section 3.4 provides the results obtained by implementing the proposed heuristic and experimenting it on benchmark circuits.

3.1 PITMP and BTSP

In this section, we shall see the definitions of the Peak Input Toggle Minimization Problem (PITMP) and Bottleneck Traveling Salesman Problem (BTSP) respectively, and how both of them map to each other.

PITMP Definition: Given a combinational circuit C, and a set of test vectors $T = \{T_1 \ldots T_k\}$, the problem is to find an ordering π of these test vectors such that the $max\{Hd\ (T_{\pi_1},\ T_{\pi_2}), Hd\ (T_{\pi_2},\ T_{\pi_3}), ...Hd\ (T_{\pi_{k-1}},\ T_{\pi_k})\}$ is *minimized*, where $Hd\ (T_{\pi_i},\ T_{\pi_{i+1}})$ is the *Hamming distance* between test vectors T_{π_i} and $T_{\pi_{i+1}}$.

BTSP Definition: Given an edge-weighted undirected complete graph G, the problem is to find an *Hamiltonian cycle* in G, such that the *largest edge cost in this cy-*

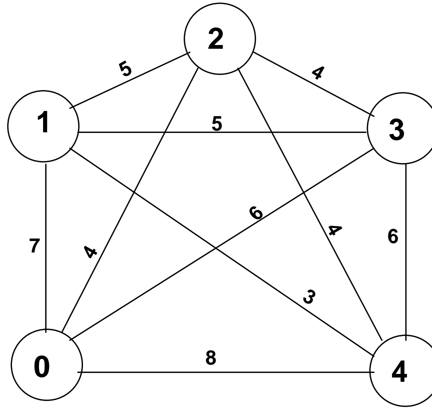


Figure 3.1: An example of edge-weighted undirected complete graph, G

cle is minimized (Garey and Johnson, 1990) (which is the Bottleneck Hamiltonian cycle). Figure 3.1 shows an example edge-weighted undirected complete graph G. Figures 3.2(a) and 3.2(b) show the Bottleneck Hamiltonian cycles (BHCs) in the complete graph (G) shown in Figure 3.1. In this specific example, there are two BHCs inside G. Thus, this example illustrates that the complete graph G, in general, can contain one or more BHCs. It depends on the distribution of weights on the edges of G. Now, we are interested in the peak switching activity, which is the largest edge-weight in the BHC (which will be explained later in the next section). Keeping in this mind, and the fact that the largest edge-weight in all the BHCs are equal, it is straightforward to see that all of the BHCs are equivalent, for the problem under consideration. This will become clearer as we go to the next section. Next, to take the discussion further, we will discuss the *bottleneck traveling salesman path problem*.

The BTSP is NP-Hard (Garey and Johnson, 1990). Next, we shall define the *Bottleneck Traveling Salesman Path Problem* and prove that it is equivalent to BTSP.

Bottleneck Traveling Salesman Path Problem (BTSPP):

Given an edge-weighted undirected complete graph G, the *Bottleneck Traveling Salesman Path Problem (BTSPP)* is to find an *Hamiltonian path* in G, such that the *largest edge cost in this path is minimized*. The BTSPP can be reduced to the BTSP, by adding a vertex to G, and connecting the same to all other vertices of G through edges with weight $zero$. Note that, after solving the BTSP on the modified graph, and removing the newly added vertex from the cycle thus computed, gives a bottleneck traveling salesman path in G.

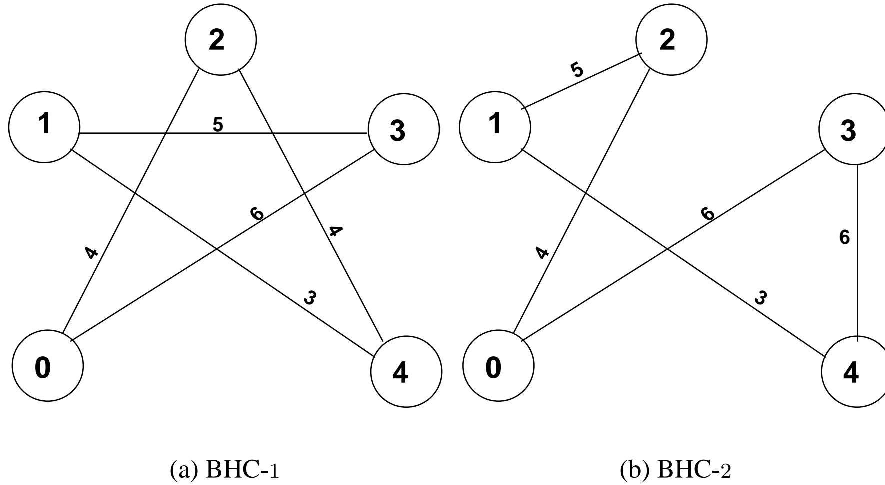


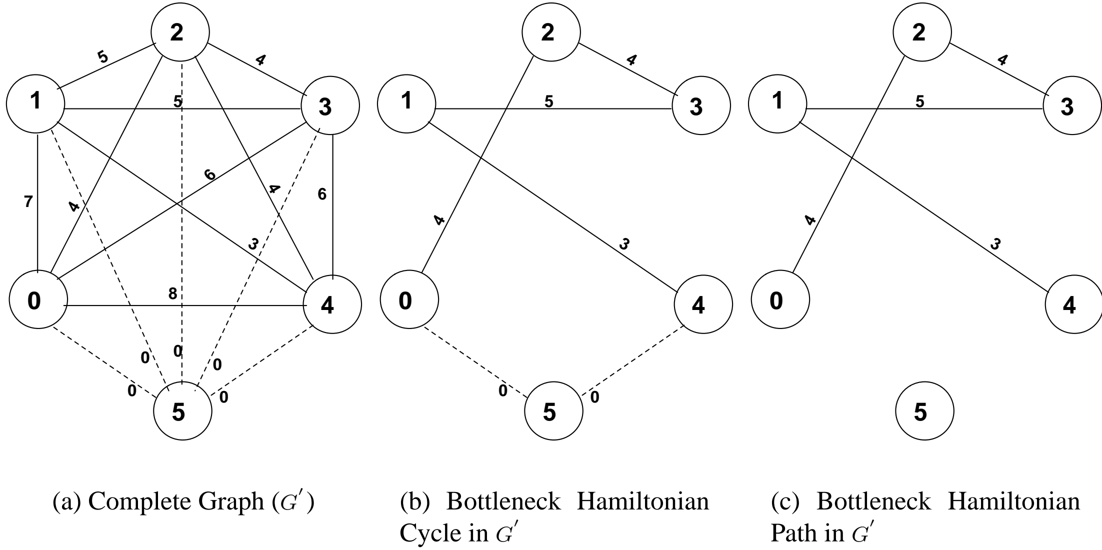
Figure 3.2: Bottleneck Hamiltonian Cycles (BHCs) in G

It is interesting to note that, unless $P = NP$, there does not exist a polynomial time ϵ-approximation algorithm for BTSP for any $\epsilon > 0$ (Doroshko and Sarvanov, 1981; Parker and Rardin, 1984; Sarvanov, 1995). Several heuristics were reported in the literature for the BTSP problem, for example (Ramakrishnan *et al.*, 2009; Manku, 1996; Larusic *et al.*, 2012).

Next, we proceed towards showing that the PITMP can be reduced to BTSP. In this context, we define the $Hamming\ distance(Hd)$ between test vectors T_i, T_j is defined as the number of positions in which (T_i=0 and T_j=1) or (T_i=1 and T_j=0). We denote this by $Hd(T_i,\ T_j)$. The proof of reduction is shown in the next section.

3.2 Mapping of PITMP vs BTSP

In this section we show that PITMP is NP-Hard. We do this by two way reduction between these two problems. Since BTSP is known to be NP-hard due to reduction PITMP is also NP-hard. Next we see the first reduction.



(a) Complete Graph (G')

(b) Bottleneck Hamiltonian Cycle in G'

(c) Bottleneck Hamiltonian Path in G'

Figure 3.3: Bottleneck Hamiltonian Cycle, Path in G'

3.2.1 Mapping of PITMP to BTSP

In this section, we prove that the PITMP can be reduced to BTSP. To begin with, we construct a graph $TVG = (V, E)$ as follows:

- Let $V = \{v_1 \dots v_k\}$ be a set of vertices such that vertex v_i corresponds to test vector T_i, for $1 \leq i \leq k$.
- Place an edge (v_i, v_j) between vertices v_i, v_j with cost $c_{i,j}$, where $c_{ij} = Hd\,(T_i,\ T_j), \forall i, j, 1 \leq i < j \leq k$.
- Add a new vertex v_{k+1} to G and place edge between v_{k+1} and $v_i, \forall 1 \leq i \leq k$, with a cost $c_{k+1,i} = 0, \forall 1 \leq i \leq k$.

In the graph so constructed, let C be an optimal BTSP solution. Let P be a path obtained by removing vertex v_{k+1} from C. Now, the ordering of vertices in P, gives the optimal ordering of the test vectors such that the maximum Hamming distance between any two consecutive test vectors is minimized. Next we see the second reduction.

3.2.2 Mapping of BTSP to PITMP

In this section, we prove that the BTSP can be reduced to PITMP.

Input : An edge-weighted undirected graph $G = (V, E)$

Output : A Hamiltonian path in G, such that the Bottleneck edge-weight is minimized.

- **The Construction Step :**
  1. Let $T = t_1, t_2, t_3, ...t_n$ be a set of test vectors, where t_i corresponds to $v_i \epsilon V$ and $|V| = n$.
  2. Let $PIT(t_i, t_j) = e_{ij}$ where $e_{ij} = w(V_i, V_j) \epsilon E$.

- **The Solution Step :**

  The above construction creates an instance of PITMP. Solve this instance and output ordering of test vectors $(t_1^{'}, t_2^{'}, t_3^{'}, ...t_n^{'})$

- **The Reporting Step :**

  Output the order of vertices in V, corresponding to the test vector sequence $(t_1^{'}, t_2^{'}, t_3^{'}, ...t_n^{'})$.

The Solution step shows that the BTSP is solved as an instance of PITMP. The Construction and Reporting steps takes $O(n^2)$ time. Hence, the BTSP is *polynomially reduced* to an instance of PITMP. Given that the BTSP problem is NP-hard, it is easy to see that PITMP problem is also NP-hard.

Since BTSP is NP-hard, it is important to suggest good heuristics to solve the problem at hand, so that we arrive at fast solutions with reasonable savings in peak input switching activity during testing. The next section explains the BTSP heuristic that we use, to minimize peak input switching activity during testing.

3.3 Algorithm for BTSP

We have used the heuristic proposed by (Larusic *et al.*, 2012) for solving BTSP. This algorithm uses the $Nearest\ Neighbour\ Heuristic(NNH)$ proposed in (Lawler, 1985), for computing upper-bound and $Bottleneck\ Biconnected\ Spanning\ Subgraph(BBSS)$ algorithm proposed in (Punnen and Nair, 1994) for computing lower-bound. These algorithms are explained in Algorithm 1 and Algorithm 2 respectively. Algorithm 1 finds a Hamiltonian Cycle in a complete graph G and returns max cost edge in this cycle. This is an upper-bound for $BTSP$ solution. Algorithm 2 finds an biconnected spanning subgraph of G by

ordering the edges in non decreasing order of edge weights, and does a binary search to find the set of edges in the required biconnected subgraph. To motivate, the bottleneck biconnected spanning subgraph and the nearest neighbourhood of the complete graph in Figure 3.1, are shown in Figures 3.4(a) and 3.4(b) respectively. The BTSP algorithm is shown in Algorithm 3.

Algorithm 1: NNH Algorithm

Input: Graph G
Output: An upper-bound UB for $BTSP$ Solution

```
1  /* Let C be an Hamiltonian cycle in G. Output of this
      algorithm is maximum cost edge in C.                */
2  Let current_vertex be any vertex in graph G and mark current_vertex as
   visited ;
3  Let start_vertex be current_vertex ;
4  Let max_cost be zero. ;
5  while there is any unvisited vertex in graph G do
6  |   Let V be any unvisited vertex in G such that edge cost between
   |   current_vertex and V is minimum;
7  |   Let current_cost be edge cost between current_vertex, V ;
8  |   Let max_cost be the max(max_cost, current_cost) ;
9  |   Let current_vertex be V;
10 end
11 Let current_cost be edge cost between current_vertex, start_vertex ;
12 Let max_cost be the max(max_cost, current_cost) ;
13 return max_cost.
```

Before going into the details of this algorithm, we will next explain an operation called $Controlled\ shake\ operation$, which is extensively used in this algorithm. Let G' be a graph and δ be a positive number. $Controlled\ shake\ operation$ on graph G' with value δ creates a graph G^s as follows

- Vertex set of G^s is the same as vertex set of G'
- Edge set of G^s is the same as edge set of G'
- cost of an edge e in G^s is zero if the cost of the corresponding edge in G' is less than or equal to δ
- cost of an edge e in G^s is any positive random number if the cost of the corresponding edge in G' is greater than δ

Having understood the $Controlled\ shake\ operation$, we will now try to briefly understand the different steps in Algorithm 3. A detailed description of the same can be found

Algorithm 2: BBSSP Algorithm

Input: Graph G
Output: A lower-bound LB for $BTSP$ Solution

```
1  /* Let G' be a biconnected spanning subgraph of G such
      that maximum cost edge in G' is minimum.  Output of
      this algorithm is maximum cost edge in G'.          */
2  Let Z_1 < Z_2 < ... < Z_k be the distinct edge costs of G sorted in increasing
   order;
3  Let l = 1, u = k;
4  while l < u do
5      δ = ⌊(u−l)/2⌋ + l;
6      G' = (V, E') where E' = {(i, j) ∈ E : C_ij ≤ Z_δ }
7      if G' is biconnected then
8          u = δ;
9      end
10     else
11         l = δ + 1;
12     end
13 end
14 return Z_l.
```

in (Larusic *et al.*, 2012). The following points summarize the different steps involved in Algorithm 3.

- Let G^s be a graph obtained from a graph $G^{'}$ by $controlled\ shake\ operation$ with value δ. Note that if G^s contains Hamiltonian tour with cost zero then $G^{'}$ contains a BTSP tour with cost at most δ.

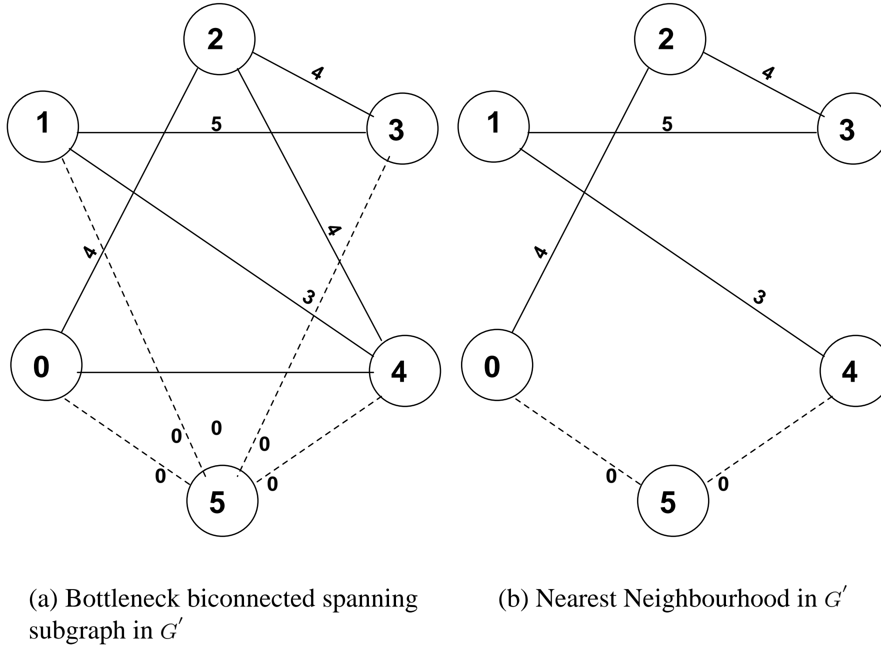
  Suppose BTSP tour cost in a graph $G^{'}$ is $\leq \delta$. Then if we apply $controlled\ shake\ operation$ on $G^{'}$ several times with the same δ then one of the graphs generated by these operations will have Hamiltonian tour with cost zero with high probability.

  In the algorithm 3, $while\ loop$ from line 10 to 18 uses these two ideas while trying to find $BTSP$ tour with cost atmost δ. With high probability it will find such a tour if there exists one.

- The $while\ loop$ from line 5 to 26 tries to find an index i, where $1 \leq i \leq k$, such that given graph $G^{'}$ contains a BTSP tour with cost at most Z_i using binary search.

- In line 21 of the algorithm, whenever we are setting upper bound u equal to mid then we are certain that BTSP tour cost in $G^{'}$ is at most Z_u.

- In line 24 in the algorithm, if we are setting lower bound l to $mid + 1$ does not mean that BTSP tour cost in $G^{'}$ is at least Z_l. It can be less than Z_l with some small probability. This is because we are using a heuristic to test whether the given graph contains a Hamiltonian cycle or not.



(a) Bottleneck biconnected spanning subgraph in G'

(b) Nearest Neighbourhood in G'

Figure 3.4: BBSS and NN in G'

- Whenever this algorithm terminates, lower bound l is equal to upper bound u and given graph contains a BTSP tour with cost at most Z_u.

Now, the lower bound given by $BBSS$ Algorithm 2 is not a tight lower bound. Keeping this in mind, in order to optimize further, we propose Algorithm 4, which tries to tighten this lower bound value. It compares bottleneck value given by $BTSP$ Algorithm 3 and maximum cost edge value given by $BBSS$ Algorithm 2, and tries to tighten the lower bound value. We refer to the lower bound thus obtained by Algorithm 4, as $Enhanced\ lower\ bound$. In the experimental results, we use this $Enhanced\ lower\ bound$ to quantify the performance of $BTSP$ Algorithm.

Algorithm 5 explains the entire process of computing bottleneck value.

3.4 Experimental Results

3.4.1 Experimental Setup

We have considered ITC'99 benchmark circuits listed in table 3.1 for all of our experiments. Each of the ITC'99 benchmark circuit is synthesized using Synopsys®Design

Compiler with a *45nm* standard-cell library. Test vectors were generated for each of the synthesized netlists using *Mentor's FastScan* ATPG tool. The synthesized netlists are taken through Place And Route (PAR) phase using *Cadence Encounter* tool, which is subsequently taken through *Cadence RCXtract* to extract gate and interconnect capacitance values. Next, we explain the results obtained by applying the proposed heuristic on these netlists.

