Integrated Enhancement of Testability and Diagnosability for Digital Circuits

Nikhil P. Rahagude

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Dr. Michael S. Hsiao, Chair
Dr. Dong S. Ha
Dr. Patrick R. Schaumont

November 02, 2010
Blacksburg, Virginia

Keywords: Fault Coverage, Diagnostic Resolution, Weighted Average, Test Point Insertion,
Design for Testability, Design for Diagnosability, Built-in Self Test

Integrated Enhancement of Testability and Diagnosability for Digital Circuits

Nikhil P. Rahagude

(ABSTRACT)

While conventional test point insertions commonly used in design for testability can improve *fault coverage*, the test points selected may not necessarily be the best candidates to aid *silicon diagnosis*. In this thesis, test point insertions are conducted with the aim to detect more faults and also synergistically distinguish currently indistinguishable fault-pairs. We achieve this by identifying those points in the circuit, which are not only hard-to-test but also lie on distinguishable frontiers, as Testability-Diagnosability (*TD*) points. To this end, we propose a novel low-cost metric to identify such *TD* points. Further, we propose a new DFT + DFD architecture, which adds just one pin (to identify test/functional mode) and small additional combinational logic to the circuit under test. Our experiments indicate that the proposed architecture can distinguish 4× more previously indistinguishable fault-pairs than existing DFT architectures while maintaining similar fault coverages. Further, the experiments illustrate that quality results can be achieved with an area overhead of around 5%. Additional experiments conducted on hard-to-test circuits show an increase in *fault coverage* by 48% while maintaining similar *diagnostic resolution*.

Built-in Self Test (BIST) is a technique of adding additional blocks of hardware to the circuits to allow them to perform self-testing. This enables the circuits to test themselves thereby reducing the dependency on the expensive external automated test equipment (ATE). At the end of a test session, BIST generates a signature which is a compaction of the obtained output responses of the circuit for that session. Comparison of this signature with the reference signature categorizes the circuit as error free or buggy. While BIST provides a quick and low cost alternative to check circuit's correctness, diagnosis in BIST environment remains poor because of the limited information present in the lossily compacted final signature. The signature does not give any information about the possible defect location in the cir-

cuit. To facilitate diagnosis, researchers have proposed the use of two additional on-chip embedded memories, *response memory* to store reference responses and *fail memory* to store failing responses. We propose a novel architecture in which only one additional memory is required. Experimental results conducted on benchmark circuits substantiate that the same *fault coverage* can be maintained using just 5% of the available test vectors. This reduces the size of memory required to store responses which in turn reduces area overhead. Further, by adding test points to the circuit using our proposed architecture, we can improve the *diagnostic resolution* by 60% with respect to external testing.

Dedication

To my family

Acknowledgments

There are a lot of people who have supported me directly or indirectly throughout the completion of my thesis. I would start with expressing my sincere gratitude to my advisor Dr. Michael S. Hsiao, for his kind support and constant inspiration throughout my graduate program. He not only helped me develop my thought process through research but also guided me in improving my professional values. I would also like to thank Dr. Dong S. Ha and Dr. Patrick R. Schaumont for serving on my thesis committee. A special thanks to Dr. Charles W. Bostian for giving me an opportunity to work on his project.

I can never forget the eventful life that I had with the PROACTIVE members - Mahesh, Mainak, Min, Swapneel, Sandesh, Neha, Saparya, Wei, Sarvesh, Supratik, Dhrumeel, Huy and Chinmay. I would also thank the CWT Lab members - Andrew, Sujit, Qinqin, Al, Aravind, Jeannette - for working with me.

I will always cherish the moments - support, discussions, funny quotients, gettogethers, trips - that I was lucky enough to share with my friends in Blacksburg. They made my entire stay at Virginia Tech a memorable one.

Last but not the least, I would thank my father, Prakash, my mother, Neela and both my sisters, Anushree and Sonali for being there for me and supporting me throughout my life.

Finally, I am grateful to God for showering His blessings on me.

Nikhil P. Rahagude

November 02, 2010

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

With advances in technology, digital circuits continue to follow Moore's law and contain tens to hundreds of millions of transistors packed on a single wafer that run in the gigahertz frequency (GHz) range. As minimum feature size is shrinking continuously, precise control of the fabrication process on silicon is becoming more challenging and thus circuits are more susceptible to manufacturing defects. Wide use of such circuits in day-to-day life demands error free operation, which makes it necessary to identify as much defects as possible within the testing stage itself. To cope up with the technological advancement, improvement in testing methodologies is inevitable, otherwise the cost of test would eventually surpass the cost of silicon manufacturing, as shown in Figure 1.1.

For the past several decades, Manufacturing Test has been a prominent methodology to screen and identify defective Integrated Circuits (ICs). With shrinking sizes and increasing design complexity, defects have become harder to detect and diagnose, resulting in more test escapes. For defective chips that are captured during manufacturing testing as well as field-returned test escapes, a process for locating the defects is needed - a process often referred to as *silicon diagnosis* [Marinissen]. The diagnosis process usually returns a set of possible defect locations (candidates) in the defective chip. Subsequently, physical failure analysis - an extremely time consuming process - is performed on the failed chips with the aid of these

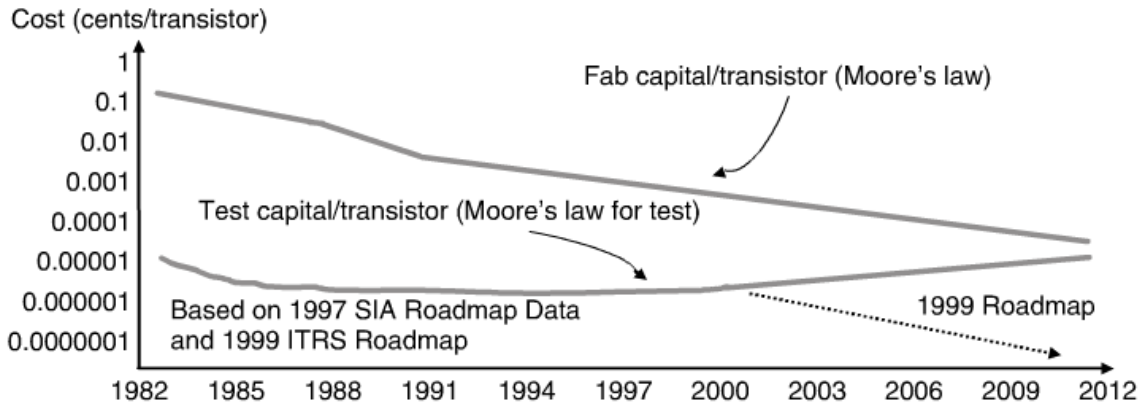


Figure 1.1: Fabrication Capital *versus* Test Capital [Wang SOC]

candidates. For efficient physical analysis, the cardinality of the candidate set should be as small as possible [Wang VLSI]. With the ever-increasing transistor count, the numbers of hard-to-detect faults and hard-to-distinguish fault-pairs also increase. To this end, test patterns that can differentiate between different faults (candidates) can play a critical role in both increasing the *diagnostic resolution* and reducing diagnosis time.

In the past, researchers have addressed the issue of handling such faults and fault-pairs in two complementing ways: The first is to generate high-quality ATPG (Automatic Test Pattern Generation) vectors as well as high-quality DTPG (Diagnostic Test Pattern Generation) vectors to achieve coverage and diagnostic goals [Chandrasekar]. The second is to enhance the testability of the CUT (Circuit Under Test) by inserting additional logic into it. Such techniques are referred to as DFT (Design For Testability) [Hayes Detection] and usually aid in reducing the effort to detect those hard-to-detect faults. The DFT inserted may also enhance diagnosability slightly, but often the enhancement is not significant, as will be shown in the results. Test point insertion is also done to improve the diagnosability of CUT, referred to as DFD (Design for Diagnosability).

In this thesis, we simultaneously enhance the testability and diagnosability of the CUT by inserting testability and diagnosability *(TD)* points.

Contributions of this thesis:

1. We propose a novel low-cost heuristic to obtain a set of *TD* points that synergistically improves both the *fault coverage* and the number of distinguishable fault-pairs (*diagnostic resolution*) in the CUT for a given test set.

2. We also propose a new method to achieve the above objective without any knowledge of a test set, utilizing the structural characteristics of the circuit.

3. We propose a novel design-for-testability and design-for-diagnosability (DFT + DFD) architecture based on the *TD* points selected. To the best of our knowledge, this is the first work simultaneously targeting the enhancement of both the testability and the diagnosability of the CUT.

4. We also developed a method to enhance the diagnosability of BIST architectures by selecting a subset of test vectors that achieve the same *fault coverage* as well as same *diagnostic resolution* as achievable by the original test set T.

Thesis Outline:
The remainder of the thesis is outlined as below:

- Chapter 2: This chapter discusses the concepts, definitions and related previous works required for our approach.

- Chapter 3: This chapter explains our low-cost heuristic for *TD* point selection for a given test set T. It also explains our DFT + DFD architecture in detail.

- Chapter 4: This chapter proposes novel heuristics to select *TD* points using structural characteristics of the circuit, when no test set is given.

- Chapter 5: This chapter deals with new architecture developed to improve diagnosability of BIST architectures.

- Chapter 6: This chapter concludes the thesis.

Chapter 2

Background

This chapter introduces the preliminaries, related definitions used in context of our approach. It illustrates the terms Testability Analysis (TA), Design for Test (DFT) and Design for Diagnosis (DFD) in detail and summarizes the related previous works done in this area.

2.1 Related Work

The idea of modifying the design to enhance testability and to minimize test set size was initially proposed in [Reddy]. The authors showed that their network realizations require only few fault detection tests. Next, in [Hayes Diagnosability], Hayes showed that network realization of any given combinational or sequential designs required only five tests. However, such a technique relied on inserting an EXCLUSIVE-OR circuit at the inputs of each gate in the given design. This is impractical given the size of the modern-day designs. In [Hayes Detection], Hayes and Friedman proposed the *test point insertion* method to minimize the fault detection test set size for single stuck-at fault models. They used a labeling scheme and modeled the test point insertion problem as an Integer Programming problem, which is known to be NP-hard. Later, in [Krishnamurthy], it was shown that the optimal test point

placement in designs with reconvergent fan-out is NP-complete. They also presented a dynamic programming approach to handle test point placement problem. Since then, *test point insertions* have become a popular DFT technique. In past, *test point insertions* have been studied with respect to different objectives. For instance, *test point insertions* have been conducted to improve *fault coverage* [Nakao, Seiss, Tamarapalli, Touba, SWang, Geuzebroek 03], minimize test set size [Balakrishnan, Yoshimura, Remersaro, Geuzebroek 00, Sethuram], to realize scan paths through functional logic in order to alleviate scan-based DFT overhead [Lin], and to avoid peak power violation in capture cycles [Sankaralingam]. Finally, there has also been some work on inserting test points for improving the diagnosability (increase *diagnostic resolution*) of the CUT [Ravikumar]. But as our experiments indicated, such points that focus only on diagnosability do not significantly enhance testability.