3.4.2 Results

Tables 3.2, 3.3 and 3.4 show the savings in peak input toggles, peak circuit toggles and peak circuit power obtained upon applying the proposed heuristic on benchmark circuits for the cost-functions *primary input toggles*, *total circuit toggles* and *total circuit power* respectively. In all the three tables, *LB* corresponds to the lower bound obtained using the $Enhanced\ Lower\ Bound$ Algorithm 4. Similarly *Tool* and *BTSPP* correspond to the peak toggles/power values in the combinational circuit C obtained, by applying the test vectors in the order suggested by the $FastScan^{TM}$ tool and the BTSPP algorithm respectively.

Table 3.2 shows that when *primary input toggles* is used as the cost-function, the peak input-toggles in C, obtained using the BTSPP algorithm is equal to the LB computed, for all benchmark circuits, while the peak input-toggles in C got by applying the test vectors in the order suggested by $FastScan^{TM}$ is 31.56$\%$ higher than LB, on the average. Similarly Table 3.3 shows that when *total circuit toggles* is used as the cost-function, the peak total-toggles in C, obtained by applying the test vectors in the order suggested by the BTSPP algorithm is equal to the lower bound LB value for all benchmarks considered, except `b19`. In the case of `b19`, step. 9 in $Enhanced\ Lower\ Bound$ algorithm ran for many days and did not converge. This step was thus aborted, and the value of EW_2 was assigned to the LB, since EW_2 is also a lower bound to the BTSPP algorithm. It is interesting to note that although EW_2 $(37,387)$ was not proved to be a tight lower bound, the peak total toggles in C got by applying vectors in ordering suggested by BTSPP (37,726) is within 1% of EW_2 value, indicating the good performance of the BTSPP algorithm, in terms of the solution quality. On the other hand, the peak-toggles in C got by applying the test vectors in the order suggested by

$FastScan^{TM}$ is 59.67% higher than LB, on the average. Similarly, Table 3.4 shows that when *Circuit Total Power* is used as the cost-function, the peak-power dissipated in C by applying the test vectors in the order suggested by the BTSPP algorithm is equal to lower bound LB value for all considered benchmarks, similar to the case of *primary input toggles* (Table. 3.2), while the peak-power dissipated in C got by applying the test vectors in the order suggested by $FastScan^{TM}$ is 62.99% higher than LB, on the average. Since, total energy consumed by the circuit during capture cycles is dependent on average capture-power, it is interesting to analyze the impact of the ordering suggested by the BTSPP algorithm on *average capture-power*. Table 3.5 shows the results for the same for all the three cost-functions discussed previously. It can be observed that for all benchmarks, for all three cost-values, the average toggles/power values in C got by applying the test vectors in the ordering suggested by BTSPP, is lesser than that got by applying the test vectors in the ordering as suggested by tool. On the average, taken over all benchmark circuits, the reduction in average toggles/power for the three cost-values was 27.2%, 27.8% and 28.3% respectively when compared with those yielded by the commercial tool.

```
Algorithm 3: BTSP Algorithm
  Input: Graph G'
  Output: Bottleneck Edge
1  Compute lower-bound lb and upper bound ub using Bottleneck Biconnected
   Spanning Subgraph Problem (BBSSP) algorithm and Nearest Neighbour
   Heuristic (NNH) respectively in given graph G';
2  Let Z_1 < Z_2 < ··· < Z_k be an ascending arrangement of the distinct edge costs
   in graph G' such that Z_1 ≥ lb and Z_k ≤ ub;
3  /* find an index i, where 1 ≤ i ≤ k, such that given
      graph G' contains a BTSP tour with cost at most Z_i
      using binary search                                    */
4  Let l ← 1, u ← k;
5  while l < u do
6  |  mid ← ⌊(l + u)/2⌋ ;
7  |  count ← some positive integer say N ;
8  |  flag ← 1 ;
9  |  δ ← Z_mid ;
10 |  while count > 0 and flag = 1 do
11 |  |  Apply controlled shake on graph G' with value δ to get graph G^s ;
12 |  |  Find a lowest cost TSP tour in G^s using Lin-Kernighan TSP heuristic ;
13 |  |  Let T be this tour ;
14 |  |  if the length of T is zero then
15 |  |  |  flag ← 0;
16 |  |  end
17 |  |  count ← count-1;
18 |  end
19 |  if flag = 0 then
20 |  |  u ← mid;
21 |  end
22 |  else
23 |  |  l ← mid+1;
24 |  end
25 end
  Result: BTSP cost is equal to Z_u.
```

Algorithm 4: $Enhanced\ Lower\ Bound$ Algorithm

Input: TVG of a combinational circuit C constructed for a given set of test vectors T
Output: An Enhanced Lower Bound LB

```
1  Solve the BTSPP on TVG as described earlier. Let the maximum weight of
   any edge on the computed Bottleneck Traveling Salesman Path be EW_1;
2  Compute BBSS(TVG); Let the maximum weight of any edge on
   BBSS(TVG) be EW_2;
3  /* When the cost of the bottleneck edge(EW_1) is same
      as max cost edge in Biconnected spanning subgraph
      (EW_2) then the solution given by BTSP is
      optimal solution, and max cost edge in Biconnected
      spanning subgraph is greatest lower - bound          */
4  if EW_1 == EW_2 then
5  |  LB ← EW_1;
6  end
7  if EW_1 > EW_2 then
8  |  Remove all edges in TVG with edge-weight greater than or equal to EW_1.
   |  Let the new graph be G';
9  |  Test if G' has a Hamiltonian cycle using the methodology suggested in
   |  (Vandegriend, 1998);
10 |  if G' does not have a Hamiltonian cycle then
11 |  |  /* Since G' does not have a Hamiltonian cycle the
   |  |     solution given by BTSP is optimal solution,
   |  |     hence the greatest lower - bound value is EW_1  */
12 |  |  LB ← EW_1;
13 |  end
14 |  else
15 |  |  /* Since G' has a Hamiltonian cycle the solution
   |  |     given by BTSP might not be optimal solution,
   |  |     hence the lower - bound value is at least EW_2  */
16 |  |  LB ← EW_2;
17 |  end
18 end
19 return LB;
```

Algorithm 5: BTSPP Algorithm

Input: π = {T_1, T_2,.. T_n} set of completely specified test vectors

Output: $\pi^{'}$ = {T_1, T_2,.. T_n} sequence of completely specified test vectors

1 Let T_i...T_k be the set of test vectors of Circuit C

2 Let $c_{i,j}$ be the cost of applying test vector j after i, $c_{j,i}$ be the cost of applying test vector i after j.Note that $c_{i,j}$=$c_{j,i}$,$1 \leq i \leq k; 1 \leq j \leq k; i \neq j$.

3 /* Construct a Graph G as follows */

4 Let {v_1...v_k} be the vertex set of G. Note that vertex v_i corresponds to test vector T_i, for 1≤ i ≤ k.

5 Place an edge between v_i,v_j whose cost is c_{ij},$\forall$ i, j. $1 \leq i \leq j \leq k$.

6 /* Construct a Graph $G^{'}$ as follows */

7 Add an vertex v_{k+1} to G.

8 Place an edge between v_{k+1},v_i in G with a cost $zero$, $\forall i, 1 \leq i \leq k$.

9 Compute lower bound LB and upper bound UB using Bottleneck Biconnected Spanning Subgraph ProblemBBSSP Alogrithm and Nearest Neighbor Heuristic(NNH) respectively in given graph $G^{'}$.

10 Let $Z_1 < Z_2 < ... < Z_k$ be an ascending arrangement of the distinct edge costs in graph $G^{'}$ such that $Z_1 \geq LB$ and $Z_k \leq UB$.

11 Let $l = 1, u = k$

12 **while** $l < u$ **do**

13   Let $mid = (l + u)/2)$

14   /* Construct a graph G^s as follows */

15   Vertex set of G^s is the same as vertex set of $G^{'}$

16   Edge set of G^s is the same as edge set of $G^{'}$

17   Cost of an edge e in G^s is $zero$ if the cost of the corresponding edge in $G^{'}$ is less than or equal to Z_{mid}.

18   Cost of an edge e in G^s is $any\ positive\ random\ number$ if the cost of the corresponding edge in $G^{'}$ is greater than Z_{mid}.

19   Find a lowest cost $TSP\ tour$ in G^s using $Lin - Kernighan\ TSP\ heuristic$.

20   **if** $tour\ lenght = 0$ **then**

21   Let $u = mid$;

22   **else**

23   Let $l = mid + 1$;

24   **end**

25 **end**

26 Let P be a path in $G^{'}$ obtained by removing vertex v_{k+1} from $TSP\ tour$. Note that P is a $Hamiltonian\ path$ in G such that cost of any edge is atmost Z_u.

27 Ordering of the vertices in path P gives the required test vector ordering such that $peak\ cost$ is atmost Z_u.

Table 3.1: ITC'99 Benchmarks

Circuit	# PIs	#Gates	# Test Vectors
b01	5	57	17
b02	4	31	11
b03	29	103	16
b04	77	615	98
b05	35	608	81
b06	5	60	19
b07	50	431	61
b08	30	196	49
b09	29	162	33
b10	28	217	54
b11	38	574	104
b12	126	1.6K	118
b13	53	396	44
b14	275	5.4K	658
b15	485	8.7K	594
b17	1452	28K	786
b18	3357	75.8K	913
b19	6666	146.52K	1,147
b20	522	9.4K	652
b21	522	9.4K	671
b22	767	13.4K	589

Table 3.2: Edge cost : Primary input toggles per vector pair

Circuit	LB	Tool	BTSPP	% gap with LB		Run Time
				Tool	BTSPP	
b01	2	5	2	150.00	0.00	0.04s
b02	2	4	2	100.00	0.00	0.05s
b03	7	11	7	57.14	0.00	0.34s
b04	31	49	31	58.06	0.00	2.01s
b05	12	24	12	100.00	0.00	0.33s
b06	2	4	2	100.00	0.00	0.06s
b07	19	34	19	78.95	0.00	1.33s
b08	11	21	11	90.91	0.00	0.22s
b09	10	21	10	110.00	0.00	0.05s
b10	11	19	11	72.73	0.00	1.52s
b11	14	27	14	92.86	0.00	0.88s
b12	53	79	53	49.06	0.00	1.00s
b13	21	34	21	61.90	0.00	0.11s
b14	114	158	114	38.60	0.00	4.47m
b15	216	280	216	29.63	0.00	2.97m
b17	679	785	679	15.61	0.00	22.93s
b18	1,601	1,760	1,601	9.93	0.00	3.18m
b19	3,218	3,447	3,218	7.12	0.00	15.20m
b20	231	294	231	27.27	0.00	16.70s
b21	228	294	228	28.95	0.00	4.06m
b22	349	512	349	46.70	0.00	13.76s
Average	-	-	-	31.56	0.00	1.47m

Table 3.3: Edge cost : Circuit total toggles per vector pair

Circuit	LB	Tool	BTSPP	% gap with LB		Run Time
				Tool	BTSPP	
b01	21	37	21	76.19	0.00	0.02s
b02	13	18	13	38.46	0.00	0.07s
b03	29	52	29	79.31	0.00	0.07s
b04	226	375	226	65.93	0.00	0.68s
b05	189	323	189	70.90	0.00	2.81s
b06	18	39	18	116.67	0.00	0.02s
b07	143	252	143	76.22	0.00	0.15s
b08	64	119	64	85.94	0.00	1.28s
b09	51	107	51	109.80	0.00	0.70s
b10	66	132	66	100.00	0.00	1.36s
b11	154	264	154	71.43	0.00	5.32s
b12	443	682	443	53.95	0.00	3.49s
b13	149	212	149	42.28	0.00	0.17s
b14	1,565	2,449	1,565	56.49	0.00	16.77s
b15	2,078	3,148	2,078	51.49	0.00	8.90m
b17	7,125	9,217	7,125	29.36	0.00	15.68m
b18	20,103	24,694	20,103	22.84	0.00	27.81s
b19	37,387	44,934	37,726	20.19	0.91	112.77m
b20	3,003	4,007	3,003	33.43	0.00	7.75m
b21	2,962	3,911	2,962	32.04	0.00	9.27m
b22	4,341	5,188	4,341	19.51	0.00	3.31m
Average	-	-	-	59.67	0.05	7.56m

Table 3.4: Edge cost : Circuit total power (in μW) per vector pair

Circuit	LB	Tool	BTSPP	% gap with LB		Running Time
				Tool	BTSPP	
b01	1.92	3.97	1.92	107.04	0.00	0.02s
b02	1.47	2.30	1.47	56.08	0.00	0.07s
b03	2.03	3.54	2.03	74.58	0.00	0.07s
b04	14.319	24.94	14.32	74.17	0.00	11.40s
b05	10.34	17.31	10.34	67.46	0.00	0.51s
b06	2.04	4.40	2.04	115.76	0.00	0.28s
b07	10.09	18.42	10.09	82.54	0.00	4.31s
b08	4.41	8.64	4.41	95.81	0.00	1.75s
b09	4.71	11.18	4.71	137.53	0.00	0.80s
b10	5.64	10.91	5.64	93.37	0.00	1.03s
b11	10.28	17.28	10.28	68.11	0.00	5.42s
b12	32.08	53.430	32.08	66.54	0.00	1.07s
b13	11.97	17.65	11.97	47.45	0.00	0.11s
b14	71.19	115.23	71.19	61.86	0.00	18.89s
b15	140.36	199.50	140.36	42.14	0.00	14.03m
b17	808.36	967.69	808.36	19.71	0.00	18.98m
b18	2,451.40	2,729.24	2,451.40	11.33	0.00	29.51m
b19	7,205.46	7,815.73	7,205.46	8.47	0.00	44.09m
b20	198.49	275.54	198.49	38.82	0.00	19.50s
b21	188.82	245.53	188.82	30.04	0.00	20.05s
b22	321.32	397.66	321.32	23.76	0.00	9.29m
Average	-	-	-	62.99	0.00	5.59m

Table 3.5: Impact of Test Vector Ordering on Average Toggles/Power for cost values (PIT: Primary Input Toggles, CTT: Circuit Total Toggles, CTP: Circuit Total Power)

Circuit	Edge Cost : PIT per Vector Pair			Edge Cost : CTT per Vector Pair			Edge Cost : CTP per Vector Pair		
	Average Input Toggles			Average Circuit Toggles			Average Circuit Power (in μW)		
	Tool	BTSPP	% Improvement	Tool	BTSPP	% Improvement	Tool	BTSPP	% Improvement
b01	2	1	50.00	29	18	37.93	2.76	1.64	40.83
b02	2	1	50.00	15	11	26.67	1.80	1.28	28.75
b03	8	3	62.50	34	16	52.94	2.41	1.06	56.01
b04	37	29	21.62	283	209	26.15	18.66	13.54	27.45
b05	17	11	35.29	239	167	30.13	13.04	9.32	28.58
b06	3	1	66.67	28	14	50.00	3.19	1.49	53.35
b07	24	16	33.33	187	128	31.55	13.56	8.92	34.19
b08	14	10	28.57	85	55	35.29	6.09	3.71	39.03
b09	14	9	35.71	82	42	48.78	8.19	4.01	51.00
b10	14	9	35.71	93	56	39.78	8.19	4.76	41.86
b11	18	13	27.78	198	132	33.33	13.79	8.95	35.08
b12	62	51	17.74	555	412	25.77	40.00	29.83	25.42
b13	27	20	25.93	182	140	23.08	14.87	11.29	24.11
b14	134	109	18.66	1957	1457	25.55	91.66	66.33	27.63
b15	241	212	12.03	2547	1950	23.44	167.92	133.33	20.33
b17	722	672	6.93	8025	6976	13.07	877.52	797.09	9.17
b18	1671	1591	4.79	21846	19875	9.02	2574.81	2433.27	5.50
b19	3317	3206	3.35	40703	37434	8.03	7458.16	7174.18	3.81
b20	257	226	12.06	3451	2909	15.71	229.95	192.01	16.50
b21	257	223	13.23	3446	2856	17.12	215.51	183.17	15.01
b22	387	344	11.11	4712	4265	9.49	353.27	316.21	10.49
Average	-	-	27.2	-	-	27.8	-	-	28.3

Since three different cost functions are considered, it is interesting to see the differences in using them. Table 3.6 provides the peak capture power values for the test vector orderings obtained by using cost functions (PIT: Primary Input Toggles, CTT: Circuit Total Toggles, CTP: Circuit Total Power). Since CTP considers the actual power, it gives the best power saving for almost all benchmark circuits. Additionally Table 3.6 shows that among the three cost-functions, CTP is the best, followed by CTT and then PIT, in their effectiveness in saving power. As far computational requirements are concerned, PIT is the best since only input toggles need to be computed. Since total circuit activity needs to be computed to compute CTT, it is slower than PIT computation. In addition to total circuit activity, since placement and routing also need to be done to compute CTP, it is slowest of all the three. Thus, one has to strike a trade-off between power saving and computational efficiency in choosing the appropriate cost function among PIT, CTT and CTP cost functions, for the test vector ordering process.

Table 3.7 provides the average capture power values for the test vector orderings obtained by using these different cost functions (PIT: Primary Input Toggles, CTT: Circuit Total Toggles, CTP: Circuit Total Power). This table shows that CTP is most effective, followed by CTT and then PIT, in saving average power. Even though this BTSP algorithm is designed to minimize peak power, it did reasonably well for reducing average power also.