2.2 Terminologies

This section defines the key terms used to explain our technique.

2.2.1 Reconvergence

The inputs of a gate are referred to as the fanins of that gate while the outputs are referred to as the fanouts of that gate. We say that a circuit contains reconvergent gates when the fanouts of a gate meet or merge at another gate in the circuit.

2.2.2 Fault Model

It is the logical representation of a *defect*. Over the years, large number of fault models have been proposed to characterize *defects*. They are stuck-at, bridging, transition etc. The most widely used model is the stuck-at fault model.

2.2.3 Test Vector

An input pattern or sequence of input patterns that can produce different output responses for the fault-free and faulty circuit.

2.2.4 Logic Simulation

It is the process of applying a pattern to primary inputs of the circuit and evaluating the internal gates (updating the fanouts based on the values at fanins) to obtain a response at the primary outputs. The logic values used for this process are $\{0, 1, X\}$.

2.2.5 Fault Simulation

It is logic simulation performed in the presence of a fault in the circuit. So, in addition to the logic values $\{0, 1, X\}$ used in logic simulation, it also uses the logic values $\{D, \overline{D}\}$ to represent the fault effects; $D = (1/0)$ and $\overline{D} = (0/1)$ has the form (fault-free value/faulty value).

2.2.6 Testing

Testing is the process of applying a test pattern to the primary inputs of the circuit and checking whether the output response obtained at the primary outputs conforms with the expected response for that particular test pattern. This process is done for all the test patterns available. It tells us if the circuit is faulty or not.

2.2.7 T-Testable Fault

A fault is said to be T-testable by a test set T if the output response of the faulty circuit is different from that of fault-free circuit for at least one vector in the given test set T. Figure

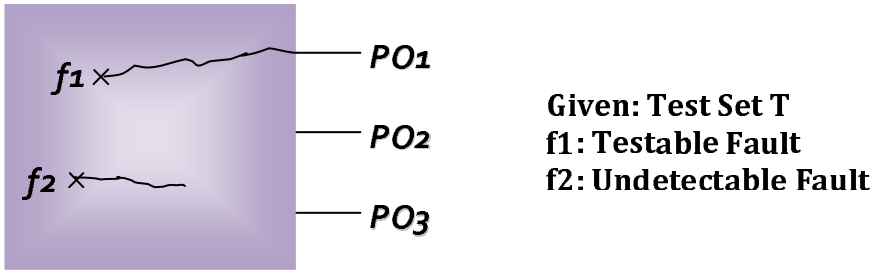


Figure 2.1: T-Testable and T-Undetectable Fault

2.1 shows a testable fault $f1$ which produces erroneous response at $PO1$ for the test set T.

2.2.8 T-Undetectable Fault

A fault is said to be T-undetectable if there is no test vector in test set T that can produce different output responses in the faulty and fault-free circuits for that fault. Figure 2.1 shows an undetectable fault $f2$ which does not produce any erroneous response at circuit outputs.

2.2.9 Fault Coverage

Fault Coverage (FC) is the ratio of number of T-Testable faults to the total number of faults in the circuit. It is the percentage of modeled faults for which the CUT gives erroneous output responses if T is applied.

$$FC = \frac{Number\ of\ T - Testable\ Faults}{Total\ Number\ of\ Faults} \tag{2.1}$$

2.2.10 Diagnosis

After the testing phase categorizes a digital circuit as faulty, the next step is to locate the defect in the digital circuit. This process is termed as *Diagnosis* [Wang SOC].

Diagnosis can be categorized as Silicon Debug and Defect Diagnosis. Silicon Debug targets the bugs that have escaped the pre silicon design process. These errors are caused by limitations or flaws in circuit models, simulations, or verification. Defect Diagnosis deals with locating the manufacturing defects, which are nothing but physical imperfections in the manufactured chip.

The process of diagnosis involves comparing the faulty output response of the circuit and mapping them to a small candidate set of faults which when present produce the same faulty response. Ideally, for efficient diagnosis, each fault present in the circuit should produce different output response.

2.2.11 T-Distinguishable Fault-Pair

A fault pair ($f1$, $f2$) is said to be T-distinguishable if the output response of the circuit in presence of $f1$ and that in the presence of $f2$ is different for at least one vector in T, where $f1$ and $f2$ are T-Testable faults. Figure 2.2 (a) and 2.2 (b) show distinguishable fault pairs for a test set T.

2.2.12 T-Indistinguishable Fault-Pair

A fault-pair ($f1$, $f2$) is said to be T-indistinguishable if the output response of the circuit in presence of $f1$ and that in the presence of $f2$ is identical for all the vectors in T, where $f1$ and $f2$ are T-Testable faults. Figure 2.2 (c) shows an indistinguishable fault pair for a given test set T.

2.2.13 Diagnostic Resolution

Diagnostic Resolution (DR) is the ratio of number of T-Distinguishable fault-pairs to the total number of fault-pairs in the circuit, where each fault is T-Testable fault. It is also

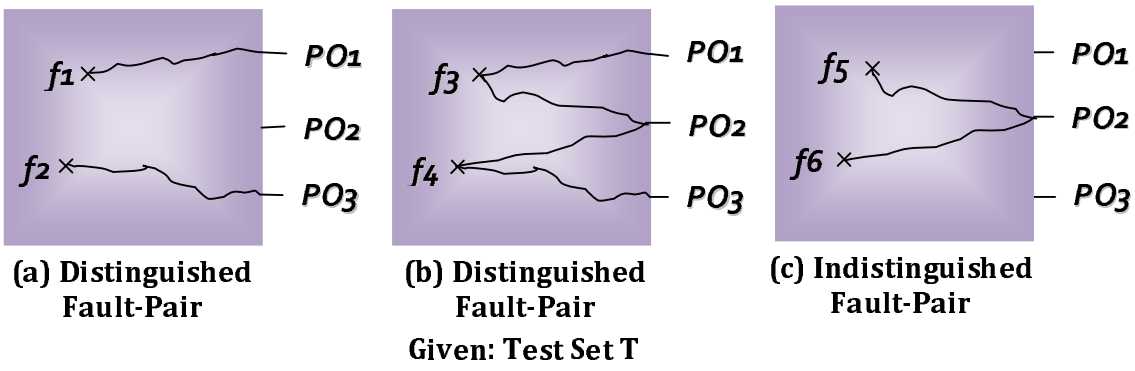


Figure 2.2: T-Distinguished and T-Indistinguished Fault Pairs

defined as the total number of defect candidates reported by a tool.

$$DR = \frac{Number\ of\ T - Distinguishable\ Fault\ Pairs}{Total\ Number\ of\ Fault\ Pairs} \tag{2.2}$$

2.3 Testability Analysis - TA

Testability analysis is a way of measuring the effort or the cost required to test a digital circuit. A manufactured digital circuit just has the primary input and primary output pins accessible to the outside world. TA tells us how difficult it is to control an internal node in the circuit to a particular value using the primary input pins of the circuit. Similarly, it also gives an idea about the difficulty in observing an internal node value using the primary output pins of the circuit. TA for any node in a circuit is represented in terms of a numerical value. Typically, the smaller the numerical value, the easier it is to control or observe the node. They are classified as follows:

2.3.1 Topology-Based TA

The most popular topology based TA is SCOAP testability.

Table 2.1: SCOAP Combinational Controllability Calculations [Wang VLSI]

Input	**0-Controllability**	**1-Controllability**
AND	min(input 0-controllabilities)+1	$\sum$(input 1-controllabilities)+1
OR	$\sum$(input 0-controllabilities)+1	min(input 1-controllabilities)+1
NOT	input 1-controllability+1	input 0-controllability+1
NAND	$\sum$(input 1-controllabilities)+1	min(input 0-controllabilities)+1
NOR	min(input 1-controllabilities)+1	$\sum$(input 0-controllabilities)+1
BUF	input 0-controllability+1	input 1-controllability+1
XOR	$\min(CC1(a) + CC1(b), CC0(a) + CC0(b)) + 1$	$\min(CC1(a) + CC0(b), CC0(a) + CC1(b)) + 1$
XNOR	$\min(CC1(a) + CC0(b), CC0(a) + CC1(b)) + 1$	$\min(CC1(a) + CC1(b), CC0(a) + CC0(b)) + 1$
Branch	Stem 0-controllability	Stem 1-controllability

SCOAP

SCOAP is an acronym for *Sandia Controllability Observability Analysis Program*, proposed by [Goldstein]. SCOAP values are defined separately for combinational and sequential circuits. For combinational circuits, it is an estimate of the number of signals in the circuit that must be changed in order to control or observe a particular node from primary inputs or primary outputs. For sequential circuits, it is an estimate of the number of clock cycles needed to control or observe a particular node from primary inputs or primary outputs. The controllability values range from 1 to ∞, while the observability values range from 0 to ∞. The higher the value, the harder it is to control a node from primary inputs or observe the node at primary outputs. The controllability values for primary inputs are set to 1, while the observability values for primary outputs are set to 0. Since, only full scan versions of sequential circuits are used in this thesis, we just show the SCOAP calculations for combinational circuits - Table 2.1 for controllability and Table 2.2 for observability.

The SCOAP values are computed by first calculating the controllability values followed by the calculation of observability values. This TA method is computationally efficient but the accuracy is compromised when it is used in a circuit containing many reconvergent fanouts.

Table 2.2: SCOAP Combinational Observability Calculations [Wang VLSI]

Primary Output	**Observability**
AND/NAND	$\sum$ (output observability, 1-controllabilities of other inputs)+1
OR/NOR	$\sum$ (output observability, 0-controllabilities of other inputs)+1
NOT/BUFFER	Output observability+1
XOR/XNOR	a: $\sum$ (output observability, $\min[CC0(b), CC1(b)]) + 1$ b: $\sum$ (output observability, $\min[CC0(a), CC1(a)]) + 1$
Stem	min[branch observabilities]

It is a static method in the sense that no test patterns are used.

2.3.2 Probability-Based TA

Probability-Based TA use signal probabilities to determine the testability of the circuit. A combinational circuit can have three probabilistic measures for each node e.g. $C0$ - probability of controlling a node to 0 from primary inputs, $C1$ - probability of controlling a node to 1 from primary inputs and O - probability of observing a node at primary output.