3.5 Summary

In this chapter we mapped the peak-power minimization problem on to an instance of the Bottleneck Traveling Salesman Problem (BTSP), which is known to be NP-hard. An efficient BTSP heuristic is deployed to find the minimum peak capture-power. Three different cost functions were used for evaluating the proposed heuristic. For each cost function, the solution given by this BTSP heuristic is optimal for almost all ITC'99 benchmark circuits, in optimizing the corresponding BTSP cost-function. As far as minimizing peak capture power is concerned, it is found that *total circuit power* is most effective, however is computationally most expensive; *primary input toggles* is the fastest, however the solution is most inferior among three. Hence, there is a trade-

Table 3.6: Peak Circuit Power Comparisons (in μW) for different cost functions considered (PIT: Primary Input Toggles, CTT: Circuit Total Toggles, CTP: Circuit Total Power)

Circuit	Peak Circuit Power				% Improvement of BTSP over Tool		
	Tool	BTSP			for edge cost value		
		PIT	CTT	CTP	PIT	CTT	CTP
b01	3.97	2.34	2.12	1.92	41.06	46.6	51.64
b02	2.3	1.8	1.79	1.47	21.74	22.17	36.09
b03	3.54	2.42	2.11	2.03	31.64	40.4	42.66
b04	24.94	19.56	16.73	14.32	21.57	32.92	42.58
b05	17.31	14.74	11.52	10.34	14.85	33.45	40.27
b06	4.4	3.32	2.18	2.04	24.55	50.45	53.64
b07	18.42	15.76	12.12	10.09	14.44	34.2	45.22
b08	8.64	7.49	5.09	4.41	13.31	41.09	48.96
b09	11.18	9.31	4.77	4.71	16.73	57.33	57.87
b10	10.91	9.27	6.43	5.64	15.03	41.06	48.3
b11	17.28	15.05	11.78	10.28	12.91	31.83	40.51
b12	53.43	41.77	35.59	32.08	21.82	33.39	39.96
b13	17.65	14.87	12.83	11.97	15.75	27.31	32.18
b14	115.23	105.44	84.43	71.19	8.5	26.73	38.22
b15	199.5	187.88	148.96	140.36	5.82	25.33	29.64
b17	967.69	901.29	854.22	808.36	6.86	11.73	16.46
b18	2729.24	2649.9	2578.74	2451.4	2.91	5.51	10.18
b19	7815.73	7540.99	7559.69	7205.46	3.52	3.28	7.81
b20	275.54	250.85	218.58	198.49	8.96	20.67	27.96
b21	245.53	230.6	212.74	188.82	6.08	13.35	23.1
b22	397.66	371.55	341.77	321.32	6.57	14.05	19.2
Average	-	-	-	-	14.98	29.18	35.83

Table 3.7: Average Circuit Power Comparisons (in μW) for different cost functions considered (PIT: Primary Input Toggles, CTT: Circuit Total Toggles, CTP: Circuit Total Power)

Circuit	Average Circuit Power				% Improvement of BTSP over Tool for edge cost value		
	Tool	BTSP					
		PIT	CTT	CTP	PIT	CTT	CTP
b01	2.76	2.11	1.72	1.64	23.55	37.68	40.58
b02	1.8	1.28	1.26	1.28	28.89	30	28.89
b03	2.41	0.99	1.12	1.06	58.92	53.53	56.02
b04	18.66	14.99	14.05	13.54	19.67	24.71	27.44
b05	13.044	10.62	9.42	9.32	18.58	27.78	28.55
b06	3.19	2.44	1.6	1.49	23.51	49.84	53.29
b07	13.56	10.33	8.84	8.92	23.82	34.81	34.22
b08	6.09	4.88	3.89	3.71	19.87	36.12	39.08
b09	8.19	5.37	4.04	4.01	34.43	50.67	51.04
b10	8.19	5.82	4.92	4.76	28.94	39.93	41.88
b11	13.79	10.95	9.15	8.95	20.59	33.65	35.1
b12	40	34.19	29.76	29.83	14.53	25.6	25.43
b13	14.87	12.41	11.47	11.29	16.54	22.86	24.08
b14	91.66	80.16	69.61	66.33	12.55	24.06	27.63
b15	167.92	153.41	133.85	133.77	8.64	20.29	20.34
b17	877.52	828.02	807.11	797.09	5.64	8.02	9.17
b18	2574.81	2483.06	2466.77	2433.27	3.56	4.2	5.5
b19	7458.16	7261.64	7280.73	7174.18	2.63	2.38	3.81
b20	229.95	211.46	194.35	192.01	8.04	15.48	16.5
b21	215.51	196.29	184.65	183.17	8.92	14.32	15.01
b22	353.27	333.22	320.14	316.21	5.68	9.38	10.49
Average	-	-	-	-	18.45	26.92	28.29

off between solution quality and computational efficiency.

Usually, the test vectors generated by the ATPG tool are typically dominated by don't care (X) bits, especially for large circuits. Thus, X-filling is a very effective technique for peak power minimization during testing. It should be noted that after the X-bits are already filled, the algorithms proposed in this chapter, offer very elegant solutions. However, it is possible that X-bit filling and ordering of the test vectors can be done in an integrated fashion, to obtain much better peak power savings. The next chapter explores this possibility and reports the results thereby obtained.

CHAPTER 4

An Efficient X-filling algorithm for Minimizing Peak Switching Activity

The ATPG tool will give us the option to identify the don't care bits that can be replaced with 0 or 1, without loss in fault coverage (Miyase and Kajihara, 2006). Interestingly, the percentage of don't care bits is 67.8% on an average in the ITC circuits shown in Table 4.1.

Table 4.1: ITC'99 benchmarks (X % : Average % of X-bits in test cubes)

Benchmark	# PIs	# Gates	# Test Cubes	X %
b01	5	57	14	7.14
b02	4	31	10	5.00
b03	29	103	19	70.42
b04	77	615	67	64.35
b05	35	608	69	36.77
b06	5	60	16	12.50
b07	50	431	46	58.57
b08	30	196	38	60.44
b09	29	162	23	38.23
b10	28	217	43	58.72
b11	38	574	83	64.11
b12	126	1.6K	100	76.94
b13	53	596	36	65.41
b14	275	5.4K	511	77.90
b15	485	8.7K	405	87.75
b17	1452	27.99K	618	89.85
b18	3357	75.8K	666	86.92
b19	6666	146.5K	953	89.81
b20	522	9.4K	476	75.29
b21	522	9.4K	479	73.20
b22	767	13.4K	435	74.05

A test vector, when some bits are left as don't cares, is known as a *test cube*. In this chapter we concentrate on filling these don't cares in test cubes and ordering them to minimize the peak power. We propose an efficient heuristic for test cube ordering and

don't care filling in an integrated fashion, that produces solution which reduce peak test power significantly. The organization of this chapter is as follows: section 4.1 defines the Peak Input Toggle (PIT) Minimization Problem in the presence of X-bits, following which we explain our balanced X-Filling Algorithm in section 4.2. Next, we explain the Test Vector Ordering (TVO) Algorithm in the presence of X-bits in section 4.3.1. Finally, we explain the integrated ordering+X-filling algorithm in section 4.4 and provide the experimental results in section 4.5.

4.1 Peak Input Toggle Minimization Problem (PITMP)

Problem Definition: Given a combinational circuit C and a set of test cubes $TC = \{TC_1 \ldots TC_k\}$, the problem is to compute an ordering π of these test cubes and filling the don't cares to generate test vector sequence $T_{\pi_1}, \ldots T_{\pi_k}$ such that the $max\{Hd(T_{\pi_1},\ T_{\pi_2}),\ Hd(T_{\pi_2},\ T_{\pi_3}) \ldots Hd(T_{\pi_{k-1}},\ T_{\pi_k})\}$ is *minimized*, where $Hd(T_{\pi_i},\ T_{\pi_{i+1}})$ is the Hamming distance between test vectors T_{π_i} and $T_{\pi_{i+1}}$.

4.2 Balanced X-Filling (B-Fill) Algorithm

Problem Definition: Given a sequence of test cubes $TC_1 \ldots TC_k$ each of length m, replace each don't care in test cubes by either 0 or 1 such that $max\{Hd(TC_1,\ TC_2),\ Hd(TC_2,\ TC_3) \ldots Hd(TC_{k-1}, TC_k)\}$ is *minimized*, where $Hd(TC_i, TC_{i+1})$ is the Hamming distance between test cubes TC_i, TC_{i+1} after replacing don't cares by either 0 or 1.

4.2.1 Motivation

By definition, the X-bits (don't care bits) in the test cubes generated by ATPG tool can be filled with 0 or 1, without affecting fault coverage. Table 4.1 shows the average percentage of X-bits (over all the test cubes) for each benchmark circuit. It can be seen that as circuit size increases, the average number of X-bits also increases, motivating the need for an effective and efficient X-filling algorithm to reduce the peak capture power

consumption during scan test. Our aim is to perform X-filling to convert test cubes to test vectors in such a way that peak input toggles is minimized. The existing well known X-filling techniques $random$-fill (R-fill), $zero$-fill (0-fill), one-fill (1-fill), and $Minimum\,Transition$-fill (MT-fill) (Sankaralingam *et al.*, 2000), which are explained in the following subsection, are not optimized for minimizing peak input toggles. This motivated us to design an efficient algorithm which is customized for peak input toggle minimization without compromising in average number of toggles. Additionally there are serveral other X-filling techniques proposed in the literature such as (Wu *et al.*, 2011; Miyase *et al.*, 2011; Li *et al.*, 2010; Balatsouka *et al.*, 2010; Wu *et al.*, 2009; Kundu and Chattopadhyay, 2009; Tzeng and Huang, 2009; Li *et al.*, 2008; Wen *et al.*, 2007; Remersaro *et al.*, 2006) which are not feasible with CSP-Scan, hence not very effective and therefore not compared against techniques proposed in this thesis.

4.2.2 Existing X-Filling Techniques

A list of existing techniques to X-filling for power reduction are as follows:

1. **R-fill** : This technique replaces all don't cares by zero or one randomly. As a result of randomness in filling, with high probability this technique will also not give the optimum value. As shown in Fig. 4.1, after applying this type of filling of X-bits in the test cubes, peak input toggle count is 4.
2. **0-fill** : This technique replaces all don't cares by zero, as shown in Fig. 4.1. As a result of applying this type of filling of X-bits in the test cubes, peak input toggle count is 4.
3. **1-fill** : This technique replaces all don't cares by one, as shown in Fig. 4.1. As a result of applying this type of filling of X-bits in the test cubes, peak input toggle count is 4.
4. **MT-fill** : This technique attempts to reduce adjacent toggles between vectors. As a result, it minimizes the total number of input toggles, as shown in Fig. 4.1. Note that, as a result of applying this type of filling of X-bits in the test cubes, peak input toggle count is 5, which is more than optimum value which is 2.

(a) Input			(b) 0-Fill			(c) 1-Fill			(d) R-Fill			(e) MT-Fill			(f) B-Fill		
TC_1	TC_2	TC_3	TC_1	TC_2	TC_3	TC_1	TC_2	TC_3	TC_1	TC_2	TC_3	TC_1	TC_2	TC_3	TC_1	TC_2	TC_3
1	X	0	1	0	0	1	1	0	1	0	0	1	1	0	1	1	0
1	X	0	1	0	0	1	1	0	1	1	0	1	1	0	1	1	0
1	X	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0
X	X	X	0	0	0	1	1	1	0	1	1	0	0	0	0	0	0
1	X	0	1	0	0	1	1	0	1	0	0	1	1	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

(a) Input (b) 0-Fill (c) 1-Fill (d) R-Fill (e) MT-Fill (f) B-Fill

Figure 4.1: Motivation for Balanced-X-Filling (B-Fill)

4.2.3 Algorithm Details

The proposed algorithm for X-filling is shown in Algorithm 6. Before getting into the details of this algorithm, let us explain a few things. We define the $modified\ Hamming\ distance\ (mHd)$ function between test cubes TC_i, TC_j as follows:

$mHd\ (TC_i, TC_j)$ = Number of positions in which $(TC_i,\ TC_j)$ = 01 or 10.

Note that while computing this function, we ignore the positions where don't cares appear either in TC_i or TC_j. These don't cares are filled at a later step based on the X-filling strategy shown in Table 4.2.

Table 4.2: Lookup table for X-filling

Input			Output	
$V(i-1)$	$V(i)$	$V(i+1)$	$V(i)$	% of cases
0	X	0	0	100
1	X	1	1	100
0	X	X	0	100
1	X	X	1	100
X	X	0	0	100
X	X	1	1	100
X	X	X	X	100
0	X	1	$V(i-1)$	50
0	X	1	$V(i+1)$	50
1	X	0	$V(i-1)$	50
1	X	0	$V(i+1)$	50

Whenever TC_i and TC_j are free from don't cares, then this function is equivalent to the $Hamming\ distance$ function. Having understood this, a brief explanation of Algorithm 6 is given as follows:

1. For loop between lines 1 to 13 in algorithm scans the test vectors from two to n-1 in sequential manner and perform the following actions
   - Let TC_i be the current test cube under consideration for this for loop. Let $TC_{j,i}$ denotes j^{th} bit in i^{th} test cube.
   - If $(TC_{j,i-1}, TC_{j,i}, TC_{j,i+1}) = (0, X, 0)\ or\ (0, X, X)$ then replace $TC_{j,i}$ by zero.
   - If $(TC_{j,i-1}, TC_{j,i}, TC_{j,i+1}) = (1, X, 1)\ or\ (1, X, X)$ then replace $TC_{j,i}$ by one.

   At the end of execution of this for loop, if we look at any row it contains one of XX..X0, XX..X1 as a prefix or one of X0, X1 as a suffix or one of 0X1, 1X0 as a substring.
2. At the end of the lines 14 to 18, any row contains one of 0X1, 1X0 as a substring.
3. For loop between lines 28 to 42 scans the test vectors from two to n-1 in sequential manner and fill the don't cares in TC_i such that difference between $mHd\,(TC_{i-1}, TC_i)$ and $mHd\,(TC_i, TC_{i+1})$ is minimized, where TC_i is the current test cube under consideration for this for loop.

Running time of this algorithm is $O(mn)$ where m is number of bits in test cube and n is number of test cubes.

In this section, we have explained the balanced X-filling algorithm, that aims at minimizing peak input toggles by filling the X-bits in an efficient manner. Next, we show the importance of initial test vector order, that maximizes the savings produced by this balanced X-filling algorithm in minimizing peak test power.

4.3 Test Cube Ordering Algorithm

As already explained, test vector ordering and balanced X-filling are both efficient in reducing peak test power. Then, it is important to use an intelligent mix of the two, to obtain the best possible peak test power savings. In this context, we propose an algorithm that achieves this objective. Before getting into the details of the proposed *Integrated Test Cube Ordering and X-filling Algorithm*, we will motivate the need for

Algorithm 6: Balanced X-Fill (B-Fill) Algorithm

Input: $TC = TC_1, TC_2,.. TC_n$ be the sequence of input test cubes
Output: $T = T_1, T_2,.. T_n$ sequence of completely specified test vectors

```
1  /* processing of 0XX..XX0 and 1XX..XX1 stretches */
2  for i = 2 → n − 1 do
3  |  for j = 1 → Number of bits in Test Vector do
4  |  |  /* TC_{j,i} denotes j^th bit in i^th vector */
5  |  |  switch (TC_{j,i−1}, TC_{j,i}, TC_{j,i+1}) do
6  |  |  |  case (0, X, 0)or(0, X, X)  TC_{j,i} ← 0;
7  |  |  |  case (1, X, 1)or(1, X, X)  TC_{j,i} ← 1;
8  |  |  |  otherwise
9  |  |  |  |  TC_{j,i} ← TC_{j,i};
10 |  |  |  end
11 |  |  end
12 |  end
13 end
14 /* processing of XX..X0, XX..X1, 0X and 1X stretches
      */
15 If { j^th row contain a prefix XXX...XX0} then replace every don't care in this
   prefix by zero
16 If { j^th row contain a prefix XXX...XX1} then replace every don't care in this
   prefix by one
17 If { j^th row contain a suffix 0X} then replace don't care in this suffix by zero
18 If { j^th row contain a suffix 1X} then replace don't care in this suffix by one
19 /* processing of 1X0 and 0X1 stretches */
20 for i = 2 → n do
21 |  Count_i ← 0;
22 |  for j = 1 → Number of bits in Test Vector do
23 |  |  if (TC_{j,i−1}, TC_{j,i}) = (0, 1)or(1, 0) then
24 |  |  |  Count_i ← Count_i+1;
25 |  |  end
26 |  end
27 end
28 for i = 2 → n − 1 do
29 |  for j = 1 → Number of bits in Test Vector do
30 |  |  if (TC_{j,i−1}, TC_{j,i}, TC_{j,i+1}) = (0, X, 1) or (1, X, 0) then
31 |  |  |  if Count_i ≤ Count_{i+1} then
32 |  |  |  |  TC_{j,i} ← TC_{j,i+1};
33 |  |  |  |  Count_i ← Count_i+1;
34 |  |  |  end
35 |  |  |  else
36 |  |  |  |  TC_{j,i} ← TC_{j,i−1};
37 |  |  |  |  Count_{i+1} ← Count_{i+1}+1;
38 |  |  |  end
39 |  |  end
40 |  end
41 end
42 Let T_i=TC_i for 1 ≤ i ≤ n
   Result: return T
```

an efficient test cube order for balanced X-filling to be very effective in reducing peak test power.

4.3.1 The Need for an Efficient Test Cube Order

According to our observation, for a given test cube sequence, the highest number of toggles occur between a pair of adjacent test cubes, in which both test cubes have large number of specified bits. If we take a test cube sequence generated by ATPG, we can find such an adjacent test cube pair quite often. This is because ATPG follows two phase approach of test generation. The distribution of don't cares in test cubes generated by commercial ATPG tool is shown in Figure 4.2, for four different benchmarks. For a given benchmark, the scale on X-axis signifies the percentage of don't care bits in a given test cube, and the scale on Y-axis signifies the number of test cubes whose don't care percentage lies in a given interval on X-axis. The X-axis is divided into ten intervals of size 10% each. The distribution for a given benchmark shown in Figure 4.2 shows the collection of data points corresponding to each interval on X-axis, for that particular benchmark, joined through straight lines. The reason for this trend is that the test cubes in the initial part of the ordering are typically random vectors, also known as fault independent tests and test cubes in the latter part of the ordering are fault oriented vectors targeted to detect hard-to-detect faults (Abramovici *et al.*, 1994). Thus, the initial vectors have few X-bits, and the number of X-bits in each test cube tapers down as we go further into the ordering given by the ATPG tool. As a result of this behavior, the difference in X-bit count between adjacent vectors in the original test cube order, does not give much room for minimizing toggle count through X-filling. Keeping this in mind, we need to order the test cubes such that the test cubes with large number of don't cares are interspersed with those test cubes with less number of don't cares. This gives a lot of freedom while converting don't cares into zero or one for minimization of peak input toggles. Having motivated the relationship between test cube order and effective of the balanced X-filling algorithm in reducing peak test power (or toggles), we will next proceed to explain the proposed *Integrated Test Cube Ordering and X-filling Algorithm*. From now on, for the ease of explanation, we will use the words *test vector* and *test cube* interchangeably, without loss of generality.

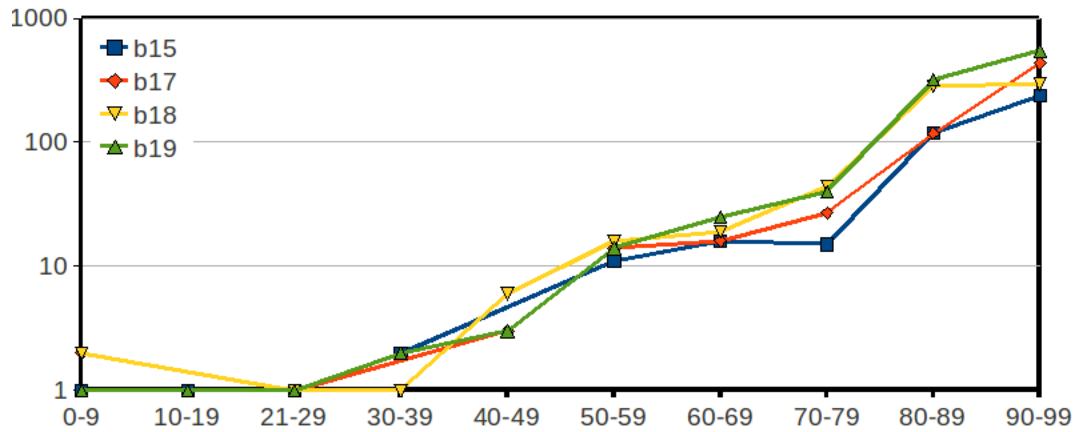


Figure 4.2: Don't care distribution in test cubes generated by commercial tool ; X axis: Percentage of X-bits in a given test cube ; Y axis: Number of test cubes

4.3.2 The X-Based Ordering Algorithm

We partition the proposed *Integrated Test Cube Ordering and X-filling Algorithm* into two phases: the first phase is the test cube ordering step, following which the second step of balanced X-filling is performed. The first step, which pertains to the ordering of the test cubes is performed according to Algorithm 7. This algorithm can be briefly explained as follows:

1. Line 2 of the algorithm sorts the input test cubes into non decreasing order of number of don't cares in the test cubes;
2. Lines 3 to 8 of the algorithm intersperse the test cubes; and

Running time of this algorithm is $O(nlog(n))$ where n is number of test cubes.

Next, we shall see the effectiveness of the proposed *X-Based Ordering Algorithm*.

4.3.3 Effectiveness of *X-Based Ordering Algorithm*

We introduce a new statistic called *X-base* to analyze the adjacency X-bit distribution in test cube pairs, for a given test cube ordering. Given a ordered set consisting of non-specified test cubes , we form a binary matrix by placing the test cubes in columns, as shown in Fig. 4.3. Each row corresponds to a Primary Input (PI) or Pseudo Primary

Algorithm 7: X-Based Test Cube Ordering Algorithm

Input: Non-Specified Test Cube Set $\{TC_1\ , TC_2, TC_3 \ldots TC_n\}$ in the order suggested by the Tool.

Output: Reordered Test Cube Set $\pi^{'} = \{TC_1\ , TC_2, TC_3 \ldots TC_n\}$

```
1 /* X-based TVO                                          */
```

2 Let $\pi = \{TC_1^{'}, TC_2^{'}, TC_3^{'} \ldots TC_n^{'}\}$ be an ordering of $\{TC_1, TC_2, TC_3 \ldots TC_n\}$ such that the number of X-bits in $TC_i^{'}$ is less than or equal to the number of X-bits in $TC_{i+1}^{'}$, for all $1 \leq i \leq n-1$;

3 **if** n *is even* **then**

4 $\quad\pi^{'} = \{T_1, T_2 \ldots T_n\} = \{TC_1^{'}, TC_n^{'}, TC_2^{'}, TC_{n-1}^{'}, TC_3^{'}, TC_{n-2}^{'} \ldots TC_{n/2}^{'}, TC_{n-(n/2-1)}^{'}\}$;

5 **end**

6 **else**

7 $\quad\pi^{'} = \{T_1, T_2 \ldots T_n\} = \{TC_1^{'}, TC_n^{'}, TC_2^{'}, TC_{n-1}^{'}, TC_3^{'}, TC_{n-2}^{'} \ldots TC_{\lfloor n/2 \rfloor}^{'}, TC_{n-(\lfloor n/2 \rfloor-1)}^{'}, TC_{\lceil n/2 \rceil}^{'}\}$;

8 **end**

Result: $return\ \pi^{'}$

Input (PPI), i.e., output of a scan flip flop. Hence each row is denoted with the label (P)PI, and the corresponding index as subscript.

Fig. 4.3 shows how to compute the *X-base* for adjacent test cubes TC_1 and TC_2. For every pair of test cubes, the *X-base* is initialized to zero. Each row in the sliding window contains two bits and all rows are sequentially visited to increment *X-base*. When a row is visited, even if one among the two bits is an X-bit, the *X-base* is incremented by one, before visiting next row. For TC_1, TC_2 pair shown in Fig. 4.3, we encounter four cases of $'XX'$ and one case of $'X0'$, making the *X-base* settle at 4+1=5. If we analyze the same for all adjacent column pairs in the binary matrix, we get a distribution for *X-base*. Let *MIN-X-BASE* be the $minimum$ of X-base values of all adjacent test cube pairs and *MAX-X-BASE* be the $maximum$ of X-base values of all adjacent test cube pairs.

Figure. 4.4 shows the MIN-X-BASE and MAX-X-BASE values for different benchmarks for Test cube ordering given by the Tool. It is interesting to note that the more the MIN-X-BASE value, there is huge scope for X-filling to reduce peak toggles. Since peak toggle computation requires a consideration of all the test cube pairs in the ordering, the *MIN-X-BASE* column is especially significant since it creates a bottleneck as to how much the peak toggles can be reduced by X-filling. It can be seen that for larger

Figure 4.3: Computing X-base metric, X-base $(TC_1,\ TC_2) = 5$

circuits, *MIN-X-BASE* is *an order of magnitude* smaller than *MAX-X-BASE*, showing a very weak scope for minimizing peak toggles. This motivates the need to reorder the test cubes to increase the *MIN-X-BASE*, thereby reducing peak toggles.

Figure. 4.5 shows the MIN-X-BASE and MAX-X-BASE values for different benchmarks for Test cube ordering given by the X-based TVO Algorithm. It is observed that by performing test cube ordering in this fashion, the *MAX-X-BASE* value remains as high as before but the *MIN-X-BASE* value approaches very close to *MAX-X-BASE* value. This is a very good sign, since bringing *MIN-X-BASE* close to *MAX-X-BASE* by retaining the *MAX-X-BASE* nearly intact signifies maximizing scope for peak toggle reduction.

This motivates that X-based test cube ordering has the potential to reduce the peak toggles during capture cycles.

Having seen the effectiveness of both X-based ordering and balanced X-filling algorithms, next we shall see how the combination of both minimizes the *bottleneck* toggle count. [1]

[1] It is to be noted that Figures 4.4 and 4.5 are having Y-axis in logarithmic scale, making the reductions very significant.

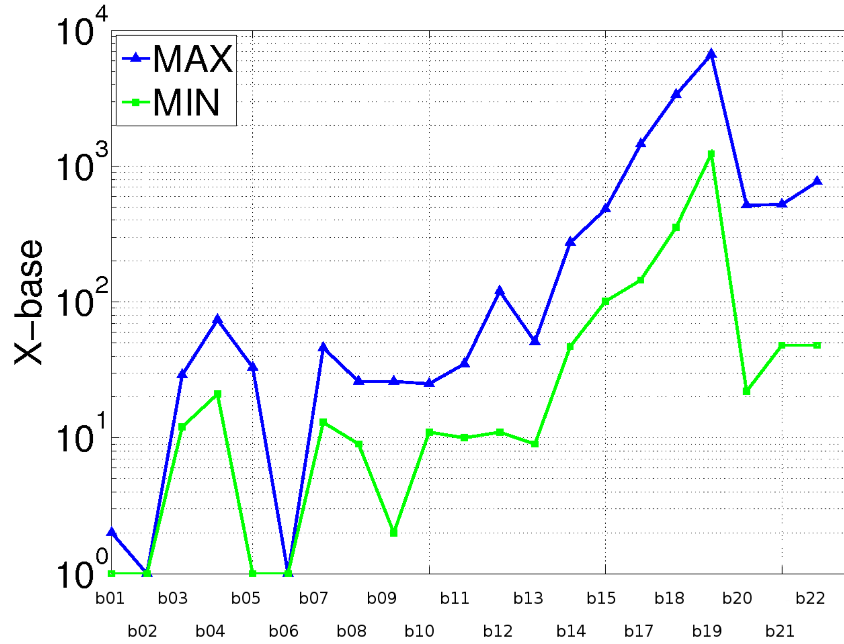


Figure 4.4: Gap between MAX-X-BASE and MIN-X-BASE for test cube ordering given by commercial tool

4.4 Integrated Test Vector Ordering and X-filling Algorithm

Algorithm 8 takes a set of input test cubes and finds the ordering of test cubes and filling of don't care bits such that peak input toggles is minimized.

Running time of this algorithm is $max(O(nlogn), O(nm))$, where m is number of bits in test cube and n is number of test cubes.

Next, we shall see the experimental results obtained by applying the proposed *Bottleneck Minimization Algorithm* on test sets of benchmark circuits.

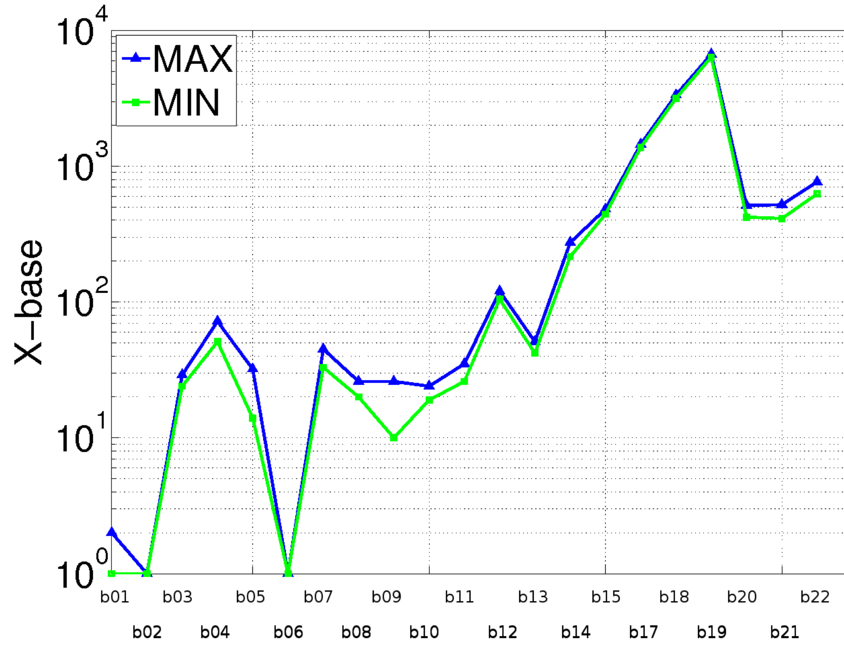


Figure 4.5: Gap between MAX-X-BASE and MIN-X-BASE for X-based test cube ordering

---

Algorithm 8: Bottleneck Minimization Algorithm

Input: $TC = TC_1, TC_2,.. \ TC_k$ be the set of input test cubes.

Output: TV = Sequence of fully specified input test vectors, $bottleneck_value$

1 `/* Reorder the test cubes by interspersing test cube with high don't cares with test cube with low don't cares */`

2 Let S be the Test cube sequence given by the Algorithm 7 by taking TC as input.

3 `/* Perform balanced don't care filling */`

4 Let TV be the Test vector sequence given by the Algorithm 6 by taking S as input.

5 `/* Compute bottleneck value for the given test vector sequence */`

6 Let $bottleneck_value$ be $max\{Hd(TV_1, TV_2), Hd(TV_2, TV_3), ...Hd(TV_{k-1}, TV_k)\}$, where $Hd(TV_i, TV_{i+1})$ is the Hamming distance between test vectors TV_i, TV_{i+1}

Result: $return \ TV, bottleneck_value$

Table 4.3: Peak input toggles : Tool-Ordering with different X-filling methods

Circuit	MT-Fill	R-Fill	0-Fill	1-Fill	B-Fill
b01	4	4	4	4	4
b02	4	4	4	4	4
b03	15	21	17	16	14
b04	41	50	47	45	39
b05	20	23	19	20	17
b06	4	4	5	4	4
b07	31	30	34	27	23
b08	20	20	20	18	14
b09	18	20	22	18	18
b10	12	19	17	15	10
b11	22	27	29	21	20
b12	63	76	62	89	59
b13	31	34	38	30	30
b14	181	180	194	159	157
b15	305	334	344	298	292
b17	916	923	943	880	871
b18	2134	2167	2251	2114	2066
b19	3926	4099	4201	3955	3819
b20	309	314	315	305	302
b21	317	307	315	305	276
b22	489	494	507	471	472

4.5 Experimental Results

4.5.1 Experimental Setup

We have considered the ITC'99 benchmark suite to validate our algorithms. A $45nm$ standard library is used for synthesis and placement. $DesignCompiler^{TM}$, $TetraMax^{TM}$ and $SoCEncounter^{TM}$ are used for Synthesis, ATPG and Place-And-Route (PAR) phases respectively. After PAR, using $SoCEncounter^{TM}$ interconnect capacitances are extracted to compute actual power values. Table 4.3, shows comparison of peak input toggles for various X-Filling methods w.r.t test cube ordering given by $TetraMax^{TM}$ (commercial tool). Table 4.4, shows comparison of peak input toggles for BTSP-Ordering applied after different X-fillings methods on vector sequence given by the commercial tool. We name this procedure as ISA. Table 4.5, shows comparison of peak input toggles for various X-Filling methods w.r.t Test Vector Ordering given by X-Base-Ordering. Table 4.6 shows Peak Input Toggles comparison between

Table 4.4: Peak input toggles : BTSP-Ordering followed by different X-filling methods

Circuit	MT-Fill	R-Fill	0-Fill	1-Fill	B-Fill
b01	2	2	2	2	2
b02	1	1	1	1	1
b03	10	12	11	10	11
b04	35	31	41	35	32
b05	12	13	12	12	13
b06	2	2	2	2	2
b07	20	21	28	18	21
b08	11	12	13	11	10
b09	12	12	11	12	12
b10	10	10	10	10	9
b11	15	15	13	12	14
b12	46	53	59	51	54
b13	22	22	23	20	22
b14	124	119	142	89	110
b15	226	219	231	172	200
b17	648	683	747	585	573
b18	1482	1604	1765	1384	1416
b19	2875	3235	3290	2609	2864
b20	242	234	265	214	238
b21	249	235	288	181	256
b22	364	350	407	324	360

proposed technique and existing techniques. Column 2 shows minimum input toggles among all existing X-filling methods for Vector ordering given by the tool (circled values from Table 4.3). Column 3 shows minimum input toggles among BTSP-Ordering applied after different X-fillings methods on vector sequence given by the commercial tool (circled values from Table 4.4). Column 4 shows minimum input toggles given by the method in (Wu *et al.*, 2011). Column 5 shows minimum toggles using proposed balanced-X-filling method for proposed X-Base Vector ordering. Columns 6,7 and 8 show percentage improvement of proposed technique over existing techniques. It is evident that proposed technique outperforms all existing techniques and percentage improvement is consistently increasing as circuit size increases. Similarly, Table 4.7 shows Peak Power comparison between proposed technique and existing techniques. Proposed technique outperforms all existing techniques and percentage improvement is consistently increasing as circuit size increases. We can observe that the magnitude of improvement in Tables 4.6 and 4.7 is not same. The difference is due to the fact

Table 4.5: Peak input toggles : X-Base-Ordering with different X-filling methods

Circuit	MT-Fill	R-Fill	0-Fill	1-Fill	B-Fill
b01	3	4	4	3	3
b02	4	4	4	4	4
b03	15	19	18	15	8
b04	45	52	47	43	25
b05	21	24	21	23	15
b06	5	4	5	5	5
b07	27	33	38	25	15
b08	16	20	18	15	8
b09	20	19	17	16	14
b10	14	20	16	14	10
b11	18	26	22	20	10
b12	60	76	99	68	31
b13	37	32	28	23	17
b14	181	164	208	152	79
b15	308	277	314	198	144
b17	912	774	953	680	421
b18	2130	1752	2200	1569	1011
b19	3926	3457	4340	3168	1877
b20	314	291	352	297	152
b21	288	290	346	237	130
b22	483	419	475	440	237

that the relation between Peak Input Toggles and Circuit Toggles is not perfectly linear and while computing Peak Power of the Circuit we need to consider interconnect capacitances into account. However our proposed method is outperforming all existing methods considerably both in Peak Input Toggles and Peak Circuit Power.

4.6 Summary

We have shown that test vector ordering or X-filling, when applied separately, are inadequate for producing the best possible savings in peak power dissipation during testing. We showed that the X-based test vector ordering method supplemented with balanced X-filling technique is shown to be very effective in reducing peak capture power, compared to other existing test vector ordering and X-filling techniques.