They have the following properties:

Table 2.3: Probability-Based Observability Calculations [Wang VLSI]

Primary Output	**Observability**
AND/NAND	$\prod$ (output observability, 1-controllabilities of other inputs)
OR/NOR	$\prod$ (output observability, 0-controllabilities of other inputs)
NOT/BUFFER	Output observability
XOR/XNOR	a: $\prod$ (output observability, max[0-controllability(b), 1-controllability(b)]) b: $\prod$ (output observability, max[0-controllability(a), 1-controllability(a)])
Stem	max[branch observabilities]

Table 2.4: Probability-Based Controllability Calculations [Wang VLSI]

	0-Controllability	**1-Controllability**
Primary Input	p_0	$p_1 = 1 - p_0$
AND	1−(output 1-controllability)	$\prod$ input 1-controllabilities
OR	$\prod$ input 0-controllabilities	1-(output 0-controllability)
NOT	input 1-controllability	input 0-controllability
NAND	$\prod$ input 1-controllabilities	1−(output 0-controllability)
NOR	1−(output 1-controllability)	$\prod$ input 0-controllabilities
BUFFER	input 0-controllability	input 1-controllability
XOR	$1 -$ (1-controllability)	$\sum(C1(a) \times C0(b), C0(a) \times C1(b))$
XNOR	$1 -$ (1-controllability)	$\sum(C0(a) \times C0(b), C1(a) \times C1(b))$
Branch	Stem 0-controllability	Stem 1-controllability

- All probabilities range between 0 and 1.
- For each node in the circuit, $C0 + C1 = 1$.
- $C0$ and $C1$ values of primary inputs are set to 0.5, while O value for primary outputs is set to 1.

The smaller the value, the harder it is to control or observe the node. Calculation of $C0$, $C1$ is shown in Table 2.4, while calculation of O is shown in Table 2.3.

The computational efficiency of this method is similar to topology-based TA. It fails to give correct values for circuits with many reconvergent gates. Since no input patterns are used, it is a static method.

2.3.3 Simulation-Based TA

These method rely on logic simulation and fault simulation to come up with a testability measure. Normally, a subset of test patterns are selected, simulation is performed and responses like the occurances of 0's, 1's, 0-to-1, 1-to-0 are collected from the signals of interest. The subset of test patterns is obtained through statistical sampling.

This TA can generate more accurate values as compared to previous two methods even for circuits with many reconvergent fanouts, but it can become computationally inefficient because of long simulation time.

2.4 Design for Testability

In early days of digital circuits, design used to be followed by testing. Design engineers would implement the functionality based on specifications with no focus on test. The built functionality would then go to a test engineer who would figure out ways to efficiently test the circuit to find whether it is defect free or not. But as industry moved into the VLSI domain, it became evident that designing circuits without focusing on testing them would lead to increased test time and cost. This led to development of Design for Testability (DFT) engineering in order to facilitate testing. DFT, since then, has become an essential part of modern digital circuits [Wang VLSI]. The DFT techniques can be categorized as follows:

2.4.1 *Ad Hoc* Approach

Ad Hoc approaches rely on circuit modifications to make testing easier. These modifications can involve but are not restricted to following good design practices. One of such prominent approaches is *test point insertion.*

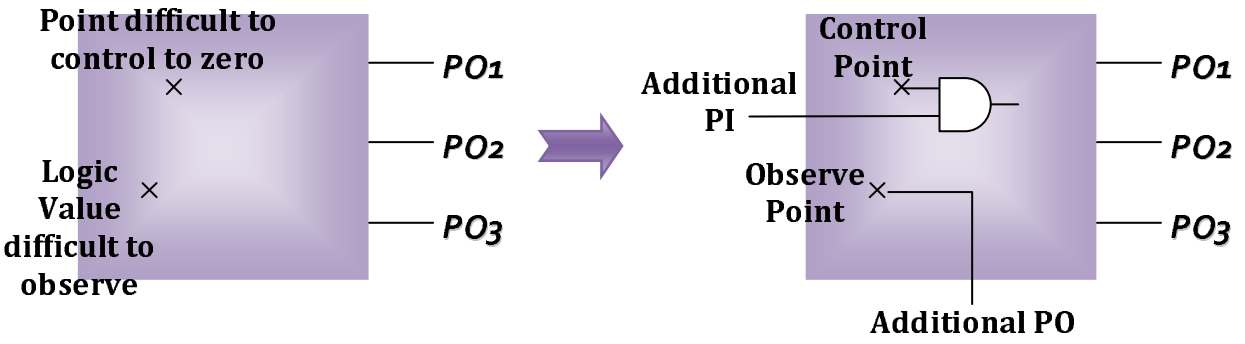


Figure 2.3: Control and Observe Points

Test Point Insertion

Test points are added to the circuit to make testing easier without changing the functionality of the circuit in the functional mode. In general, it involves adding *control/observe points* to improve the controllability and observability of the internal nodes respectively. A *control point*, when activated, increases the controllability of one or more signals in the circuit and requires an additional primary input. Further, it may also help in observability by sensitizing the otherwise blocked paths for certain faults in the CUT. On the other hand, an *observe point* enhances the observability of some internal signals and requires an additional primary output [Touba]. Figure 2.3 show a control point and observe point added to the circuit.

2.4.2 Structured Approach

Structured approach follow a more methodological and systematic flow to increase testability of the CUT. Scan design is the most widely used in this approach.

Scan Design

Sequential circuits update themselves based on the primary input and storage element values when the system clock is applied. Since, we cannot control the storage element values, it is

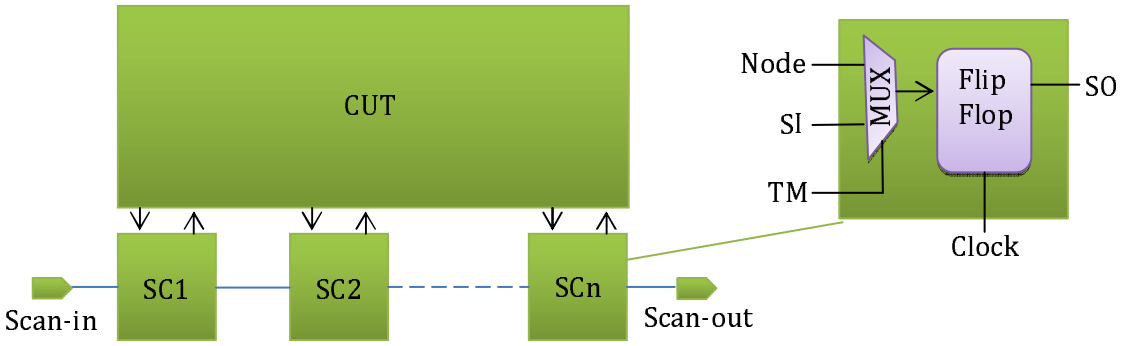


Figure 2.4: Scan Cell and Scan Chain

difficult to test such circuits. Scan design aims at making the storage elements in a sequential circuit completely controllable and observable. Each storage element is replaced by a scan cell as shown in Figure 2.4. The scan cells are then stitched together to form a chain by connecting the output of one cell to the input of next cell. Some additional pins, scan-in (SI), scan-out (SO) and test mode (TM) are also added. By setting the value of TM to logic '0', the circuit operates in its normal functional mode. When TM is logic '1', the circuit is in the test mode. In this mode, we can control each storage element by shifting in a particular value through the SI pin. Similarly, we can also observe the value of any storage element by shifting out the data through the SO pin in test mode, as shown in Figure 2.4. This transforms the sequential circuit into a combinational circuit.

Usually, a separate slow clock named test clock is used to shift data in and out of the storage elements. This increases the test application time. So, to reduce it, instead of having one single long chain, storage elements are divided into multiple scan chains.

2.4.3 Built-in Self Test

Built-in Self Test (BIST) employs a test pattern generator (TPG), logic controller and output response analyzer (ORA) as shown in the Figure 2.5. The TPG generates test vectors to be applied to the circuit automatically. Typically TPG is constructed from linear feedback

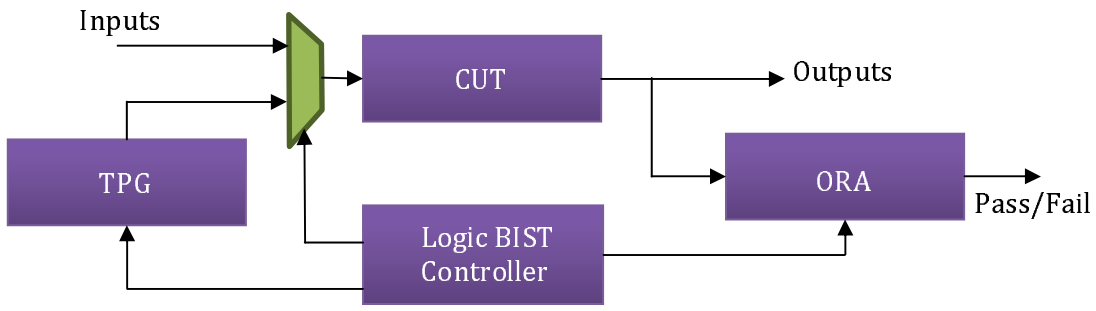


Figure 2.5: BIST Architecture

shift registers (LFSRs) and pseudo-random testing is used to generate a subset of test vectors $t \subset 2^n$, where n is the number of primary inputs of the circuit. The ORA compacts the output responses of the circuit into a *signature*. Typically ORA is constructed from multiple input signature register (MISR). This is a lossy compression and the final signature can have aliasing problems. The logic controller is responsible for generating control signals that trigger the BIST operation among the TPG, CUT and ORA. It gives a pass/fail indication at the end of BIST operation. Broadly BIST architectures can be classified as:

Online BIST

In this case, the BIST operation takes place when the circuit is in its normal functional mode. They can be further classified as:

- **Concurrent BIST**: The circuit is modified or duplicated to detect any errors and the detected error is corrected on the spot or an interrupt is generated.
- **Nonconcurrent BIST**: It is performed when the circuitry is in idle mode by executing some microcodes. When the circuitry wants to return from idle to functional mode, the microcode is stopped through interrupt.

Offline BIST

In this case, the BIST operation takes place when the circuit is not in the functional mode. They can be further classified as:

- **test-per-scan BIST**: They take advantage of built-in scan chains and apply a test pattern to the circuit after a shift operation is completed. The area overhead in this case is low.
- **test-per-clock BIST**: It applies a test pattern to the circuit and captures the response every system clock cycle. They can execute much faster as compared to *test-per-scan BIST* but the area overhead is more.

2.5 Design for Diagnosability

Diagnosis requires a high level of controllability and observability to come up with a small set of candidate locations where the defect manifests itself. Most digital circuits have low diagnostic resolution which makes it difficult to map a faulty response to its actual location in the circuit. So, it is necessary to modify the circuit so as to achieve an acceptable diagnostic resolution. The techniques achieving this are termed as Design for Diagnosability (DFD) and can be categorized as:

1. Logic DFD structures
    (a) Scan
    (b) Observation-Only Scan
    (c) Test Point Insertion
    (d) Array Dump and Logic Analyzer
    (e) Clock Control
    (f) Partitioning, Isolation and De-featuring

   (g) Reconfigurable Logic

2. Physical DFD structures

   (a) Mechanical Probing

   (b) Injection-Based Probing

   (c) Emission-Based Probing

We target Logic DFD structures in this thesis and explain few of them.