Table 4.6: Peak input toggles : Comparison of XStat-Method (X-Base-Ordering+B-Fill) over existing Ordering+Filling methods

	Peak Input Toggles				% Improvement of XStat-Method over		
Circuit	Tool	ISA Method	Adj-Fill Method	XStat Method	Tool	ISA Method	Adj-Fill Method
b01	4	2	4	3	25	-50	25
b02	4	1	3	4	0	-300	-33.33
b03	15	10	6	8	46.67	20	-33.33
b04	41	31	29	25	39.02	19.35	13.79
b05	19	12	19	15	21.05	-25	21.05
b06	4	2	4	4	0	-100	0
b07	27	18	17	15	44.44	16.67	11.76
b08	18	11	9	8	55.56	27.27	11.11
b09	18	12	17	14	22.22	-16.67	17.65
b10	12	10	9	10	16.67	0	-11.11
b11	21	12	18	10	52.38	16.67	44.44
b12	62	46	77	31	50	32.61	59.74
b13	30	20	26	17	43.33	15	34.62
b14	159	89	69	79	50.31	11.24	-14.49
b15	298	172	149	144	51.68	16.28	3.36
b17	880	585	438	421	52.16	28.03	3.88
b18	2114	1384	1065	1011	52.18	26.95	5.07
b19	3926	2609	2100	1877	52.19	28.06	10.62
b20	305	214	198	152	50.16	28.97	23.23
b21	305	181	182	130	57.38	28.18	28.57
b22	471	324	232	237	49.68	26.85	-2.16

Table 4.7: Peak circuit power : Comparison of XStat-Method (XBase-Ordering+B-Fill) over existing Ordering+Filling methods

	Peak Circuit Power (in μW)				% Improvement of XStat-Method over		
Circuit	Tool	ISA Method	Adj-Fill Method	XStat Method	Tool	ISA Method	Adj-Fill Method
b01	3.8	2.3	3.3	3.07	19.21	-33.48	6.97
b02	2.4	1.5	2.8	2.8	-16.67	-86.67	0
b03	6.3	4.63	4.6	3.95	37.3	14.69	14.13
b04	18.43	18.43	15.8	16.9	8.3	8.3	-6.96
b05	16	13.59	16.4	14.63	8.56	-7.65	10.79
b06	4.4	2.64	4.4	4.35	1.14	-64.77	1.14
b07	16.28	14.83	13.1	14.55	10.63	1.89	-11.07
b08	8.2	6.8	8.1	7.74	5.61	-13.82	4.44
b09	10.05	8.42	10.7	8.93	11.14	-6.06	16.54
b10	9.73	8.76	9	8.74	10.17	0.23	2.89
b11	16.37	15.36	15.2	14.58	10.93	5.08	4.08
b12	57.82	49.38	58.4	39.3	32.03	20.41	32.71
b13	18.04	13.69	15.1	14.65	18.79	-7.01	2.98
b14	102.6	101.7	99	86.46	15.73	14.99	12.67
b15	204.1	171	155.3	140.44	31.19	17.87	9.57
b17	1087.5	873.3	665.5	641.7	40.99	26.52	3.58
b18	3382.4	2405.3	2012.2	1761	47.94	26.79	12.48
b19	8014.7	6708.3	5885	4412.15	44.95	34.23	25.03
b20	255.2	243	214.8	202.62	20.6	16.62	5.67
b21	251.3	226.1	223.8	183.17	27.11	18.99	18.15
b22	395.6	372.8	328.9	304.75	22.97	18.25	7.34

Table 4.8: Computation time in performing test vector ordering

Circuit	# PIs	ISA	X-base	Speed Up
b01	5	0.1s	0.027s	3.7×
b02	4	0.05s	0.027s	2.0×
b03	29	0.24s	0.029s	8.1×
b04	77	0.42s	0.031s	13.9×
b05	35	3.73s	0.032s	116.7×
b06	5	0.243s	0.024s	10.1×
b07	50	0.227s	0.03s	7.6×
b08	30	2.80s	0.025s	112.0×
b09	29	0.05s	0.026s	2.0×
b10	28	0.19s	0.031s	6.2×
b11	38	7.31s	0.035s	208.9×
b12	126	0.95s	0.055s	17.2×
b13	53	1.37s	0.032s	42.8×
b14	275	12.87s	0.316s	40.7×
b15	485	221.48s	0.408s	542.8×
b17	1452	20.36s	1.752s	11.6×
b18	3357	39.34s	4.332s	9.1×
b19	6666	20.72s	12.309s	1.6×
b20	522	11.73s	0.508s	23.1×
b21	522	12.32s	0.514s	24.0×
b22	767	11.18s	0.685s	16.3×

CHAPTER 5

An Optimal X-Filling algorithm for Minimizing Peak Switching Activity

In the previous chapter, we have seen how *XStat* is capable of performing of simultaneous test vector ordering and X-filling to produce very effective savings in peak power dissipation during testing, and that the solutions converge very fast. It is also interesting to see if, for a given ordering of test vectors, there is an optimal way of filling the X-bits, such that the peak toggles at the inputs is minimized. Clearly, we cannot optimally fill the X-bits such that the peak circuit toggles is minimized, since it relates to the Boolean Satisfiability problem, which is NP-hard. Since we already know that input toggles correlate well to total circuit toggles (Girard *et al.*, 1998), we are interested to find an optimal way of filling the X-bits, so as to minimizing the peak input toggles during testing. Interestingly, the answer to this question is positive. We propose an algorithm using *Dynamic Programming*, that produces the optimal solution. The algorithm and its proof of optimality, can be explained as follows:

5.1 Peak Input Toggle Minimization Problem (PITMP)

Problem Definition: Given a combinational circuit C and a set of test cubes $TC = \{TC_1 \ldots TC_k\}$ the problem is to compute an ordering π of these test cubes and filling the don't cares to generate test vector sequence $T_{\pi_1},\ T_{\pi_2} \ldots T_{\pi_{k-1}},\ T_{\pi_k}$ such that the $max\{Hd(T_{\pi_1},\ T_{\pi_2}),\ Hd(T_{\pi_2},\ T_{\pi_3}) \ldots Hd(T_{\pi_{k-1}},\ T_{\pi_k})\}$ is *minimized*, where $Hd(T_{\pi_i},\ T_{\pi_{i+1}})$ is the Hamming distance between test vectors T_{π_i} and $T_{\pi_{i+1}}$. We decompose the solution into three components, which are explained in sections 5.2, 5.3, 5.4 and the final algorithm is explained in section 5.5.

5.2 Bottleneck Coloring Problem (BCP)

5.2.1 Problem Statement

Problem Definition in terms of Hotel Room Booking

Suppose a hotel received several guest requests for accommodation each of which is giving start date and end date of a time period, and asking the hotel to provide accommodation for exactly one day which falls in the given period. The aim of the hotel is to assign rooms to all guest requests such that number of guests staying in the hotel on any given day is minimized.

Mathematical Definition of Problem

- Let $S = (s_1, e_1), (s_2, e_2) \ldots (s_k, e_k)$ be a sequence of intervals such that s_i and e_i are integers corresponding to starting and ending times of interval i respectively, for all $1 \leq i \leq k$.
- Let $max_color = max(e_1,\ e_2,\ e_3, \ldots e_k)$.
- Let $min_color = min(s_1,\ s_2,\ s_3, \ldots s_k)$.
- Let $\{\ c_{min_color},\ c_{min_color+1},\ c_{min_color+2} \ldots c_{max_color}\ \}$ be a set of colors.
- For each interval (s_i, e_i) assign a color c_j such that $s_i \leq j \leq e_i$.
- Let $h_{min_color},\ h_{min_color+1},\ h_{min_color+2} \ldots h_{max_color}$ be a sequence of integers such that h_j be the number of intervals which are assigned color c_j.
- Our objective is to assign colors to intervals such that $max(h_{min_color},\ h_{min_color+1},\ h_{min_color+2} \ldots h_{max_color})$ is minimized.

Each interval corresponds to an accommodation request in the subsection 5.2.1. Each color corresponds to a day. Assigning color c_j to the interval (s_i, e_i) is same as allocation of hotel room on j^{th} day to this request. Note that h_j denotes the number of guests who are assigned room on j^{th} day.

Algorithm 9: Algorithm for Computing Lower-Bound

Input: S= (s_1, e_1),$(s_2, e_2) \dots (s_k, e_k)$ be a sequence of intervals
Output: *Lower-Bound Value.*

1 Let $\pi_1, \pi_2, ...\pi_{m-1}, \pi_m$ be the increasing sorted sequence of distinct possible values in the sequence $s_1, e_1, s_2, e_2 \dots s_{k-1}, e_{k-1}, s_k, e_k$
2 Let $T_{i,j}$, where $i \leq j$, denotes number of intervals whose starting time is $\geq \pi_i$ and ending time is $\leq \pi_j$, where $1 \leq i \leq j \leq m$;
3 If $i > j$ then let $T_{i,j}$ = 0 else $T_{i,j}$ can be expressed recursively as follows : $T_{i,j}$ = $T_{i,j-1}$ + $T_{i+1,j}$ - $T_{i+1,j-1}$ + Number of intervals whose staring time is equal to π_i and ending time is equal to π_j.
4 /* Note that $T_{i+1,j-1}$ is subtracted since the set of intervals whose starting time is at least π_{i+1} and ending time is at most π_{j-1} are counted in both $T_{i,j-1}$,$T_{i+1,j}$. */
5 $Lowerbound\ LB = max\{\lceil T_{i,j}/(\pi_j - \pi_i + 1)\rceil | 1 \leq i \leq j \leq m\}$
6 /* If we take any interval whose starting time is at least π_i and ending time at most π_j then we should assign a color c_k to this interval such that $\pi_i <= k <= \pi_j$. This means there exists a color c_k such that at least $\lceil T_{i,j}/(\pi_j - \pi_i + 1)\rceil$ intervals are assigned color c_k, where $\pi_i <= k <= \pi_j$ */

Result: $return\ LB$

5.2.2 Dynamic Programming Algorithm to compute Lower-Bound (LB) for *Bottleneck Coloring Problem*

Algorithm 9 gives the lower bound on the number of intervals which are assigned the same color. Running time of this algorithm is $O(k^2)$, where k is the number of intervals.

5.2.3 Greedy Algorithm for *Bottleneck Coloring Problem*

Algorithms 10 assign colors to intervals such that for each interval (s_i,e_i) it assigns a color c_j such that $s_i \leq j \leq e_i$ and maximum number of intervals which are assigned the same color is at most the lower bound value computed in Algorithm 9. Running time of this algorithm is $O(klogk)$, where k is the number of intervals.

Algorithm 10: Algorithm for assigning color to intervals

Input: S= (s_1, e_1),(s_2, e_2) $\ldots$ (s_k, e_k) be a sequence of intervals, LB - lower-bound

Output: Intervals with assigned colors

1 Sort the intervals in S based on starting time.

2 Let *H be a min heap*. Each node of this heap can store information of an interval (starting time and ending time). Nodes of this heap are ordered by ending times of intervals i.e ending time of interval stored in a node is less than or equal to ending times of intervals stored in that node's children.

3 **for** $i = 1 \rightarrow n$ **do**

4 Insert into heap H all intervals whose starting time is equal to i.

5 `/* if we take any interval in` H `starting time is at most` i`. */`

6 Greedily remove top l elements from heap and assign color c_i, where $l = min(heap_size, LB)$;

7 `/* The reason for picking top elements and assigning colors` c_i `is we want to assign colors to intervals which are ending soon. We prove in section` *Proof of correctness* `that ending times of all these removed intervals are at least` i`. */`

8 **end**

5.2.4 Proof of correctness

In the following paragraph we will prove that at the end of ith iteration of the above algorithm ending times of all intervals contained in *min heap* are greater than i. This means each interval (s_i,e_i) it assigned a color c_j such that $s_i \leq j \leq e_i$.

Suppose at the end of some iteration i *min heap* contains an interval whose ending time is less than or equal to i. Let i be such that it's value is minimum. Let $j < i$ such that number of intervals which are assigned color in j^{th} iteration is less than *lower bound*. Let j be such that it's value is maximum. If there is no such a j then let $j = 0$. We selected j such that heap became empty after iteration j, and in each iteration from iteration $j+1$ to i, number of intervals assigned color are exactly equal to lower bound. Let $j < k < i$ such that in the kth iteration the above algorithm assigned color to an interval whose ending time is more than i. Let k be such that it's value is maximum. If there is no such k then let $k = j$. We selected k such that, all intervals which are assigned color from iteration $k + 1$ and iteration i have ending times $\leq i$ and their starting times cannot be less than $k + 1$, as we assigned color to an interval whose

ending time is more than i in k^{th} iteration. Ending times and starting times of all intervals which are assigned color from $k+1^{th}$ iteration to i^{th} iteration are less than or equal to i and greater than k respectively. Number of intervals which are assigned color in from $k+1^{th}$ iteration to i^{th} is equal to $lowerbound*(i-k)$ and *min heap* contains an interval whose ending time is equal to i and starting time is greater than k. This implies number of intervals whose starting time is greater than k and ending time is less than or equal to i is more than $lowerbound*(i-k)$, which is a contradiction.

5.3 Optimal X-Filling Algorithm

Problem Definition: Given a sequence of test cubes $TC_1, TC_2, ...TC_n$ each of length m, replace each don't care in test cubes by either 0 or 1 such that $max\{Hd(TC_1,\ TC_2),\ Hd(TC_2, TC_3)\dots Hd(TC_{n-1},\ TC_n)\}$ is *minimized*, where $Hd(TC_i,\ TC_{i+1})$ is the Hamming distance between test cubes TC_i and TC_i+1, after replacing don't cares by either 0 or 1.

5.3.1 Motivation

The X-Stat algorithm follows a two phase approach. In the first phase, it uses adjacent X-fill technique to convert don't care (X-bit) stretches $0XX...X1$ and $1XX...X0$ into smaller X-bit stretches $0X1$ and $1X0$ respectively as shown in *Phase 1* column of Fig 5.1. In the second phase, it replaces X-bits by either 0 or 1 in order to minimize peak toggles as shown in *Phase 2* column of Fig 5.1. Because of greedy approach used in *Phase 1*, it does not achieve the global optimal-fill for peak toggle reduction, as shown in *Optimum-Fill* column of Fig 5.1. Motivated by this, we choose a *Dynamic Programming* paradigm which takes global picture into consideration and optimally fill the X-bits with binary values to achieve the best reduction in peak toggles.

5.3.2 Algorithm Details

In this section we will reduce the above problem to an instance of *Bottleneck Coloring Problem (BCP)* explained in section 5.2 and use the algorithm for *Bottleneck Coloring*

Vector Sequence					X-Stat										Optimum-Fill				
					Phase 1					Phase 2									
V1	V2	V3	V4	V5	V1	V2	V3	V4	V5	V1	V2	V3	V4	V5	V1	V2	V3	V4	V5
0	X	X	1	X	0	0	X	1	1	0	0	0	1	1	0	1	1	1	1
1	X	X	X	0	1	1	1	X	0	1	1	1	0	0	1	1	1	1	0
0	X	X	1	X	0	0	X	1	1	0	0	1	1	1	0	0	1	1	1
1	X	X	X	0	1	1	1	X	0	1	1	1	0	0	1	1	1	1	0
1	X	1	X	0	1	1	1	X	0	1	1	1	1	0	1	1	1	0	0
X	0	X	X	1	0	0	0	X	1	0	0	0	0	1	0	0	1	1	1
1	0	X	X	1	1	0	0	X	1	1	0	0	0	1	1	0	0	1	1
										1	1	3	3		2	2	2	2	

Figure 5.1: Motivation for Optimum-X-Filling (O-Fill)

Problem (BCP) for computing an optimal solution. We explained reduction process and construction of solution in Algorithm 11.

5.4 Test Vector Ordering Algorithm

5.4.1 Motivation

For a given any sequence of test cubes, Algorithm 11 replaces don't cares by either 0 or 1 to minimize the peak input toggles. If lengths of don't care stretches in rows of matrix A defined in Algorithm 11 are sufficiently large, then this algorithm has more freedom to decide the positions of toggles which in turn minimize the peak input toggles. To achieve such a large don't cares stretches in the rows of matrix A we propose the following test vector ordering Algorithm 12, we call this ordering as Interleaved - Test Vector Ordering (I-Ordering).

5.4.2 Algorithm Details

Algorithm 12 takes an input test cube sequence TC and an integer k (interleave count) and outputs a re-ordered test cube sequence S.

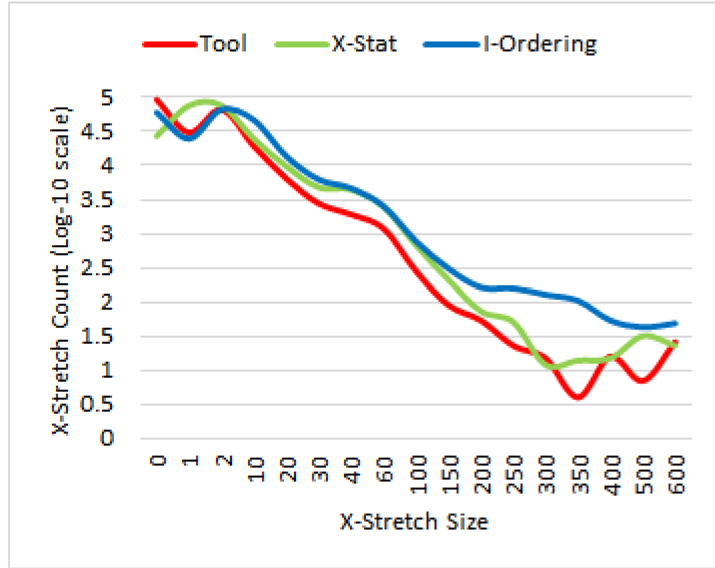


Figure 5.2: `b19` don't care-stretch analysis (Tool vs X-Stat vs I-Ordering)

5.4.3 Experimental Results

In Fig 5.2 x-axis shows different don't care stretch (0XX..X1 and 1XX..X0) sizes and y-axis shows number of such don't care stretches for Tool,X-Stat and I-Ordering for b19. One can observe that I-Ordering increasing the sizes of don't care stretches which are exploited by the Algorithm 10.

5.5 Bottleneck Minimization Algorithm

Algorithm 13 takes a set of input test cubes TC and finds the ordering of test cubes and filling of don't care bits such that peak input toggles is minimized. Fig 5.3 shows the plot between Number of iterations and Peak input toggles. For each benchmark, the number of interations corresponding to lowest peak input toggles is chosen. Fig 5.4 shows the plot between this chosen iteration count and the number of test cubes. This figure shows that iteration count varies as $log(n)$. Thus the number of times while loop in the Algorithm 13 executed is $O(log(n))$, where n is number of test cubes.

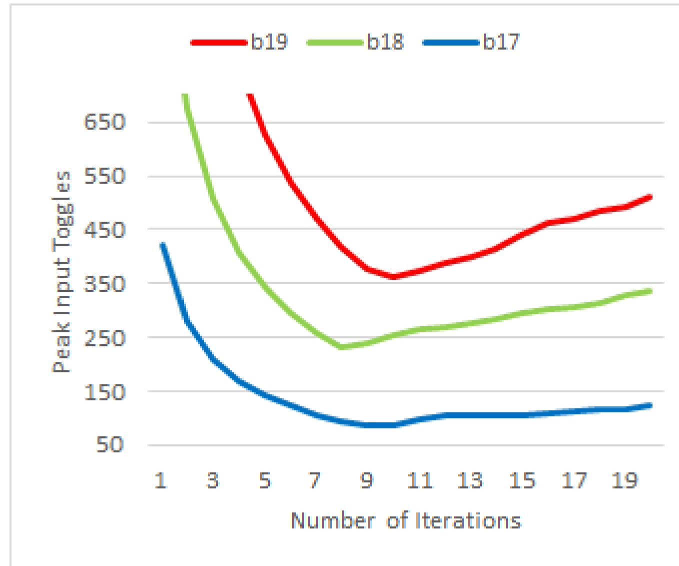


Figure 5.3: Bottleneck minimization algorithm iterations: Number of iterations vs Peak input toggles

5.6 Experimental Results

5.6.1 Experimental Setup

We have considered the ITC'99 benchmark suite to validate our algorithms. A $45nm$ standard library is used for synthesis and placement. $DesignCompiler^{TM}$, $TetraMax^{TM}$ and $SoCEncounter^{TM}$ are used for Synthesis, ATPG and Place-And-Route (PAR) phases respectively. After PAR, using $SoCEncounter^{TM}$ interconnect capacitances are extracted to compute actual power values. The test cubes for large circuits are typically dominated by don't care (X) bits as shown in Table 5.1, making X-filling an effective technique for minimizing peak test power.

5.6.2 Results

Table 5.2, shows comparison of peak input toggles for various X-Filling methods w.r.t Test Vector Ordering given by the Tool. Table 5.3, shows comparison of peak input toggles for BTSP-Ordering applied after different X-fillings methods on vector sequence given by the commercial tool. We name this procedure as ISA. Table 5.4, shows comparison of peak input toggles for various X-Filling methods w.r.t Test Vector Ordering

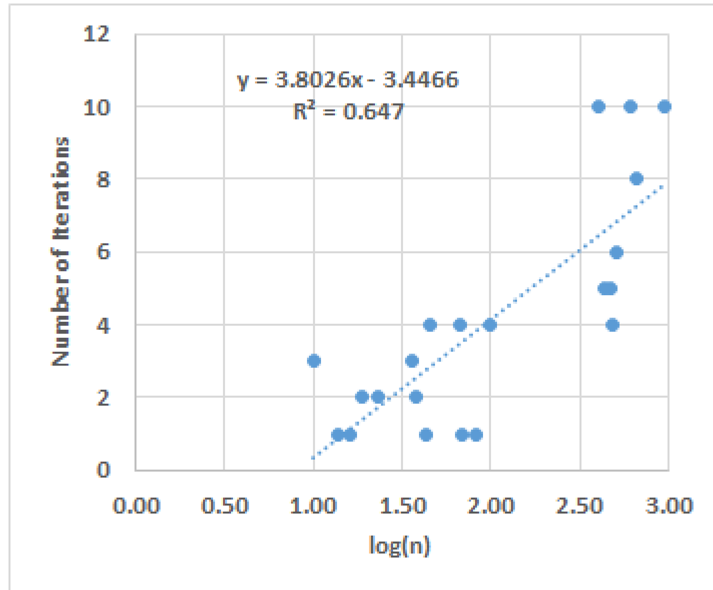


Figure 5.4: Bottleneck minimization algorithm iterations: Optimum number of iterations vs log(n)

given by $XStat$ (X-Base Ordering). In tables 5.2, 5.3 and 5.4 graded cell shows best X-filling method among all X-filling methods. We can observe that O-Fill X-Filling method is performing better than all other X-Filling methods for these three test vector ordering techniques.

Table 5.6 shows Peak Input Toggles comparison between proposed technique and existing techniques. Column 2 shows minimum input toggles among all existing X-filling methods for Vector ordering given by the Tool (circled values from Table 5.2). Column 3 shows minimum input toggles among BTSP-Ordering applied after different X-fillings methods on vector sequence given by the commercial tool (circled values from Table 5.3). Column 4 shows minimum input toggles given by the method in (Wu *et al.*, 2011). Column 5 shows minimum input toggles among all existing X-filling methods for Vector ordering given by X-Base Ordering (circled values from Table 5.4). Column 6 shows minimum toggles using proposed O-filling method for proposed Vector ordering (I-Ordering) scheme.

Columns 7,8,9 and 10 show percentage improvement of proposed technique over existing techniques. It is evident that proposed technique outperforms all existing techniques and percentage of improvement is consistently increasing as circuit size increases.

Similarly Table 5.7 shows Peak Power comparison between proposed technique and existing techniques. Proposed technique outperforms all existing techniques and percentage improvement consistently increasing as circuit size increases. We can observe that the magnitude of improvement in tables 5.6 and 5.7 is not same. The difference is due to the fact that the relation between Peak Input Toggles and Circuit Toggles is not perfectly linear and while computing Peak Power of the Circuit we need to consider interconnect capacitances into account. However our proposed method is outperforming all existing methods considerably both in Peak Input Toggles and Peak Circuit Power.

In (Girard *et al.*, 1998), it was shown that there is a strong correlation between input toggles and internal toggles inside the circuit. Based on this assumption, we went ahead to find an optimal algorithm that will minimize the input toggles to the circuit during the testing phase. In the next section, we relax this assumption and try to search for solutions near the solution so-far obtained, using the local search technique and observe that the savings is marginal, thereby proving the effectiveness of the proposed technique.

5.7 Local Search With Iterative 1-bit Neighbourhood

We denote $\mathcal{S}_{DP-fill}$ as the solution obtained using DP-fill suggested in this thesis. In every iteration, $\mathcal{S}_{cur}$ stands for the best-so-far solution in the current iteration of the local search technique. The local search technique used to prune the solutions generated by DP-fill is outlined in Figure 5.5. Although we have adhered to 1-bit neighbourhood in this thesis, in principle, the local search technique shown in Figure 5.5, can be extended to n-bit neighbourhood, for a given n, in a straightforward manner. However, it should be noted that searching all the possible n-bit neighbourhoods ($1 \leq n \leq T$, where T is test vector size) is intractable, because the size of the search space is $\sum_{n=1}^{T} \binom{T}{n} = 2^T$. The results obtained by applying the described local search technique for $greedy$ as well as $SimulatedAnnealing(SA)$ strategies is shown in Table 5.8. It can be seen that the savings is marginal, thereby validating our idea of optimal minimization of input toggles as an effective technique for minimizing peak power dissipation during testing.

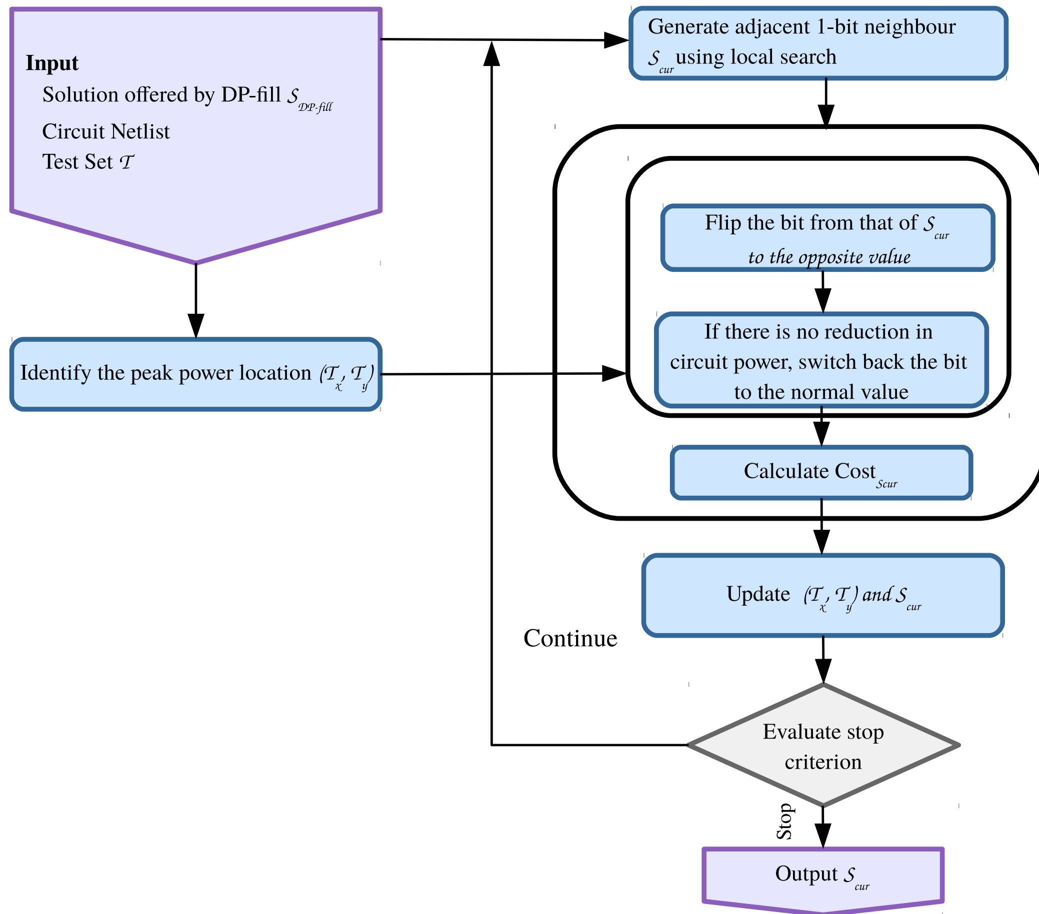


Figure 5.5: Flow chart description of the local search technique with 1-bit neighbourhood

Algorithm 11: Optimal X-Filling Algorithm

Input: $TC = TC_1, TC_2, .. TC_n$ be the sequence of input test cubes

Output: $T = T_1, T_2, .. T_n$ sequence of completed specified test vectors and $bottleneck_value$

1 Let $TC_1, TC_2, ...TC_n$ be a sequence of test cubes each of length m

2 Construct a $m \times n$ matrix A such that i^{th} column of A is equal to the test cube TC_i.

3 **for** $i = 1 \to m$ **do**

4   /* Preprocessing of 0XX..X0,1XX..X1 stretches */

5   *If* { i^{th} row contain a subsequence 0XX...X0} then replace every don't care in this subsequence by zero since there exists an optimal solution in which all of these don't cares are replaced by zeros irrespective of how other don't cares are replaced.

6   *If* { i^{th} row contain a subsequence 1XX...X1} then replace every don't care in this subsequence by one since there exists an optimal solution in which all of these don't cares are replaced by ones irrespective of how other don't cares are replaced.

7 **end**

8 Let S=ϕ

9 **for** $i = 1 \to m$ **do**

10   /* Creating intervals for 0XX..X1,1XX..X0 */

11   If there exist $k < l$ such that $A_{i,k}$=0,$A_{i,l}$=1 and $A_{i,k+1}$...$A_{i,l-1}$ are don't cares then append an interval $(k, l-1)$ to sequence of intervals S.

12   *Comment :* Note that there exists an optimal solution such that $A_{i,k}$=0,$A_{i,k+1}$=0,..., $A_{i,j}$=0, $A_{i,j+1}$=1, $A_{i,j+2}$=1,..., $A_{i,l}$=1, where $k \leq j < l$, irrespective of how other don't cares are replaced. There is only one toggle between j^{th} and $j + 1^{th}$ test vectors in this subsequence. The color assigned by the Algorithm 10 called in $line$ 17 to this newly added interval captures the location of this toggle in this subsequence.

13   If there exist $k < l$ such that $A_{i,k}$=1,$A_{i,l}$=0 and $A_{i,k+1}$...$A_{i,l-1}$ are don't cares then append an interval $(k, l-1)$ to sequence of intervals S.

14   *Comment :* Note that there exists an optimal solution such that $A_{i,k}$=1,$A_{i,k+1}$=1,..., $A_{i,j}$=1, $A_{i,j+1}$=0, $A_{i,j+2}$=0,..., $A_{i,l}$=0, where $k \leq j < l$, irrespective of how other don't cares are replaced. There is only one toggle between j^{th} and $j + 1^{th}$ test vectors in this subsequence. The color assigned by the Algorithm 10 called in $line$ 17 to this newly added interval captures the location of this toggle in this subsequence.

15 **end**

16 Let $bottleneck_value$ be the lower-bound value computed using Algorithm 9 by giving S as input.

17 Construct optimal bottleneck solution for S using Algorithm 10 by giving S,$bottleneck_value$ as input.

18 /* Constructing Optimal solution for *X-Filling* */

19 Suppose color c_j is assigned to interval (s_i, e_i) in the optimal solution given by Algorithm 10. Look at the row in matrix A correspond to interval (s_i, e_i), make all bits from column s_i to j same as bit value at column s_i and make all bits from column $j + 1$ to $e_i + 1$ same as bit value at column $e_i + 1$

20 Let $T = T_1, T_2, .. T_n$ be the columns of matrix A.

Result: $return\ T, bottleneck_value$

Algorithm 12: Test Vector Ordering Algorithm

Input: $TC = TC_1, TC_2,.. TC_n$ be the sequence of input test cubes
Input: k = an integer
Output: S = Reordered sequence of input test cubes TC.

```
1  Let S = ∅
2  for i = 1 → ⌊n/(k+1)⌋ do
3      /* pick i^th vector from TC and append to S */
4      S = S,T'_i
5      /* pick n − (i − 1) ∗ k th vector to n − (i − 1) ∗ k − k + 1 th
          vector from T' and append to S */
6      S = S,T'_{n−(i−1)∗k}, T'_{n−(i−1)∗k−1}, ..T'_{n−(i−1)∗k−k+1}
7  end
8  Select all the vectors in T' which are not in S and add them to S, there can be at
   most k such vectors.
```

Result: $return\ S$

Algorithm 13: Bottleneck Minimization Algorithm

Input: $TC = TC_1, TC_2,.. TC_n$ be the set of input test cubes.
Output: TVS = Sequence of fully specified input test vectors.

```
1  /* Sort the test cubes in non decreasing order of
      number of don't cares */
2  Let TC' = TC^1_1, TC'_2,.. TC'_n be an ordering of input test cubes such that number
   of don't cares in TC'_i ≤ TC'_{i+1} where 1 ≤ i < n.
3  Let current_optimal_value = ∞
4  Let current_k = 0
5  Let exit_flag = false
6  while exit_flag = false do
7      Let current_k = current_k + 1
8      /* Reorder the test cubes by interspersing test
          cube with high don't cares with test cube with
          low don't cares */
9      Let S be the Test cube sequence given by the Algorithm 12 with input TC', k.
10     /* Compute bottleneck value for the given test cube
          sequence */
11     Let temp_optimal_value be the optimal bottleneck value computed on
       sequence S using Algorithm 11
12     if temp_optimal_value < current_optimal_value then
13         current_optimal_value = temp_optimal_value;
14     else
15         exit_flag = true;
16     end
17 end
```

Result: $return\ S$

Table 5.1: ITC'99 benchmarks (X % : Average % of X-bits in test cubes)

Benchmark	# PIs	# Gates	# Test Cubes	X %
b01	5	57	14	7.14
b02	4	31	10	5.00
b03	29	103	19	70.42
b04	77	615	67	64.35
b05	35	608	69	36.77
b06	5	60	16	12.50
b07	50	431	46	58.57
b08	30	196	38	60.44
b09	29	162	23	38.23
b10	28	217	43	58.72
b11	38	574	83	64.11
b12	126	1.6K	100	76.94
b13	53	596	36	65.41
b14	275	5.4K	511	77.90
b15	485	8.7K	405	87.75
b17	1452	27.99K	618	89.85
b18	3357	75.8K	666	86.92
b19	6666	146.5K	953	89.81
b20	522	9.4K	476	75.29
b21	522	9.4K	479	73.20
b22	767	13.4K	435	74.05

Table 5.2: Peak input toggles : Tool-ordering with different X-filling methods

Circuit	MT-Fill	R-Fill	0-Fill	1-Fill	B-Fill	O-Fill
b01	4	4	4	4	4	4
b02	4	4	4	4	4	4
b03	15	21	17	16	14	14
b04	41	50	47	45	39	39
b05	20	23	19	20	17	17
b06	4	4	5	4	4	4
b07	31	30	34	27	23	23
b08	20	20	20	18	14	12
b09	18	20	22	18	18	18
b10	12	19	17	15	10	10
b11	22	27	29	21	20	20
b12	63	76	62	89	59	58
b13	31	34	38	30	30	29
b14	181	180	194	159	157	156
b15	305	334	344	298	292	282
b17	916	923	943	880	871	841
b18	2134	2167	2251	2114	2066	2009
b19	3926	4099	4201	3955	3819	3753
b20	309	314	315	305	302	299
b21	317	307	315	305	276	260
b22	489	494	507	471	472	466

Table 5.3: Peak input toggles : BTSP-Ordering followed by different X-filling methods

Circuit	MT-Fill	R-Fill	0-Fill	1-Fill	B-Fill	O-Fill
b01	2	2	2	2	2	1
b02	1	1	1	1	1	1
b03	10	12	11	10	8	11
b04	35	31	41	35	32	36
b05	12	13	12	12	13	12
b06	2	2	2	2	2	2
b07	20	21	28	18	21	20
b08	11	12	13	11	10	12
b09	12	12	11	12	12	12
b10	10	10	10	10	9	10
b11	15	15	13	12	14	15
b12	46	53	59	51	54	46
b13	22	22	23	20	22	22
b14	124	119	142	89	110	124
b15	226	219	231	172	200	224
b17	648	683	747	585	573	648
b18	1482	1604	1765	1384	1416	1473
b19	2875	3235	3290	2609	2864	2861
b20	242	234	265	214	238	242
b21	249	235	288	181	256	252
b22	364	350	407	324	360	364

Table 5.4: Peak input toggles : X-Base-Ordering with different X-filling methods

Circuit	MT-Fill	R-Fill	0-Fill	1-Fill	B-Fill	O-Fill
b01	3	4	4	3	3	3
b02	4	4	4	4	4	4
b03	15	19	18	15	8	7
b04	45	52	47	43	25	24
b05	21	24	21	23	15	14
b06	5	4	5	5	5	4
b07	27	33	38	25	15	14
b08	16	20	18	15	8	7
b09	20	19	17	16	14	14
b10	14	20	16	14	10	7
b11	18	26	22	20	10	9
b12	60	76	99	68	31	31
b13	37	32	28	23	17	17
b14	181	164	208	152	79	79
b15	308	277	314	198	144	144
b17	912	774	953	680	421	421
b18	2130	1752	2200	1569	1011	1008
b19	3926	3457	4340	3168	1877	1877
b20	314	291	352	297	152	152
b21	288	290	346	237	130	130
b22	483	419	475	440	237	234

Table 5.5: Peak input toggles : I-Ordering with different X-filling methods

Circuit	MT-Fill	R-Fill	0-Fill	1-Fill	B-Fill	O-Fill
b01	3	4	4	3	3	3
b02	3	3	3	3	3	3
b03	12	19	15	15	8	6
b04	41	45	43	39	23	15
b05	20	22	21	23	15	14
b06	4	4	4	4	4	4
b07	24	31	38	23	15	11
b08	16	18	16	14	8	6
b09	14	18	16	16	11	11
b10	10	18	14	13	9	7
b11	15	25	22	18	10	9
b12	59	72	99	65	30	15
b13	28	31	28	23	15	10
b14	168	158	208	148	77	40
b15	296	267	314	193	141	33
b17	882	770	953	676	419	85
b18	2030	1741	2200	1550	980	232
b19	3862	3436	4340	3167	1871	364
b20	301	285	352	284	143	65
b21	280	286	333	237	129	67
b22	451	409	475	425	210	91