2.5.1 Scan

This is similar to the Scan in DFT. The increased observability of storage elements enables diagnosis. The test can be executed for a certain number of clock cycles and then stopped. The latched data at the storage elements can be dumped out through *SO* pin. This value can be compared with the expected state to find the buggy nodes. The only drawback is when the state is shifted out, the state is lost and test cannot be resumed from where it was stopped. In other words, this is destructive testing.

This can be overcome by wrapping the scan chain from scan-out to scan-in so that the contents of the storage elements are restored after completing a scan-out operation. This can be used only when all the storage elements are scanned.

2.5.2 Observation-Only Scan

It is also called as *scanout* or *shadow scan*. It is a separate chain of scan cells usually operated with system clock, specifically used for debugging purposes. Figure 2.6 shows a typical Observation-Only Scan cell. The LOAD signal when enabled latches the value at a point of interest. Enabling SHIFT signal shifts out the captured values of signals of interest.

These cells can be placed intelligently to observe hard-to-observe signals. The drawback of this technique is the high area overhead because of extra scan cells.

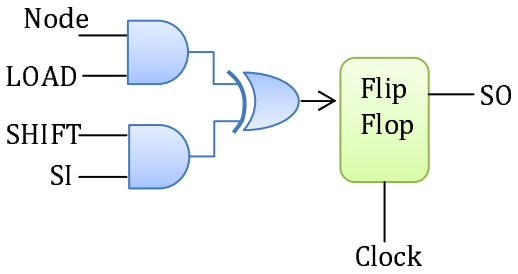


Figure 2.6: Observation-Only Scan Cell

2.5.3 Test Point Insertion

This is similar to the test point insertion in DFT. The only difference is here only *observe points* are added to the CUT so as to help diagnosis. Again, the points can be selected intelligently to observe hard-to-observe signals.

2.6 Summary

In this thesis, we employ the test point insertion technique for DFT as well as DFD. Also we focus on enhancing *observability* to aid both testability and diagnosability (i.e. add *observe points* to the CUT). In the sequel, we use the terms test points and *observe points* interchangeably. We also discussed relevant previous works to highlight what problems have already been addressed and what challenges are remaining.

Chapter 3

TD Points Selection for a Given Test Set

3.1 Motivation

Traditionally, test points added to the circuit were targeted specifically to increase either its testability or diagnosability. In this chapter, we develop a method to find intelligent test points such that they improve not only the *fault coverage* but also the *diagnostic resolution* of the CUT.

3.2 Problem Statement

Given a full-scan circuit C and a test vector set T, our objective is to insert as few TD points into the CUT such that both the *fault coverage* as well as the *diagnostic resolution* achievable by T can be improved as much as possible.

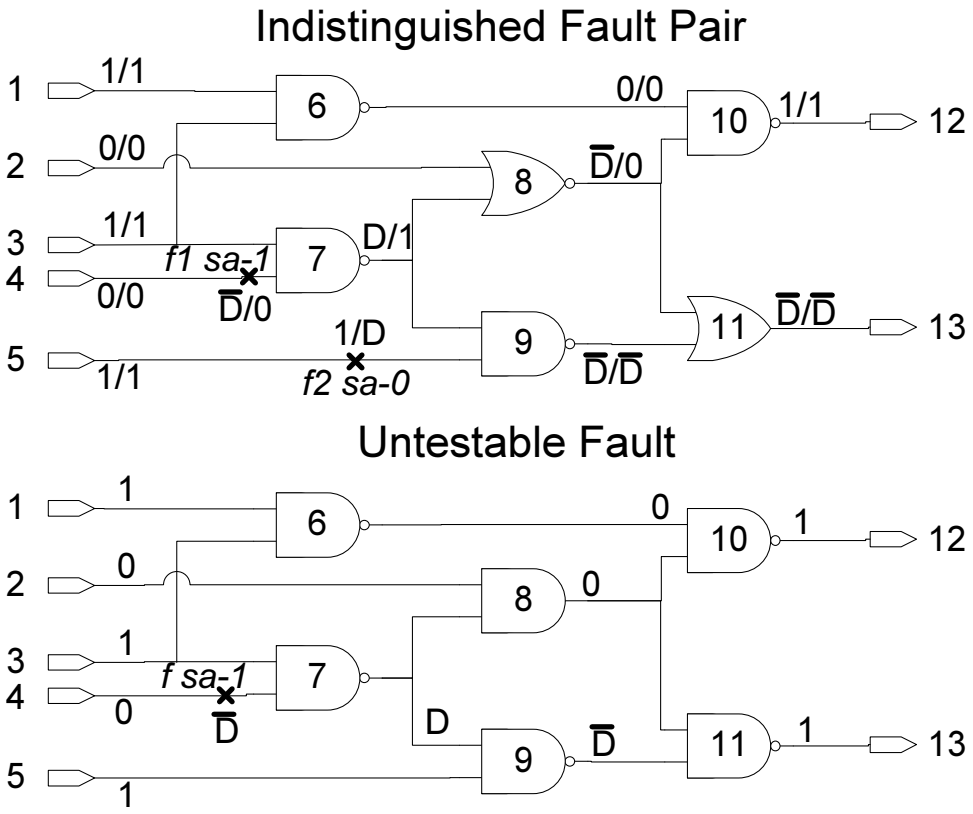


Figure 3.1: nodeCount example

3.3 Key Idea

The set of desired *TD* points are identified as the set of nodes in the CUT, which when made observable would either make an indistinguishable fault-pair distinguishable or detect a previously undetectable fault. In order to explain our method, the following definitions are needed.

3.3.1 nodeCount

With every node (gate) n in the circuit, we associate two counts: $nodeCount_u(n)$ and $nodeCount_i(n)$. $nodeCount_u(n)$ is the count for T-undetectable faults while $nodeCount_i(n)$

is the count for T- indistinguishable fault-pairs at node n. $nodeCount_u(n)$ is incremented whenever the faulty and fault-free values at node n are different for any T-undetectable fault during the fault simulation of T. Similarly, for a T- indistinguishable fault-pair ($f1$, $f2$), $nodeCount_i(n)$ is incremented whenever only one of $f1$ or $f2$ has its fault effect on node n for the test set T. Consider Figure 3.1. Without loss of generality, let us assume that only one vector -10101 - is present in the test set T. The top diagram in the figure shows two output responses for each gate. The first response is in the presence of fault $f1$ (2^{nd} input of gate 7 sa-1) while second response is in the presence of $f2$ (2^{nd} input of gate 9 sa-0). Now, ($f1$,$f2$) is a T-indistinguishable pair. If the output of gate 8 is made observable, $(f1, f2)$ will be T-distinguishable since $f1$ has its fault effect propagated to gate 8 but $f2$ has not. So, the $nodeCount_i$ for gate 8 is incremented. This process is repeated for all the T-indistinguishable pairs.

The bottom circuit in the figure shows T-undetectable fault f (2^{nd} input of gate 7 sa-1). Now, by making the output of gate 9 observable, f can be made T-testable since the fault effect of f has propagated to gate 9. So, $nodeCount_u$ for gate 9 is incremented. This process is performed for all T-undetectable faults. The above process can be easily repeated for test sets with more than one vector by updating the $nodeCount_{(u/i)}$ values for every vector. Note that $nodeCount_{(u/i)}$ values at fault sites are not updated (in Fig 3.1 top diagram, node 7 and 9 and in Fig 3.1 bottom diagram, node 7).

After obtaining $nodeCount_u$ and $nodeCount_i$ values for each node in the CUT, any gate with a non-zero $nodeCount_u(n)$ value indicates that at least one T-undetectable fault had propagated its fault effect to node n. Similarly, any gate with a non-zero $nodeCount_i(n)$ value indicates that at least one T-indistinguishable fault-pair differs in its fault effects at node n. Therefore, if we make a node with both non-zero $nodeCount_u$ and $nodeCount_i$ values observable, we can effectively detect at least one previously T-undetectable fault and distinguish at least one previously T-indistinguishable fault-pair. To select the best nodes from both $nodeCount_u$ and $nodeCount_i$, noderank and weighted average are introduced.

Table 3.1: nodeRank$_i$ Example

Before Sorting					
Node Number	1	2	3	4	5
$nodeCount_i$	15	14	7	10	20
After Sorting					
Node Number	3	4	2	1	5
$nodeCount_i$	7	10	14	15	20
Result					
$nodeRank_i(1) = 4$					
$nodeRank_i(2) = 3$					
$nodeRank_i(3) = 1$					
$nodeRank_i(4) = 2$					
$nodeRank_i(5) = 5$					

3.3.2 nodeRank

For each node n in the circuit, we associate two ranks, $nodeRank_u(n)$ for T-undetectable faults and $nodeRank_i(n)$ for T-indistinguishable fault-pairs, so as to help us rank the nodes. After computing $nodeCount_u(n)$ and $nodeCount_i(n)$, the $nodeRank_u(n)$ and $nodeRank_i(n)$ values are obtained by sorting the values of $nodeCount_u(n)$ and $nodeCount_i(n)$ in ascending order respectively. We define $nodeRank_u(n)$ as the index of the node n in the set ordered by $nodeCount_u$ value, while $nodeRank_i(n)$ is the index of the node n in the set ordered by $nodeCount_i$ value. Table 3.1 shows an example of some nodes with their $nodeCount_i(n)$ values and their $nodeRank_i(n)$ values obtained after sorting $nodeCount_i$ in ascending order. The $nodeRank_u(n)$ values can be computed similarly.

The ranking favors nodes with higher $nodeCount_{(u/i)}$ values as TD points. However, the number of T-undetected faults could be much fewer than the number of T-indistinguished fault-pairs (or vice versa), and care must be taken to evaluate the collective contribution from both factors. So, we compute a weighted average, as explained next.

3.3.3 Weighted Average

The Weighted Average (WA) is used as a metric to select the best nodes as *TD* nodes depending on $nodeCount_{(u/i)}$ and $nodeRank_{(u/i)}$ values. The skeleton of the formula is as shown below:

$$WA1(n) = \begin{cases} 0 & |U| = 0 \\ \alpha \times \frac{nodeCount_u(n) \times nodeRank_u(n)}{numUntestFlts} & |U| > 0 \end{cases} \tag{3.1}$$

$$WA2(n) = \begin{cases} 0 & |I| = 0 \\ \beta \times \frac{nodeCount_i(n) \times nodeRank_i(n)}{numIndistPairs} & |I| > 0 \end{cases} \tag{3.2}$$

$$WA(n) = WA1(n) + WA2(n) \tag{3.3}$$

where $numUntestFlts(|U|)$ is the number of T-undetectable faults, $numIndistPairs(|I|)$ is the number of T-indistinguishable fault-pairs, which help to normalize $nodeCount_u(n)$ and $nodeCount_i(n)$ values. α and β are positive weight factors.