Table 5.6: Peak input toggles : Comparison of DP-Method (I-Ordering+O-Fill) over existing Ordering+Filling methods

	Peak Input Toggles					% Improvement of DP-Fill Method over			
Circuit	Tool	ISA Method	Adj-Fill Method	XStat Method	DP-Fill Method	Tool	ISA Method	Adj-Fill Method	XStat Method
b01	4	2	4	3	3	25	-50	25	0
b02	4	1	3	4	3	25	-200	0	25
b03	14	8	6	8	6	57.1	25	0	25
b04	39	31	29	25	15	61.5	51.6	48.3	40
b05	17	12	19	15	14	17.6	-16.7	26.3	6.7
b06	4	2	4	4	4	0	-100	0	0
b07	23	18	17	15	11	52.2	38.9	35.3	26.7
b08	14	10	9	8	6	57.1	40	33.3	25
b09	18	11	17	14	11	38.9	0	35.3	21.4
b10	10	9	9	10	7	30	22.2	22.2	30
b11	20	12	18	10	9	55	25	50	10
b12	59	46	77	31	15	74.6	67.4	80.5	51.6
b13	30	20	26	17	10	66.7	50	61.5	41.2
b14	157	89	69	79	40	74.5	55.1	42	49.4
b15	292	172	149	144	33	88.7	80.8	77.9	77.1
b17	871	573	438	421	85	90.2	85.2	80.6	79.8
b18	2066	1384	1065	1011	232	88.8	83.2	78.2	77.1
b19	3819	2609	2100	1877	364	90.5	86	82.7	80.6
b20	302	214	198	152	65	78.5	69.6	67.2	57.2
b21	276	181	182	130	67	75.7	63	63.2	48.5
b22	471	324	232	237	91	80.7	71.9	60.8	61.6

Table 5.7: Peak circuit power : Comparison of DP-Method (I-Ordering+O-Fill) over existing Ordering+Filling methods

	Peak Circuit Power (in μW)					% Improvement of DP-Fill Method over			
Circuit	Tool	ISA Method	Adj-Fill Method	XStat Method	DP-Fill Method	Tool	ISA Method	Adj-Fill Method	XStat Method
b01	3.8	2.3	3.3	3.1	3.1	18.8	-33.1	6.1	0
b02	2.4	1.5	2.8	2.6	2.6	-6.2	-68.3	7.3	0
b03	5.6	4	4.6	3.9	4.2	25	-5.5	9.2	-5.6
b04	17.2	17.1	15.8	16.9	14.8	14	13.9	6.6	12.7
b05	15.6	13.6	16.4	14.6	14.9	4.4	-9.8	9	-2
b06	4.4	2.6	4.4	4.3	4.4	0.9	-67.2	-0.1	-1.7
b07	15.7	14.8	13.1	14.6	13.3	15.7	10.6	-1.5	8.9
b08	7.8	6.8	8.1	7.7	6.3	18.5	6.8	21.5	18.1
b09	9.8	8.4	10.7	8.9	7.4	24.7	12.1	30.8	17.2
b10	9.3	8.8	9	8.7	8.2	11.6	6.5	9.2	6.3
b11	16.4	15.4	15.2	14.6	13.9	15.2	9.6	8.9	4.8
b12	56.5	49.4	58.4	39.3	36.4	35.5	26.3	37.6	7.2
b13	18	13.7	15.1	14.7	10.9	39.4	20.1	27.6	25.3
b14	99.3	101.7	99	86.5	85.4	14	16.1	13.8	1.3
b15	197.1	171	155.3	140.4	122	38.1	28.7	21.4	13.1
b17	1085.5	847.1	665.5	641.7	431.6	60.2	49.1	35.1	32.7
b18	3350.7	2405.3	2012.2	1761	1192	64.4	50.4	40.8	32.3
b19	7621.6	6708.3	5885	4135	2699.4	64.6	59.8	54.1	34.7
b20	252.8	243	214.8	202.6	195.3	22.7	19.6	9.1	3.6
b21	248.4	226.1	223.8	183.2	166.4	33	26.4	25.6	9.2
b22	395.6	372.8	328.9	304.8	277.1	30	25.7	15.8	9.1

Table 5.8: Additional Peak Power Savings obtained by Local Search Technique with 1-bit Neighbourhood

		Greedy Pruning			SA Pruning		
Circuit	DP-fill (in μW)	DP-fill + Greedy Pruning(in μW)	%Improvement	Additional Simulation time	DP-fill + SA Pruning(in μW)	%Improvement	Additional Simulation time
b01	3.07	3.07	0	0s	3.07	0	0.001s
b02	2.6	2.19	15.67	0m0.03s	2.19	15.59	0m0.481s
b03	4.17	4.12	1.18	0m0.64s	4.12	1.3	0m4.728s
b04	14.76	13.23	10.36	0m6.83s	13.23	10.39	1m49.907s
b05	14.92	14.91	0.09	0m0.91s	14.86	0.43	0m13.423s
b06	4.35	4.28	1.68	0m0.06s	4.28	1.63	0m0.208s
b07	13.26	12.24	7.71	0m3.10s	12.24	7.68	0m26.645s
b08	6.89	6.79	1.47	0m0.48s	6.79	1.54	0m8.717s
b09	7.4	6.94	6.15	0m0.17s	6.94	6.1	0m0.790s
b10	8.19	8.03	1.92	0m0.41s	8.03	1.93	0m13.768s
b11	13.88	13.88	0	0m1.023s	13.88	0	0m26.877s
b12	36.42	36.12	0.82	0m30.62s	36.12	0.83	4m23.086s
b13	10.94	10.79	1.39	0m1.68s	10.79	1.36	0m21.156s
b14	85.37	82.78	3.03	2m47.09s	81.48	4.56	8m13.430s
b15	122.01	113.73	6.78	66m14.21s	117.18	3.96	39m20.670s
b17	431.6	422.17	2.19	45h37m37s	424.57	1.63	28h10m45s
b18	1192.03	1179.93	1.02	46h7m22s	1184.7	0.61	28h40m35s
b19	2699.35	2696.11	0.12	47h35m47s	2696.11	0.12	30h14m20s
b20	195.34	190.22	2.62	57m0.19s	189.76	2.86	56m56.429s
b21	166.38	161.54	2.91	3m35.40s	161.54	2.91	20m44.411s
b22	277.07	265.98	4.0	14h15m51s	267.39	3.5	9h12m25s
Average			3.39			3.28	

5.8 Summary

We mapped the problem of X-filling to an variant of $interval\ coloring\ problem$ called $bottleneck\ coloring\ problem$ and proposed dynamic programming based algorithm for optimal X-filling such that peak input toggles is minimized. This algorithm obtains the optimal solution for minimizing peak input toggles. Since input toggles is well correlated to circuit power (Girard *et al.*, 1998), we assume that the proposed algorithm automatically generates a good solution that minimizes peak circuit power during testing. In order to validate this assumption, we performed local search around the local solution produced by DP-fill based on the *peak circuit power* during testing. The *greedy* and *simulated annealing* based strategies are used to perform the local search. After performing this local search pruning for reducing *peak circuit power* during testing, we have observed that the savings is marginal. This helps us to understand that the solution produced by DP-fill not only optimizes peak input toggles but also automatically generates a good solution for minimizing *peak circuit power* during testing.

CHAPTER 6

Conclusions

It is well known that at-speed testing of delay faults and transition faults is necessary to catch small delay defects in modern nanometer CMOS technologies. However, in the presence of path delays that are comparable to the clock interval, delayed signal transitions or timing hazards influence the detection of defects. Due to the these variations in signalling delays, it is important to perform at-speed testing even for stuck faults, to reduce the test escapes (McCluskey and Tseng, 2000; Vorisek *et al.*, 2004). So, at-speed stuck-at testing is necessary in the nanometer CMOS regime. Since power dissipation increases proportionately with the clock speed, the power grid experiences higher *IR-drop*, that is not observed during slow speed testing. This excessive *IR-drop* on power grid, increases the delay of gates on the circuit, and leads to the following issues

1. a good chip is categorized as defective, which is the problem of *false negatives*, that impacts the yield of a product and a loss to the manufacturer; and more importantly
2. a defective chip is categorized as good, which is the problem of *false positives*, that impacts the trust of the customers on the manufacturer, and ultimately a financial loss to the manufacturer.

Keeping this in mind, under CSP-scan architecture, we proposed efficient algorithms for test vector ordering and don't care filling for peak power minimization during at-speed stuck-at testing. The major conclusions based on the work done as part of this thesis are as follows:

6.1 Test vector ordering for fully specified test sets

In Chapter 3, we had shown that given a fully specified test set, the problem of optimal test vector ordering for peak power minimization, under the CSP-scan architecture, maps to the bottleneck traveling salesman problem(BTSP), which is NP-hard. We have

used an efficient BTSP heuristic to solve the same. This heuristic is experimented on all the 21 ITC circuits and interestingly, the solution obtained in each of the benchmark circuits is *globally optimal*. Although, the used BTSP heuristic have *globally optimal* on all the benchmark circuits, the optimality for any given circuit, is not guaranteed, as the underlying problem is NP-hard. This only suggests that the heuristic is very effective in solving these instances of complete graphs, for peak power minimization during testing. This is a very interesting case study, where an NP-hard problem, can be solved very efficiently using an intelligent heuristic.

6.2 Simultaneous test vector ordering and don't care filling

In practice, the test sets generated by commercial automatic test vector generation (ATPG) tools like Mentor's $Fastscan^{TM}$ or Synopsys $Tetramax^{TM}$, are dominated by don't cares, for large circuits. This makes don't care filling very important for minimizing test power. If these don't cares are filled using random-fill, 0-fill, 1-fill or MT-fill, then the test vector ordering using the BTSP heuristic gives a very efficient solution. However, the overall problem of simultaneous test vector ordering and don't care filling, may not be best solved this way. In fact, in this thesis, we show that this leads to a sub-optimal solution.

Keeping this in mind, in this thesis, we focus on the problem of simultaneous test vector ordering and don't care filling. As we have already discussed, the problem of test vector ordering is by itself NP-hard. Thus, including don't care filling as part of the optimization engine, increases the hardness of the peak power minimization engine. Keeping this in mind, in chapter 4 we proposed an efficient heuristic $(XStat)$ for test vector ordering and don't care filling in an integrated fashion, that produces solutions which reduce peak test power significantly, while taking very little time in arriving at the solutions.

6.3 An optimal algorithm for peak input switching activity

While $XStat$ algorithm is an efficient heuristic for reducing peak input switching activity, thereby reducing peak circuit switching activity, it does not guarantee optimality. In chapter 5, to address this issue we had shown that given a test vector order, don't cares can be filled in an optimal way using *dynamic programming* so as to minimize peak input switching activity.

6.4 Future Work

Input switching activity correlates well with circuit switching activity (Girard *et al.*, 1998), and is less compute intensive than circuit switching activity, we have designed algorithms for minimization of peak input switching activity as a means to minimize peak power dissipation during at-speed testing. It is an interesting future work, to propose efficient and scalable algorithms to minimize peak circuit switching activity.

Although this thesis suggests an optimal algorithm for don't care filling, for a given test cube ordering, the global problem of minimizing input switching activity also, is not solved optimally. Optimal peak input switching activity problem by simultaneous test vector ordering and don't care filling is also an interesting open problem. Additionally, extending these algorithms for reducing peak power dissipation during testing of 3D-ICS, is another interesting future work.

REFERENCES

1. **Abramovici, M.**, **M. Breur**, and **A. D. Friedman**. *In Digital Systems Testing and Testable Design*. Wiley-Blackwell, 1994.

2. **Abramovici, M.**, **P. Menon**, and **D. Miller**, Critical path tracing - an alternative to fault simulation. *In Design Automation Conference*. IEEE, 1983.

3. **Ahmed, N.**, **M. Tehranipoor**, and **V. Jayaram**, Timing-based delay test for screening small delay defects. *In Proceedings of the 43rd Design Automation Conference, DAC 2006, San Francisco, CA, USA, July 24-28, 2006*. 2006*a*. URL `http://doi.acm.org/10.1145/1146909.1146993`.

4. **Ahmed, N.**, **M. Tehranipoor**, and **V. Jayaram**, Timing-based delay test for screening small delay defects. *In Proceedings of the 43rd Annual Design Automation Conference*, DAC '06. ACM, New York, NY, USA, 2006*b*. ISBN 1-59593-381-6. URL `http://doi.acm.org/10.1145/1146909.1146993`.

5. **Almukhaizim, S.** and **O. Sinanoglu**, Peak power reduction through dynamic partitioning of scan chains. *In Test Conference, 2008. ITC 2008. IEEE International*. 2008. ISSN 1089-3539.

6. **Balatsouka, S.**, **V. Tenentes**, **X. Kavousianos**, and **K. Chakrabarty**, Defect aware X-filling for low-power scan testing. *In 2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*. 2010. ISSN 1530-1591.

7. **Bao, F.**, **K. Peng**, **M. Tehranipoor**, and **K. Chakrabarty** (2013*a*). Generation of effective 1-detect TDF patterns for detecting small-delay defects. *IEEE Trans. on CAD of Integrated Circuits and Systems*, **32**(10), 1583–1594. URL `http://dx.doi.org/10.1109/TCAD.2013.2266374`.

8. **Bao, F.**, **K. Peng**, **M. Yilmaz**, **K. Chakrabarty**, **L. Winemberg**, and **M. Tehranipoor** (2013*b*). Efficient pattern generation for small-delay defects using selection of critical faults. *J. Electronic Testing*, **29**(1), 35–48. URL `http://dx.doi.org/10.1007/s10836-012-5345-9`.

9. **Bhattacharya, B. B.**, Double-tree scan: A novel low-power scan-path architecture. *In International Test Conference*. IEEE, 2003.

10. **Bhunia, S.**, **H. Mahmoodi**, **D. Ghosh**, **S. Mukhopadhyay**, and **K. Roy** (2005*a*). Low-power scan design using first-level supply gating. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, **13**(3), 384–395. ISSN 1063-8210.

11. **Bhunia, S.**, **H. Mahmoodi**, **A. Raychowdhury**, and **K. Roy**, A novel low-overhead delay testing technique for arbitrary two-pattern test application. *In Design Automation and Test in Europe*. IEEE, 2005*b*.

12. **Bonhomme, Y.**, **P. Girard**, **L. Guiller**, **C. Landrault**, **S. Pravossoudovitch**, and **A. Virazel**, Design of routing-constrained low power scan chains. *In Design, Automation and Test in Europe*. IEEE, 2004.

13. **Bonhomme, Y.**, **P. Girard**, **C. Landrault**, and **S. Pravossoudovitch**, Power driven chaining of flip-flops in scan architectures. *In International Test Conference*. IEEE, 2002.

14. **Borkar, S.** (1999). Design challenges of technology scaling. *IEEE Micro*, **19**(4), 23–29.

15. **Bosio, A.**, **P. Girard**, **S. Pravossoudovitch**, **P. Bernardi**, and **M. Reorda**, An exact and efficient critical path tracing algorithm. *In International Symposium on Electronic Design, Test and Application*. IEEE, 2010.

16. **Chakraborty, T.** and **V. Agrawal**, Robust testing for stuck-at faults. *In VLSI Design, 1995., Proceedings of the 8th International Conference on*. 1995*a*. ISSN 1063-9667.

17. **Chakraborty, T.** and **V. Agrawal**, Simulation of at-speed tests for stuck-at faults. *In VLSI Test Symposium, 1995. Proceedings., 13th IEEE*. 1995*b*. ISSN 1093-0167.

18. **Dabholkar, V.** and **S. Chakravarty** (1994). Minimizing power dissipation in combinational circuits during test application. *State University of New York, Buffalo*.

19. **Dabholkar, V.**, **S. Chakravarty**, **I. Pomeranz**, and **S. Reddy** (1998). Techniques for minimizing power dissipation in scan and combinational circuits during test application. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **17**(12), 1325–1333. ISSN 0278-0070.

20. **Datta, R.**, **R. Gupta**, **A. Sebastine**, **J. A. Abraham**, and **M. d'Abreu** (2004). Tri-scan: A novel DFT technique for cmos path delay fault testing. *IEEE International Test Conference (ITC)*, **0**, 1118–1127. ISSN 1089-3539.

21. **Dennard, R.**, **F. Gaensslen**, **V. Rideout**, **E. Bassous**, and **A. LeBlanc** (1974). Design of ion-implanted MOSFET's with very small physical dimensions. *IEEE Journal of Solid-State Circuits*, **9**(5), 256–268.

22. **Dervisoglu, B.** and **G. Stong**, Design for testability: Using scanpath techniques for path-delay test and measurement. *In International Test Conference*. IEEE, 1991.

23. **Devanathan, V.**, **C. Ravikumar**, **R. Mehrotra**, and **V. Kamakoti**, PMScan : A power-managed scan for simultaneous reduction of dynamic and leakage power during scan test. *In International Test Conference*. IEEE, 2007*a*.

24. **Devanathan, V. R.**, **C. P. Ravikumar**, and **V. Kamakoti**, Glitch-aware pattern generation and optimization framework for power-safe scan test. *In VLSI Test Symposium*. IEEE, 2007*b*.

25. **Devanathan, V. R.**, **C. P. Ravikumar**, and **V. Kamakoti**, A stochastic pattern generation and optimization framework for variation-tolerant, power-safe scan test. *In International Test Conference*. IEEE, 2007*c*.

26. **Doroshko, N.** and **V. Sarvanov** (1981). The minimax traveling salesman problem and hamiltonian cycles in powers of graphs. vestsi akad. navuk bssr, ser. fiz. *Mat. Navuk*, **143**, 119–120.

27. **Eichelberger, E. B.** (1974). Level sensitive logic system. US Patent 3,783,254.

28. **Eichelberger, E. B.** and **T. W. Williams**, A logic design structure for LSI testability. *In Design Automation Conference*. IEEE, 1977.

29. **Funatsu, S.**, **N. Wakatsuki**, and **T. Arima**, Test generation systems in Japan. *In Design Automation Conference*. IEEE Press, 1975.

30. **Ganesan, S.** and **S. P. Khatri**, A modified scan-d flip-flop design to reduce test power. *In International Test Synthesis Workshop*. IEEE, 2008.

31. **Garey, M. R.** and **D. S. Johnson**, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. ISBN 0716710455.