If all we care is testability, we can simply set $\alpha = 1$ and $\beta = 0$; in the sequel we refer this as DFT configuration. Likewise, if we are satisfied with the *fault coverage* achieved with the test set T, but wish to increase the diagnosability, we can set $\alpha = 0$ and $\beta =$ 1 - DFD configuration. For those cases where we wish to enhance both testability and diagnosability, we can set α and β to non-zero positive values - DFT + DFD configuration. In our experiments, we will discuss different settings and their impact on *fault coverage* and *diagnostic resolution*.

3.4 Approach

3.4.1 Overall Flow

Figure 3.2 shows the overall flow of our method. Given a circuit C, and a set of test patterns T, we want to find a small set of *TD* points such that both *fault coverage* and *diagnostic*

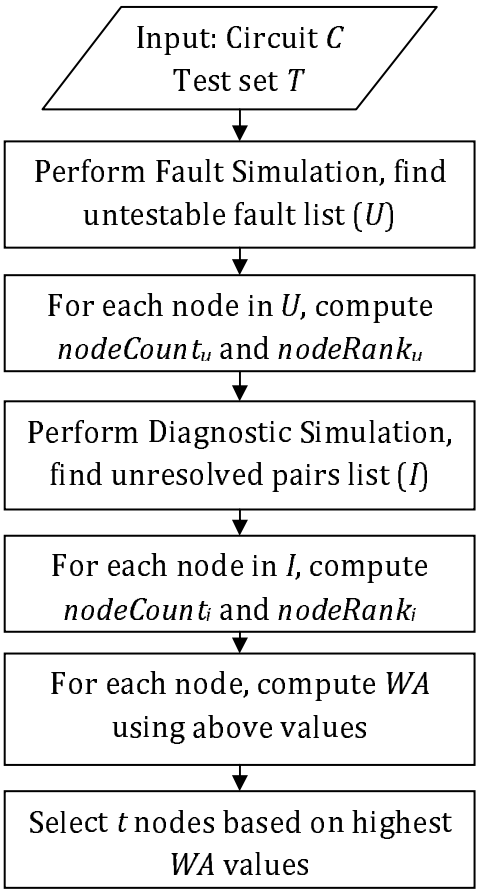


Figure 3.2: Algorithm Flow

resolution are improved simultaneously. The test vector set T can be obtained from any arbitrary test generator, including random vectors. A fault simulator is used to determine T-testable and T-undetectable faults, and a diagnostic simulator [Venkataraman] is used to determine T-indistinguishable fault-pairs from T-testable faults set.

Normally, the cost of fault simulation and diagnostic simulation varies with the size of the CUT. This is a preprocessing step required before applying our method and can be considered as just a one-time cost. At the end of these two simulations, we have an T-undetectable faults list (U) and an T-indistinguishable fault-pairs list (I). Using list U, $nodeCount_u(n)$ and $nodeRank_u(n)$ values are computed for each node n. Using list I, the

values of $nodeCount_i(n)$ and $nodeRank_i(n)$ are computed for each node n. With these values, $WA(n)$ is computed using Equation 3.3. Finally, the set of candidate nodes for *TD* points are identified by sorting the $WA(n)$ values in descending order and selecting the first t nodes in the ordered set. The value of t can be varied based on the desired *fault-coverage*, *diagnostic resolution*, and the area overhead (due to *TD* points insertion). Note that as the value of t increases, benefits obtained in terms of *fault coverage* and *diagnostic resolution* will decrease.

Overlapping issue

Inserting a node n based on the WA metric as a *TD* point provides for detecting (distinguishing) certain previously T-undetectable faults (T- indistinguishable fault-pairs) at n. The overlapping issue arises if the subsequent nodes are selected without considering the benefits brought by earlier selected node(s). Consider the case where a first *TD* node has been selected; if a second *TD* node is chosen using the same weighted average without any updates to the $nodeRank_{(u/i)}$ and $nodeCount_{(u/i)}$ values, then it may be the case that no new faults (or fault-pairs) are detected (or distinguished) at the second node. Thus, the second node does not add any value when the first node has already been added.

This overlapping issue is illustrated by the following example. Table 3.2 shows the case with five indistinguishable fault-pairs and five undetectable faults. The potential nodes (gate numbers) in the CUT where the pairs can be distinguished (faults can be detected) are listed after each pair (fault). There are 16 candidate nodes to be considered, between node 23 and node 188. The $nodeCount_{(u/i)}$ and $nodeRank_{(u/i)}$ values for these nodes are computed and shown in the table. Considering the case when $\alpha = 1$ and $\beta = 1$ in Equation 3.3, the weighted average values are also calculated and are shown at the bottom row for these 16 nodes in the table. Node 180 will be chosen first since it has the highest WA value of 19.2. Making node 180 as a *TD* point will distinguish $(f3, f4)$, $(f5, f6)$ and $(f9, f10)$ pairs and will detect $f12$, $f13$ and $f15$ faults. Now if the weighted average is used again without considering the overlapping issue, node 170 will be chosen as the next candidate since it has

Table 3.2: Overlapping Issue Example

Indistinguishable Fault Pairs	Untestable Faults
(f1, f2): 23, 45, 78, 110	f11: 34, 24
(f3, f4): 76, 85, 180, 170	f12: 180, 170, 24
(f5, f6): 180, 188, 170	f13: 170, 177, 180
(f7, f8): 23, 80, 120	f14: 23, 80
(f9, f10): 134, 180, 170	f15: 176, 180, 170

Result								
Nodes	23	24	34	45	76	78	80	85
$nodeCount_i$	2	0	0	1	0	1	1	1
$nodeRank_i$	14	1	2	6	3	7	8	9
$nodeCount_u$	1	2	1	0	0	0	1	0
$nodeRank_u$	9	14	10	1	2	3	11	4
WA	7.4	5.6	2	1.2	0	1.4	3.8	1.8
Nodes	110	120	134	170	176	177	180	188
$nodeCount_i$	1	1	1	3	0	0	3	1
$nodeRank_i$	10	11	12	15	4	5	16	13
$nodeCount_u$	0	0	0	3	1	1	3	0
$nodeRank_u$	5	6	7	15	12	13	16	8
WA	2	2.2	2.4	18	2.4	2.6	19.2	2.6

the next best value of 18.0. As can be seen from this example, making node 170 observable will neither detect any additional faults nor distinguish additional fault-pairs. This is because *only* the faults and fault-pairs that have conflicting values on node 180 have the same effect on node 170. In order to resolve this issue, after making node 180 observable, $(f3, f4)$, $(f5, f6)$, $(f9, f10)$ pairs should be removed from list I and $f12$, $f13$, $f15$ faults removed from list U. Now, based on the remaining indistinguishable fault-pairs[$(f1, f2)$ & $(f7, f8)$] and undetectable faults [$(f11$ & $f14)$], $nodeCount_{(u/i)}$, $nodeRank_{(u/i)}$ and WA values are updated. So, instead of node 170, node 23 will be chosen as the new *TD* point. Since the number of T-undetected faults and T-indistinguishable fault-pairs is small as compared to

the total number of faults and fault-pairs, and that the number of *TD* points inserted in the CUT is usually small, this method is computationally inexpensive.

3.4.2 The DFT + DFD Architecture

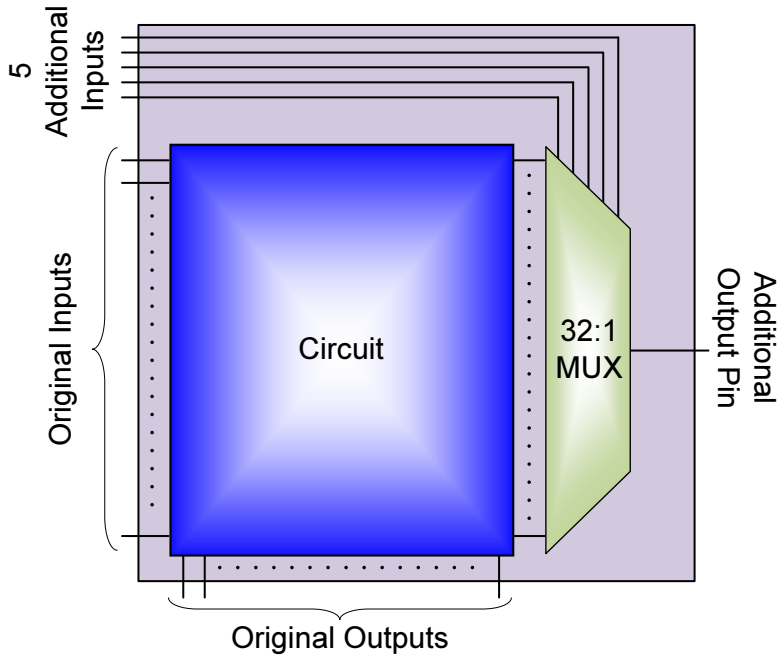


Figure 3.3: Conventional architecture Example

Traditional methods to insert observe points into the CUT adds additional circuitry and/or primary output (PO) pins to the CUT. Scan methodology is a popular DFT technique to insert test points that adds several pins like scan-in and scan-out ports, scan control signals and scan clock signals. One way of implementing non-scan designs is using MUX as shown in Figure 3.3. However, adding a MUX would lead to problems such as high area overhead and routing congestion. Also, the number of observe points added would depend upon the permissible size of the MUX. For example in Figure 3.3, only 32 internal nodes can be made observable using a 32:1 MUX which would add 6 additional pins to the CUT. In [Hsiao], the authors proposed an XOR tree based non-scan architecture. The advantage of using XOR

tree is that it reduces the number of additional pins. However, the number of gates added in a XOR tree grows exponentially to the number of nodes observed; this would lead to high area overhead. Further, there is an inherent problem of fault masking due to multiple path propagation. Since today's feature sizes are at the nano-scale, adding few additional gates to the design hardly adds any additional area when compared to adding an additional IO-pad. Hence, in our approach, we would like to add few additional gates to observe internal nodes without inserting any additional output pins!

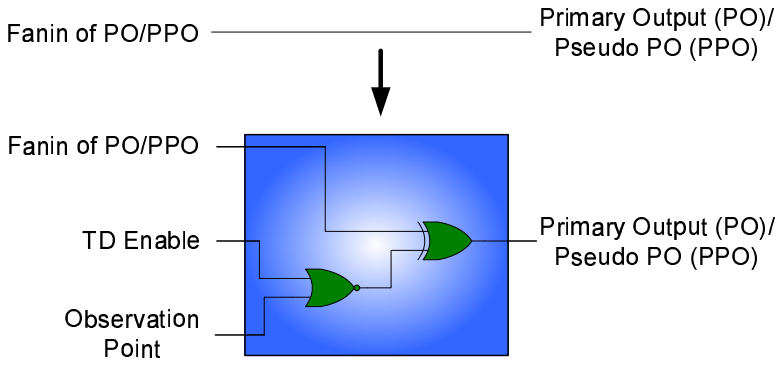


Figure 3.4: Utilizing existing POs/PPOs

Our solution is to utilize the existing pins of the CUT by adding small additional circuitries. Towards this objective, we target the re-use of (pseudo) primary outputs of the CUT. Our small additional circuitry consists of a 2-input NOR gate followed by a 2-input XOR gate. As shown in the Fig 3.4, XOR gate is added between a selected primary output 'O' and the input to 'O'. The other input of XOR gate comes from the output of NOR gate. The inputs of NOR gate are the observe point selected and an enable pin called the *TD Enable* pin. When *TD Enable* = 1, the CUT operates in its functional mode. When *TD Enable* = 0, the CUT operates in the test mode, which allows observe points to be sensitized to their respective (pseudo) POs.

The advantage of using a 2-input XOR gate is that it has no controlling value. It passes the fault effect at one of its inputs to its output regardless of whether the other input is at logic '1' or '0'. This is unlike the primitive gates, which will not propagate the fault effect if the

other input has a controlling value. Note that we need to observe the *TD* points only for the faults in U and pairs in I lists. For such cases, the original inputs to the PO will not have a fault effect for the considered test set T. Thus the probability for the *TD* points to be sensitized to their respective POs is higher for an XOR gate than a primitive gate. Further, considering 5-valued logic $\{0, 1, D, \overline{D}, X\}$, there are totally 16 possible input combinations for an arbitrary 2-input gate, where at least one input has a fault effect. Out of these 16 combinations, XOR gate will propagate the fault effect for 8 cases whereas any primitive gate will propagate for 6 cases[1]. Based on these observations, we prefer XOR gate over other primitive gate types for our architecture.

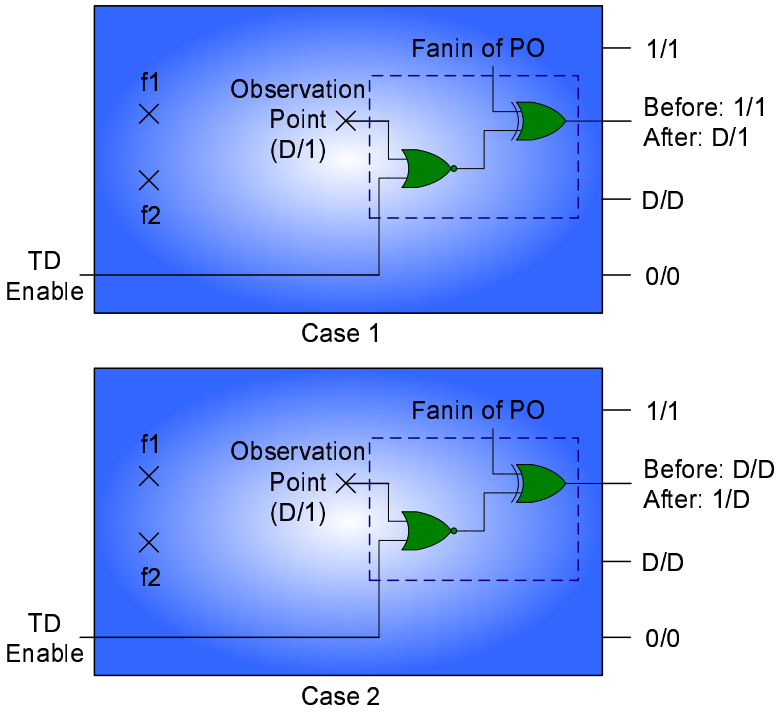


Figure 3.5: Effect on indistinguished fault-pair

Now, we discuss the proposed architecture from diagnosability viewpoint. Consider the example illustrated in Fig 3.5. It shows an observation point with the proposed architecture and an indistinguished fault-pair ($f1$, $f2$). Note that the fault-pair will have the same

[1]Refer Appendix A

output response at all the POs for all the test vectors in T. Assume that this fault-pair is distinguished at the observation point. Now, there can be two general cases.
Case 1: The observation point is routed to a PO which originally has no fault effect when considering the faults $f1$ and $f2$ individually. In this case, the response on this PO will be different for $f1$ and $f2$ since one of them will have D or $\overline{D}$ while the other will be logic '0' or '1'. So, ($f1$, $f2$) will be distinguished in the modified netlist.
Case 2: The observation point is routed to a PO which shows same fault effect for both $f1$ and $f2$. In this case too, the response would differ since it will be either D or $\overline{D}$ for one fault and either logic '0' or '1' for the other due to self-masking at the XOR gate.

XOR gate distinguishes any arbitrary fault-pair ($f1$, $f2$) for both the cases discussed above, provided the two faults differ in their values at the observation point. However, some other primitive gates may not distinguish ($f1$, $f2$) for both the cases. For example, OR/NOR gates will not distinguish ($f1$, $f2$) in Figure 3.5 *Case 1* and AND/NAND gates will not distinguish ($f1$, $f2$) in Figure 3.5 *Case 2*. This further justifies our use of an XOR gate.

One may observe that it is essential to select the right PO to sensitize a particular observation point. Otherwise, the originally detected faults may propagate a fault effect to both the inputs of the XOR gate in the modified netlist; thus becoming undetected. Similar argument can be made for an originally distinguished fault-pair. However, our experimental results based on random selection of POs to sensitize an observation point, indicate that such cases are rare. Further, in our architecture, the maximum number of internal nodes that can be made observable is equal to the number of primary outputs in the CUT. Note that the number of internal nodes to observe for a good improvement in *fault coverage* and *diagnostic resolution* are usually less than the number of POs and PPOs (scan-elements) available in the CUT. We also observed this trend in our experimental analysis. Since we do not have to utilize every PO, we can avoid those POs and PPOs that lie on critical paths. Finally, the area overhead incurred due to this architecture is minimal and will be shown in experiments section.

3.5 Experimental Results

The proposed technique was implemented in C++ and experiments were performed on Red Hat Linux workstations with Intel Pentium D 3.0GHz CPU, 8GB memory and 4 cores. We used ISCAS85 and full scan versions of ISCAS89 and ITC99 benchmark circuits for carrying out the experiments. We assumed a single stuck-at-fault model. The experiments were performed with two different test sets - a sub-optimal test set and a high quality test set. A sub-optimal test set (Part I - Section 3.5.1) is first used to show the effectiveness of our method. Next, we show that the results remain intact even when a high quality test set is used (Part II - Section 3.5.2). Finally, we report area overhead for the proposed method (Part III - Section 3.5.3).

3.5.1 Part I

In this part, the experiment was performed with a sub-optimal test set $T1$. A sub-optimal test set is used here to first illustrate the inner-workings of our method. Essentially, we avoid the cases where the *fault coverages* achieved are already 100%, leaving no room for further *fault coverage* improvement via *TD* point insertions.

Table 3.3 shows the results of our experiments for different values of α and β using $T1$. For each circuit, this table contains two sets of results - the first set reports the number of $T1$-indistinguishable fault-pairs and the second reports the number of $T1$-undetected faults. For each set, the results for different configurations are reported. Note that $|I|$ ($|U|$) represents the original number of $T1$-indistinguishable fault-pairs ($T1$-undetected faults) without any *TD* point insertions. Different values of α and β give different priorities to undetectable faults and unresolved pairs. When $\beta = 0$ and $\alpha > 0$, *TD* points are added taking into consideration just $T1$-undetectable faults, i.e., this configuration focuses only on DFT. When $\alpha = 0$ and $\beta > 0$, *TD* points are added taking into account only $T1$-indistinguishable fault-pairs, i.e., the configuration focuses only on DFD. For non-zero positive values of α and β, both $T1$-

undetected faults and $T1$-indistinguishable fault-pairs are considered for *TD* points selection. Further, α and β values decide which parameter to prioritize in *TD* point selection.

In order to achieve our goal of synergistically enhancing the testability and diagnosability of the CUT, we are interested in showing that a clever set of *TD* points can be selected from DFT + DFD configurations rather than considering DFT (or DFD) configuration alone. In Table 3.3, the column with $\alpha > 0, \beta = 0$ reports the pure DFT case. Though the number of $T1$-undetected faults is the least for this DFT-only configuration, there is no significant reduction in the number of $T1$-indistinguishable fault-pairs. For example, in circuit $b13$, while the number of undetected faults is reduced from 16 to 0 in DFT, the number of $T1$-indistinguishable fault-pairs is dropped only from 53 to 42. The observation is similar when we consider only the DFD configuration under the $\alpha = 0, \beta > 0$ columns. While the number of $T1$-indistinguishable fault-pairs is drastically reduced, the number of $T1$-undetected faults usually remains high. For instance, in c2670, although the DFD-only configuration reduces the number of $T1$-indistinguishable fault-pairs, the number of $T1$-undetected faults remained 151, which can actually go down to 0 faults by selecting different sets of test points.

Considering either DFD or DFT configurations alone are usually lop-sided; they focus on either $T1$-indistinguishable fault-pairs or $T1$-undetected faults alone. However, when combined, they can improve both aspects significantly. In particular, we were able to distinguish *all* unresolved pairs and detect *all* detectable faults for $c499$, $s510$, $s1488$, $s5378$ and $b13$ with DFT+DFD. The last row in Table 3.3 shows the percentage improvement in the number of fault-pairs distinguished over $|I|$ and the percentage improvement in the number of faults detected over $|U|$ by the corresponding configuration. On an average, we were able to resolve $4\times$ more previously indistinguishable fault-pairs for our configurations in comparison with the DFT configuration, while still achieving a similar *fault coverage* as in the DFT configuration. Similarly, on an average, DFT + DFD configuration has $1.6\times$ fewer undetected faults in comparison with the DFD-only configuration, while maintaining a similar *diagnostic resolution*. This indicates that the proposed heuristic and DFT + DFD architecture indeed helps in simultaneously enhancing the testability and diagnosability of the CUT.