32. **Gerstendorfer, S.** and **H. J. Wunderlich**, Minimized power consumption for scan-based BIST. *In International Test Conference*. IEEE, 1999.

33. **Girard, P.** (2002). Survey of low-power testing of VLSI circuits. *IEEE Design and Test of Computers*, **19**(3), 80–90.

34. **Girard, P.**, **L. Guiller**, **C. Landrault**, and **S. Pravossoudovitch**, Circuit partitioning for low power bist design with minimized peak power consumption. *In Test Symposium, 1999. (ATS '99) Proceedings. Eighth Asian*. 1999. ISSN 1081-7735.

35. **Girard, P.**, **C. Landrault**, **S. Pravossoudovitch**, and **D. Severac**, Reducing power consumption during test application by test vector ordering. *In International Symposium on Circuits and Systems*. IEEE, 1998.

36. **Glover, C. T.** and **M. R. Mercer**, A method of delay fault test generation. *In Design Automation Conference*. IEEE, 1988.

37. **Goel, S. K.**, **K. Chakrabarty**, **M. Yilmaz**, **K. Peng**, and **M. Tehranipoor**, Circuit topology-based test pattern generation for small-delay defects. *In Proceedings of the 19th IEEE Asian Test Symposium, ATS 2010, 1-4 December 2010, Shanghai, China*. 2010. URL `http://dx.doi.org/10.1109/ATS.2010.59`.

38. **Huang, W.** (2007). *HotSpot - A Chip and Package Compact Thermal Modeling Methodology for VLSI Design*. Ph.D. thesis, University of Virginia.

39. **Kavousianos, X.**, **D. Bakalis**, **M. Bellos**, and **D. Nikolos**, An efficient test vector ordering method for low power testing. *In International Symposium on VLSI*. IEEE, 2004.

40. **Kumar, S. K.**, **S. Kaundiya**, and **S. Chattopadhyay**, Customizing pattern set for test power reduction via improved x-identification and reordering. *In International Symposium on Low Power Electronics and Design*. IEEE, 2010.

41. **Kundu, S.** and **S. Chattopadhyay**, Efficient don't care filling for power reduction during testing. *In Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference on*. 2009.

42. **Kurian, G.**, **V. V. N. Rao**, **V. Patidhar**, and **V. Kamakoti** (2009). Test power reduction using integrated scan cell and test vector reordering techniques on linear scan and double tree scan architectures. *ASP Journal of Low Power Electronics*, **5**(1), 58–68.

43. **Larusic, J.**, **A. P. Punnen**, and **E. Aubanel** (2012). Experimental analysis of heuristics for the bottleneck traveling salesman problem. *Journal of Heuristics*, **18**(3), 473–503.

44. **Lawler, L. J. R. K. A. S. D., E.L.**, *Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization*. Wiley, Chichester, 1985. ISBN 978-0-471-90413-7.

45. **Lee, K.-J.**, **T.-C. Haung**, and **J.-J. Chen**, Peak-power reduction for multiple-scan circuits during test application. *In Test Symposium, 2000. (ATS 2000). Proceedings of the Ninth Asian*. 2000. ISSN 1081-7735.

46. **Li, J.**, **Q. Xu**, **Y. Hu**, and **X. Li**, ifill: An impact-oriented X-filling method for shift- and capture-power reduction in at-speed scan-based testing. *In 2008 Design, Automation and Test in Europe*. 2008. ISSN 1530-1591.

47. **Li, J.**, **Q. Xu**, **Y. Hu**, and **X. Li** (2010). X-filling for simultaneous shift- and capture-power reduction in at-speed scan-based testing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, **18**(7), 1081–1092. ISSN 1063-8210.

48. **Liaw, C.-C.**, **S. Su**, and **Y. K. Malaiya**, Test generation for delay faults using stuck-at-fault test set. *In International Test Conference*. IEEE, 1980.

49. **Lin, X.** and **J. Rajski**, Test power reduction by blocking scan cell outputs. *In Asian Test Symposium*. IEEE, 2008.

50. **Lin, X.**, **J. Rajski**, **I. Pomeranz**, and **S. M. Reddy**, On static test compaction and test pattern ordering for scan designs. *In International Test Conference*. IEEE, Baltimore, MD, USA, 2001.

51. **Liu, X.** (2004). *ATPG and DFT algorithms for delay fault testing*. Ph.D. thesis, Virginia Polytechnic Institute & State University.

52. **Magen, N.**, **A. Kolodny**, **U. Weiser**, and **N. Shamir**, Interconnect-power dissipation in a microprocessor. *In Proceedings of the 2004 International Workshop on System Level Interconnect Prediction*, SLIP ’04. ACM, New York, NY, USA, 2004. URL `http://doi.acm.org/10.1145/966747.966750`.

53. **Malaiya, Y. K.** and **R. Narayanaswamy**, Testing for timing faults in synchronous sequential integrated circuits. *In International Test Conference*. IEEE, 1983.

54. **Manku, G. S.** (1996). A linear time algorithm for the bottleneck biconnected spanning subgraph problem. *Information Processing Letters*, **59**(1), 1–7.

55. **Maxwell, P.**, **R. Aitken**, **K. Kollitz**, and **A. Brown**, IDDQ and AC scan: the war against unmodelled defects. *In International Test Conference*. IEEE, 1996.

56. **McCluskey, E. J.** and **C.-W. Tseng**, Stuck-fault tests vs. actual defects. *In IEEE International Test Conference*. IEEE, 2000.

57. **Mishra, A.**, **N. Sinha**, **Satdev**, **V. Singh**, **S. Chakravarty**, and **A. Singh**, Modified scan flip-flop for low power testing. *In Asian Test Symposium*. IEEE, 2010.

58. **Miyase, K.** and **S. Kajihara** (2006). Xid: Don't care identification of test patterns for combinational circuits. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, **23**(2), 321–326. ISSN 0278-0070. URL `http://dx.doi.org/10.1109/TCAD.2003.822103`.

59. **Miyase, K.**, **Y. Uchinodan**, **K. Enokimoto**, **Y. Yamato**, **X. Wen**, **S. Kajihara**, **F. Wu**, **L. Dilillo**, **A. Bosio**, **P. Girard**, and **A. Virazel**, Effective launch-to-capture power reduction for los scheme with adjacent-probability-based x-filling. *In 2011 Asian Test Symposium*. 2011. ISSN 1081-7735.

60. **Pant, P.**, **J. Zelman**, **G. Colon-Bonet**, **J. Flint**, and **S. Yurash**, Lessons from at-speed scan deployment on an Intel Itanium microprocessor. *In International Test Conference*. IEEE, 2010.

61. **Pant, S.** (2008). *Design and Analysis of Power Distribution Networks in VLSI Circuits*. Ph.D. thesis, University of Michigan.

62. **Parimi, N.** and **X. Sun**, Toggle-masking for test-per-scan VLSI circuits. *In International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*. IEEE, 2004.

63. **Parker, R. G.** and **R. L. Rardin** (1984). Guaranteed performance heuristics for the bottleneck travelling salesman problem. *Operations Research Letters*, **2**(6), 269–272.

64. **Peng, K.**, **M. Yilmaz**, **K. Chakrabarty**, and **M. Tehranipoor** (2013). Crosstalk- and process variations-aware high-quality tests for small-delay defects. *IEEE Trans. VLSI Syst.*, **21**(6), 1129–1142. URL `http://dx.doi.org/10.1109/TVLSI.2012.2205026`.

65. **Peng, K.**, **M. Yilmaz**, **M. Tehranipoor**, and **K. Chakrabarty**, High-quality pattern selection for screening small-delay defects considering process variations and crosstalk. *In Design, Automation and Test in Europe, DATE 2010, Dresden, Germany, March 8-12, 2010*. 2010. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5457036`.

66. **Potluri, S.** (2015). *Power: Its Manifestations in Digital Systems Testing*. Ph.D. thesis, Indian Institute of Technology Madras.

67. **Potluri, S.**, **N. Chandrachoodan**, and **V. Kamakoti** (2012). Interconnect aware test power reduction. *ASP Journal of Low Power Electronics*, **8**(4), 516–525.

68. **Potluri, S.**, **A. Satya Trinadh**, **C. Sobhan Babu**, **V. Kamakoti**, and **N. Chandrachoodan** (2015). DFT assisted techniques for peak launch-to-capture power reduction during launch-on-shift at-speed testing. *ACM Transactions on Design Automation of Electronic Systems*.

69. **Prabhu, M.** and **J. Abraham**, Functional test generation for hard to detect stuck-at faults using rtl model checking. *In Test Symposium (ETS), 2012 17th IEEE European*. 2012.

70. **Punnen, A. P.** and **K. Nair** (1994). A fast and simple algorithm for the bottleneck biconnected spanning subgraph problem. *Information Processing Letters*, **50**(5), 283 – 286. ISSN 0020-0190. URL `http://www.sciencedirect.com/science/article/pii/0020019094000417`.

71. **Qiu, X.**, **Y. Ma**, **X. He**, and **X. Hong**, IPOSA: A novel slack distribution algorithm for interconnect power optimization. *In International Symposium on Quality Electronic Design*. IEEE, 2008.

72. **Ramakrishnan, R.**, **P. Sharma**, and **A. P. Punnen** (2009). An efficient heuristic algorithm for the bottleneck traveling salesman problem. *Opsearch*, **46**(3), 275–288.

73. **Reddy, L. N.**, **I. Pomeranz**, and **S. M. Reddy**, Rotco:a reversed order test compaction technique. *In EURO-ASIC*. IEEE, Paris, France, 1992.

74. **Remersaro, S.**, **X. Lin**, **Z. Zhang**, **S. M. Reddy**, **I. Pomeranz**, and **J. Rajski**, Preferred fill: A scalable method to reduce capture power for scan based designs. *In 2006 IEEE International Test Conference*. 2006. ISSN 1089-3539.

75. **Sankaralingam, K.**, **R. Oruganti**, and **N. Touba**, Static compaction techniques to control scan vector power dissipation. *In VLSI Test Symposium, 2000. Proceedings. 18th IEEE*. 2000. ISSN 1093-0167.

76. **Sankaralingam, K.** and **N. Touba**, Controlling peak power during scan testing. *In VLSI Test Symposium, 2002. (VTS 2002). Proceedings 20th IEEE*. 2002.

77. **Sarvanov, V.**, Traveller minimax problem in plane: Complexity of approximate solution. *In DOKLADY AKADEMII NAUK BELARUSI*, volume 39. ACADEMII NAUK BELARUSI F SCORINA PR 66, ROOM 403, MINSK, BYELARUS 220072, 1995.

78. **Savir, J.** and **W. H. McAnney** (1988). Random pattern testability of delay faults. *IEEE Transactions on Computers*, **37**(3), 291–300.

79. **Saxena, J.**, **K. Butler**, and **L. Whetsel**, An analysis of power reduction techniques in scan testing. *In International Test Conference*. IEEE, 2001.

80. **Saxena, J.**, **K. M. Butler**, **V. B. Jayaram**, **S. Kundu**, **N. V. Arvind**, **P. Sreeprakash**, and **M. Hachinger**, A Case Study of IR-Drop in Structured At-Speed Testing. *In International Test Conference*. IEEE, 2003.

81. **Schulz, M.**, **E. Trischler**, and **T. M. Sarfert** (1988). Socrates: a highly efficient automatic test pattern generation system. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, **7**(1), 126–137.

82. **Singh, A.**, **J. Plusquellic**, **D. Phatak**, and **C. Patel** (2006). Defect simulation methodology for IDDT testing. *Journal of Electronic Testing*, **22**(3), 255–272.

83. **Sparso, J.** and **S. Furber**, *Principles of Asynchronous Circuit Design, A Systems Perspective*. Kluwer Publishers, 2001.

84. **Takahashi, N.**, **N. Ishiura**, and **S. Yajima** (2006). Fault simulation for multiple faults by boolean function manipulation. *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, **13**(4), 531–535.

85. **Tehranipoor, M.**, **K. Peng**, and **K. Chakrabarty**, *Test and Diagnosis for Small-Delay Defects*. Springer, 2011. ISBN 978-1-4419-8296-4. URL `http://dx.doi.org/10.1007/978-1-4419-8297-1`.

86. **Tzeng, C. W.** and **S. Y. Huang** (2009). QC-fill: Quick-and-cool X-filling for multicasting-based scan test. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **28**(11), 1756–1766. ISSN 0278-0070.

87. **Vandegriend, B.** (1998). Finding hamiltonian cycles: Algorithms. *Graphs and Performance, Master of Science Thesis, Department of Computing Science, University of Alberta, Edmonton, Alberta*.

88. **Venkataraman, S.**, **S. Sivaraj**, **E. Amyeen**, **S. Lee**, **A. Ojha**, and **R. Guo**, An experimental study of n-detect scan ATPG patterns on a processor. *In VLSI Test Symposium*. IEEE, Napa Valley, CA, USA, 2004.

89. **Vorisek, V.**, **T. Koch**, and **H. Fischer**, At-speed testing of soc ics. *In Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 3. 2004. ISSN 1530-1591.

90. **Waicukauski, J. A.**, **E. Lindbloom**, **V. S. Iyengar**, and **B. K. Rosen**, Transition fault simulation by parallel pattern single fault propagation. *In International Test Conference*. IEEE, 1986.

91. **Wen, X.**, **K. Miyase**, **T. Suzuki**, **S. Kajihara**, **Y. Ohsumi**, and **K. Saluja**, Critical-path-aware X-filling for effective IR-drop reduction in at-speed scan testing. *In Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*. 2007. ISSN 0738-100X.

92. **Williams, M.** and **J. Angell** (1973). Enhancing testability of large-scale integrated circuits via test points and additional logic. *IEEE Transactions on Computers*, **C-22**(1), 46–60.

93. **Wu, F.**, **L. Dilillo**, **A. Bosio**, **P. Girard**, **S. Pravossoudovitch**, **A. Virazel**, **M. Tehranipoor**, **K. Miyase**, **X. Wen**, and **N. Ahmed**, Power reduction through X-filling of transition fault test vectors for LOS testing. *In International Conference on Design Technology of Integrated Systems in Nanoscale Era*. IEEE, 2011.

94. **Wu, M.-F.**, **K.-S. Hu**, and **J.-L. Huang** (2009). Lptest: a flexible low-power test pattern generator. *Journal of Electronic Testing*, **25**(6), 323. ISSN 1573-0727. URL `http://dx.doi.org/10.1007/s10836-009-5115-5`.

95. **Xu, G.** and **A. Singh**, Delay test scan flip-flop: DFT for high coverage delay testing. *In International Conference on VLSI Design*. IEEE, 2007.

96. **Yao, C.**, **K. Saluja**, and **P. Ramanathan** (2011). Power and thermal constrained test scheduling under deep submicron technologies. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, **30**(2), 317–322. ISSN 0278-0070.

97. **Yilmaz, M.**, **K. Chakrabarty**, and **M. Tehranipoor**, Interconnect-aware and layout-oriented test-pattern selection for small-delay defects. *In 2008 IEEE International Test Conference, ITC 2008, Santa Clara, California, USA, October 26-31, 2008*. 2008*a*. URL `http://dx.doi.org/10.1109/TEST.2008.4700627`.

98. **Yilmaz, M.**, **K. Chakrabarty**, and **M. Tehranipoor**, Test-pattern grading and pattern selection for small-delay defects. *In 26th IEEE VLSI Test Symposium (VTS 2008), April 27 - May 1, 2008, San Diego, California, USA*. 2008*b*. URL `http://dx.doi.org/10.1109/VTS.2008.32`.

99. **Yilmaz, M.**, **K. Chakrabarty**, and **M. Tehranipoor** (2010). Test-pattern selection for screening small-delay defects in very-deep submicrometer integrated circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, **29**(5), 760–773. URL `http://dx.doi.org/10.1109/TCAD.2010.2043591`.

100. **Yilmaz, M.**, **M. Tehranipoor**, and **K. Chakrabarty** (2011). A metric to target small-delay defects in industrial circuits. *IEEE Design & Test of Computers*, **28**(2), 52–61. URL `http://doi.ieeecomputersociety.org/10.1109/MDT.2011.26`.

List of publications based on the research work

Publications based on thesis

1. A. Satya Trinadh, S. Potluri, Ch. Sobhan Babu, S. G. Singh and V. Kamakoti, “Optimal Don’t Care Filling for Minimizing Peak Toggles during At-speed Stuck-at Testing”, *ACM Transactions on Design Automation of Electronic Systems*. Vol. 23,No. 1,2017, pp. 5:1-5:26.

2. A. Satya Trinadh, Ch. Sobhan Babu, S. G. Singh, S. Potluri and V. Kamakoti, “DP-fill: A Dynamic Programming approach to X-filling for minimizing peak test power in scan tests”, *Design Automation and Test in Europe*, IEEE, 2015 (Grenoble, France).

3. A. Satya Trinadh, S. Potluri, S. Balachandran, Ch. Sobhan Babu and V. Kamakoti, “XStat: Statistical X-Filling Algorithm for Peak Capture Power Reduction in Scan Tests”, *Journal of Low Power Electronics*, Vol. 10, No. 1, 2014, pp. 107-115.

4. A. Satya Trinadh, S. Potluri, Ch. Sobhan Babu and V. Kamakoti, “An Efficient Heuristic for Peak Capture Power Minimization During Scan-Based Test”, *Journal of Low Power Electronics*, Vol. 9, No. 2, 2013, pp. 264-274.

Other related publications

1. S. Potluri, A. Satya Trinadh, C. Rajamanikkam and S. Balachandran. ”LPScan: An Algorithm for Supply Scaling and Switching Activity Minimization during Test”, *International Conference on Computer Design*, IEEE, 2013, pp. 463-466 (Asheville, USA).

2. S. Potluri, A. Satya Trinadh, R. Baskaran, N. Chandrachoodan and V. Kamakoti. ”PinPoint: An Algorithm for Enhancing Diagnostic Resolution using Capture Cycle Power Information”, *European Test Symposium*, IEEE, 2013 (Avignon, France).

3. S. Potluri, A. Satya Trinadh, Ch. Sobhan Babu, V. Kamakoti and N. Chandrachoodan. “DFT Assisted Techniques for Peak Launch-to-Capture Power Reduction during Launch-On-Shift At-Speed Testing”, *ACM Transactions on Design Automation of Electronic Systems*,21(1): 14,2015.

4. S. Potluri, A. Satya Trinadh, Siddhant Saraf and V. Kamakoti. “Component fault localization using switching current measurements”, *European Test Symposium*, IEEE, 2016. pp.1-2.