3.5.2 Part II

In the second set of experiments, a high quality test set is used to show that our method indeed improves the diagnosability of the CUT without sacrificing the fault coverage. ATOM [Hamzaoglu] is used for generating a high quality test set $T2$ for each circuit. Table 3.4 shows the results of our experiments, again for different values of α and β. The layout for Table 3.4 is similar to that of Table 3.3. Since the number of $T2$-undetected faults (not including any faults proven untestable by ATOM) is zero for all the circuits, test point insertions must be done to only improve the diagnostic resolution. Note that, in such cases, for a given value of β, irrespective of any value of α, the fault coverage and diagnostic resolution obtained will be identical (refer Equation 3.3). Thus, we just report the values when $\alpha = X$, $\beta > 0$. In circuit s9234, initially there were 1252 $T2$-indistinguished fault-pairs. By the insertion of *TD* points, we were able to reduce this number down to 147. Similarly, the number of $T2$-indistinguished fault-pairs were reduced from 1311 to 620 for c7552, 1009 to 211 for b05, 49 to 9 for b13. From this table, we observe that even for a high-quality test set, our method distinguishes several fault-pairs without compromising the fault coverage.

3.5.3 Part III

We modified the original netlist to observe the t *TD* points based on the proposed architecture. This modified netlist was then synthesized by Synopsys DC Compiler [Synopsys] using VTVT TSMC $180nm$ library [VTVT] to obtain the area overhead. Table 3.5 shows the decrease in the number of undetectable faults and indistinguished fault-pairs with increasing number of *TD* points for various circuits. The numbers in the top row are the number of *TD* points added to the respective circuit. There are two rows associated with each circuit. The first row shows the number of undetectable faults left and the second row gives the number of indistinguished fault-pairs left after adding t number of *TD* points, where t is the number in the topmost row. Using the data from Table 3.5 and area numbers after synthesis, we plot graphs for few circuits - shown in Figure 3.6 and 3.7. Each graph shows the % improve-

ment in the number of undetectable faults, % improvement in the number of indistinguished fault-pairs and % area overhead as a function of the number of *TD* points. These graphs show that as the number of *TD* points added to the circuit increases, the benefit obtained in terms of leftover undetectable faults and leftover indistinguished fault-pairs decreases, as expected. Thus, for each circuit, there will be an optimal point where a satisfactory fault coverage, reasonable diagnostic resolution and a minimum area overhead can be achieved. From the graphs, we see a trend that the optimal point lies around 5% area overhead for the considered benchmark circuits. We postulate that for industrial-sized circuits, where the number of logic gates are usually huge, the area overhead for the optimal point would be significantly lower.

3.6 Summary

In this chapter, we developed a low cost heuristic to identify TD points in the CUT to synergistically increase its *fault coverage* and *diagnostic resolution*. We also proposed a novel DFT+DFD architecture which requires just one extra input pin. Experimental results showed that the proposed architecture distinguishes 4× more fault-pairs than existing DFT architecture while maintaining a similar fault coverage even for a high quality test set. Further, quality results can be achieved with an area overhead of around 5%.

Table 3.3: Results for $\alpha = 0, 1, 2$ and $\beta = 0, 1, 2$

Circuit	$T1$-Indistinguishable Fault Pairs						$T1$-Undetected Faults					
		DFD	DFT	DFT + DFD				DFD	DFT	DFT + DFD		
	$\|I\|$	$\alpha_0\beta_{>0}$	$\alpha_{>0}\beta_0$	$\alpha_1\beta_1$	$\alpha_1\beta_2$	$\alpha_2\beta_1$	$\|U\|$	$\alpha_0\beta_{>0}$	$\alpha_{>0}\beta_0$	$\alpha_1\beta_1$	$\alpha_1\beta_2$	$\alpha_2\beta_1$
c432	28	3	25	9	9	9	4	3	0	0	0	0
c880	57	5	66	10	10	10	6	5	0	0	0	0
c1355	747	584	747	584	584	584	0	0	0	0	0	0
c1908	375	264	295	273	273	273	24	12	0	4	4	4
c2670	445	27	433	29	29	28	310	151	0	0	0	1
c3540	541	293	532	303	303	303	7	3	0	4	4	4
c5315	447	144	447	144	144	144	0	0	0	0	0	0
c6288	980	944	980	944	944	944	0	4	0	4	4	4
c7552	1311	666	994	660	660	660	260	20	0	0	0	0
s420	37	20	35	21	21	21	11	12	2	5	5	5
s526	44	2	30	9	9	5	35	21	0	0	0	4
s713	187	56	169	57	57	57	19	16	0	0	0	0
s820	58	10	35	11	15	11	108	99	0	10	3	13
s832	59	11	35	12	15	11	112	103	0	12	7	14
s838	48	3	37	3	3	3	26	19	9	9	9	9
s953	79	8	26	5	7	6	81	39	1	1	1	1
s1196	93	14	27	14	15	12	104	36	0	8	5	14
s1238	132	29	43	24	30	21	122	53	0	7	0	12
s1423	177	6	145	4	5	4	17	1	0	0	0	0
s5378	559	0	545	0	0	0	62	46	0	0	0	0
s9234	1659	289	1110	284	286	281	599	141	1	1	1	5
s13207	2490	119	2117	157	159	159	324	91	0	6	6	6
s15850	2991	185	2388	175	175	175	556	243	8	5	5	5
b04	143	12	70	13	13	12	95	27	0	4	4	4
b05	1116	139	792	143	143	143	42	9	0	0	0	0
b07	85	104	63	9	9	9	15	16	0	0	0	0
b08	89	1	15	1	1	0	41	16	0	0	0	2
b09	22	1	5	1	1	1	50	8	0	0	0	0
b11	311	147	263	149	149	149	50	7	0	0	0	0
b12	121	21	54	15	21	17	109	33	0	0	0	0
b13	53	2	42	0	0	0	16	4	0	0	0	0
% Imp	-	73.65	18.58	73.96	73.78	74.02	-	60.81	99.35	97.45	98.26	96.61

Table 3.4: Results for $\alpha = X$ and $\beta > 0$ using a high quality test set

Circuit	$T2$-Indistinguishable Fault Pairs			$T2$-Undetected Faults		
		DFT	DFT+DFD		DFT	DFT+DFD
	$\|I\|$	$\alpha_{>0}\beta_0$	$\alpha_X\beta_{>0}$	$\|U\|$	$\alpha_{>0}\beta_0$	$\alpha_X\beta_{>0}$
c432	13	13	3	0	0	0
c499	12	12	0	0	0	0
c880	57	57	26	0	0	0
c1355	740	740	628	0	0	0
c1908	295	295	247	0	0	0
c2670	476	476	23	0	0	0
c3540	532	532	365	0	0	0
c5315	447	447	143	0	0	0
c6288	980	980	931	0	0	0
c7552	1311	1311	620	0	0	0
s420	13	13	0	0	0	0
s444	99	99	9	0	0	0
s526	34	34	2	0	0	0
s713	174	174	41	0	0	0
s820	42	42	3	0	0	0
s832	48	48	8	0	0	0
s838	28	28	5	0	0	0
s1196	16	16	0	0	0	0
s1238	42	42	1	0	0	0
s1423	148	148	0	0	0	0
s1488	23	23	0	0	0	0
s1494	27	27	0	0	0	0
s9234	1252	1252	147	0	0	0
b05	1009	1009	211	0	0	0
b07	32	32	0	0	0	0
b08	72	72	4	0	0	0
b09	18	18	0	0	0	0
b10	12	12	5	0	0	0
b11	282	282	182	0	0	0
b12	54	54	15	0	0	0
b13	49	49	9	0	0	0
% Imp	-	00.00	55.70	-	00.00	00.00

Table 3.5: Untestable Faults & Indistinguished Fault-pairs with Increasing # of *TD* Points

Circuit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
c1908	24	14	6	4	2	1	0	0	0	0	0	0	0	0	2	4	4
	375	313	308	301	297	296	296	295	293	291	287	287	286	284	281	283	289
c2670	310	155	38	38	21	9	9	9	9	9	9	9	9	9	3	3	3
	445	438	436	394	393	393	383	365	356	347	338	329	320	311	311	304	297
c7552	260	112	29	29	29	29	23	23	20	20	20	18	16	14	12	12	9
	1311	1251	1093	1060	1059	1046	1045	1044	1038	1018	1017	1009	1001	997	993	991	953
s526	35	27	19	15	12	9	9	8	7	6	6	6	6	6	5	5	4
	44	40	40	38	37	37	34	32	30	28	25	23	21	19	18	16	15
s820	108	83	80	67	56	47	46	40	33	26	26	19	19	16	12	12	12
	58	55	49	48	46	45	40	38	37	36	32	32	29	28	28	26	24
s832	112	87	84	71	60	51	50	44	35	28	21	21	21	18	18	18	18
	59	56	50	50	48	47	42	40	39	38	37	33	30	29	27	25	24
s953	81	72	52	43	39	36	33	30	25	24	22	20	18	16	14	11	11
	79	62	58	51	46	44	42	40	39	36	33	32	30	29	28	28	26
s1196	104	63	47	35	28	21	17	11	9	7	6	4	2	0	0	0	0
	93	81	74	65	52	49	45	45	43	39	37	34	32	30	29	26	26
s1238	122	83	66	53	45	39	32	25	20	17	15	12	8	4	1	0	0
	132	114	104	94	79	75	73	70	66	62	58	55	53	51	50	47	42
s1423	17	12	8	5	3	2	1	0	0	0	0	0	0	0	0	0	0
	177	157	150	146	141	139	137	136	135	133	131	127	123	119	115	113	111
s5378	62	55	49	44	41	38	35	32	29	26	24	22	20	18	16	14	12
	559	556	556	555	551	548	547	547	547	547	546	546	546	546	546	546	546
s9234	599	432	325	248	206	174	154	139	122	122	122	113	106	99	92	85	77
	1659	1596	1507	1432	1597	1586	1582	1569	1565	1529	1496	1490	1480	1471	1462	1462	1441
s13207	324	318	314	314	311	309	307	305	304	302	299	297	296	295	293	292	291
	2490	2490	2488	2450	2444	2435	2434	2434	2409	2396	2396	2396	2396	2396	2396	2394	2389
s15850	556	451	364	319	281	281	257	241	241	228	216	206	196	185	174	164	154
	2991	2991	2988	2886	2826	2669	2663	2653	2564	2559	2559	2551	2545	2545	2545	2543	2543
b05	42	16	16	13	5	2	2	2	2	1	1	0	0	0	0	0	0
	1116	1008	769	613	607	577	493	469	410	400	388	388	372	339	323	323	312
b11	50	30	20	14	10	7	4	2	2	2	1	0	0	0	0	0	0
	311	286	282	282	279	276	273	263	235	218	214	211	204	204	198	197	191
b12	109	77	64	56	46	45	39	36	33	28	24	22	21	19	18	18	15
	121	116	102	91	92	92	84	81	78	78	77	72	70	66	63	61	61

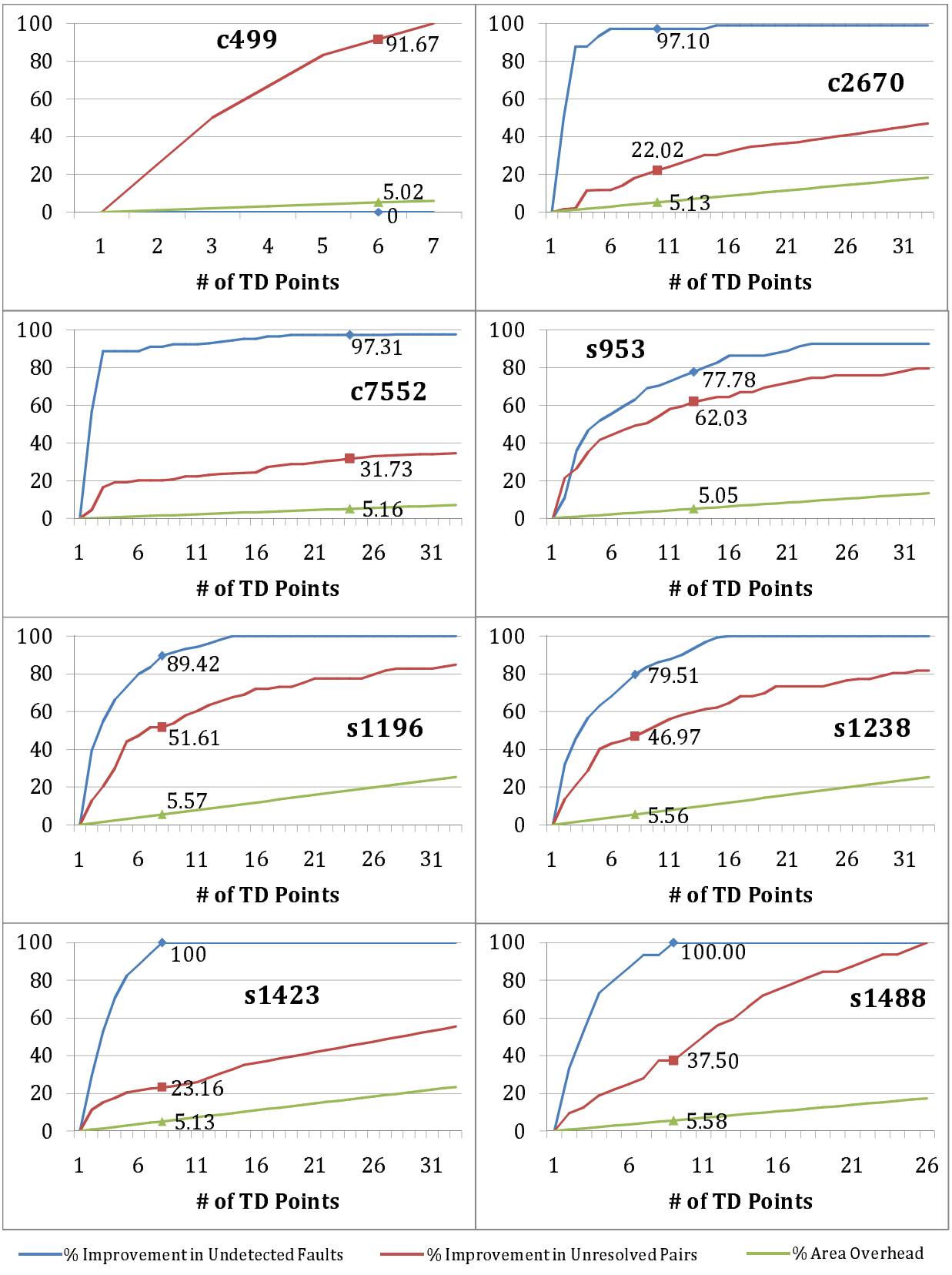


Figure 3.6: % Improvement in Undetectable Faults, % Improvement in Indistinguished Fault-Pairs, and % Area overhead vs. # TD points

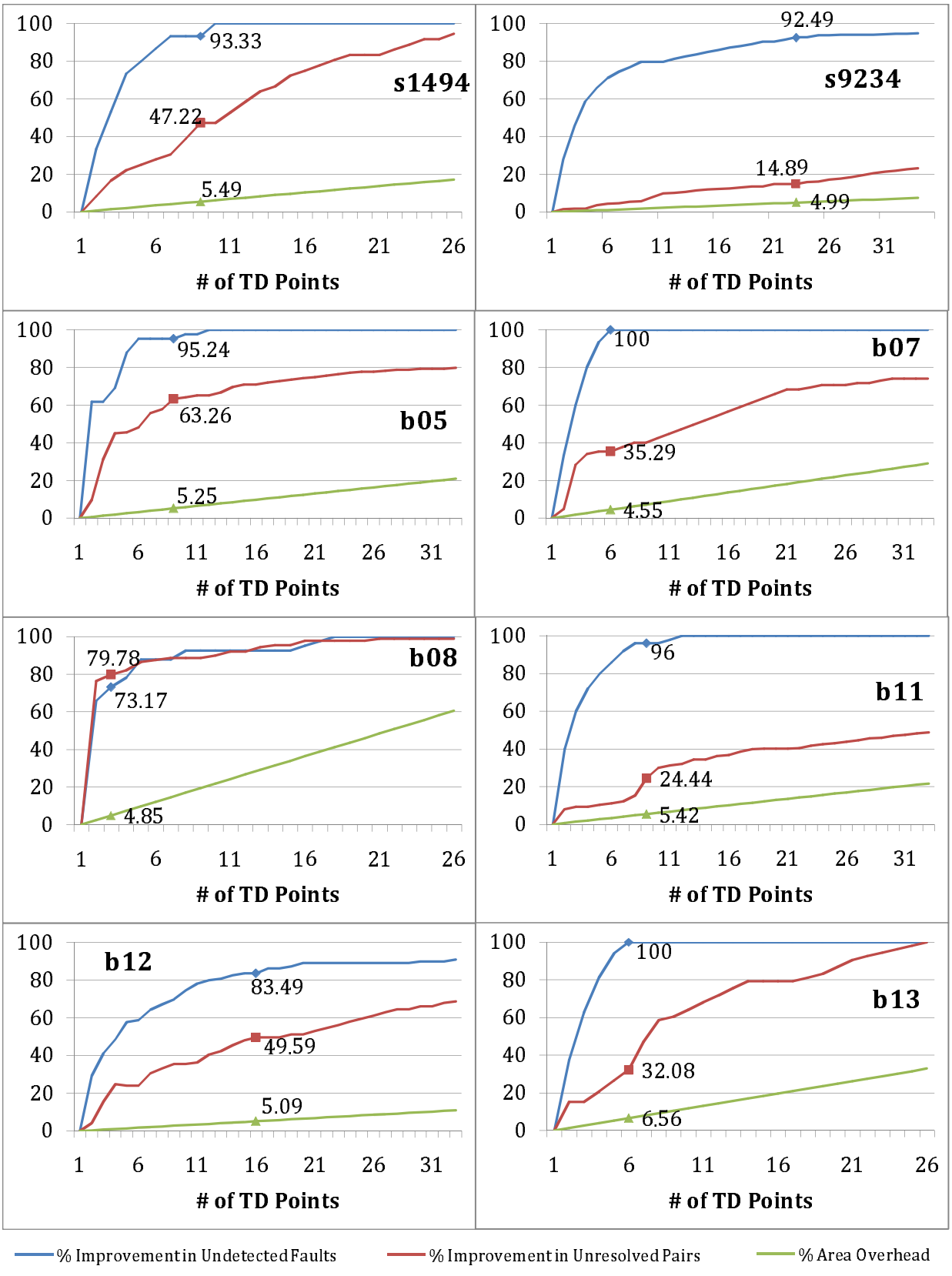


Figure 3.7: % Improvement in Undetectable Faults, % Improvement in Indistinguished Fault-Pairs, and % Area overhead vs. # TD points

Chapter 4

TD Points Selection Without Test Set

4.1 Motivation

The previous approach enabled to find TD points in the presence of a given test set T. In this chapter, we target the problem of synergistic enhancement of *fault coverage* and *diagnostic resolution* of the circuit when no test set is given. This chapter proposes novel heuristics to select TD points using structural characteristics of the circuit. refer Section 3.3.1, *nodeRank* - refer Section 3.3.2 and WA - refer Section 3.3.3. Only in this case, their values will be calculated based on circuit characteristics.

4.2 Problem Statement

Given a full scan circuit C, the objective is to insert as few TD points into the CUT such that any ATPG run on the modified C will give both enhanced *fault coverage* as well as improved *diagnostic resolution*.

4.3 Key Ideas

4.3.1 Unique Requirements

Unique requirements for a fault f is the set of gate assignments, represented as UR_f, that must be satisfied by any test vector t so as to detect f. If the set UR_f is inconsistent i.e. it is impossible for any test vector t to achieve the gate assignments in UR_f, then fault f can be concluded as a redundant fault. Though it is hard to compute the complete set of unique requirements for an arbitrary fault f, there are low cost techniques to compute an incomplete set UR_f fairly quickly [Vimjam, Iyer].

4.3.2 Z-sets

The concept of z-sets was introduced in [Pomeranz] where the circuit's structural characteristics are used to determine the number of fault-pairs that are guaranteed to be distinguished. The basic idea is to identify the primary outputs where a fault f can be detected, called a "Z-set". The faults whose Z-sets do not intersect are guaranteed to be distinguished provided a fault detection test set is available. An example is shown in Figure 4.1.

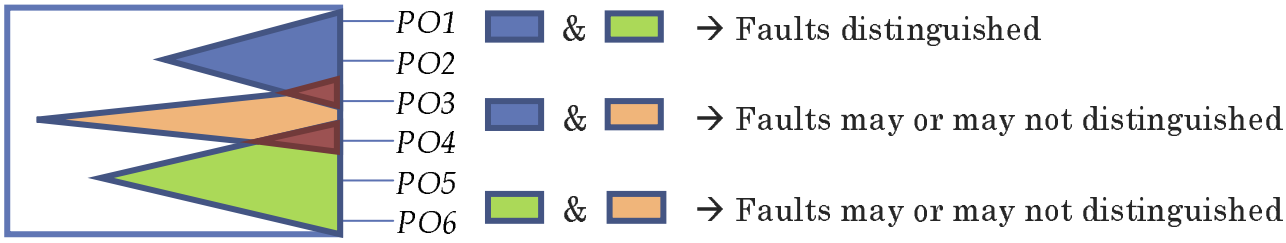


Figure 4.1: Z-set Example

4.3.3 Inversion Value - Heuristic $H1$

After using Z-sets, we are left with the faults that share or have the same Z-sets. In such cases, the faults may be detected on the same primary outputs. Now considering stuck-at fault model, it is not that such faults identified will always be indistinguished. Figure 4.2 shows two faults $(f1, f2)$ that are detected at the same primary output z (same Z-set). Without loss of generality, assume that there are even number of inverting gates between $f1$ & z and between $f2$ & z. For the stuck-at fault model, we have the following cases:

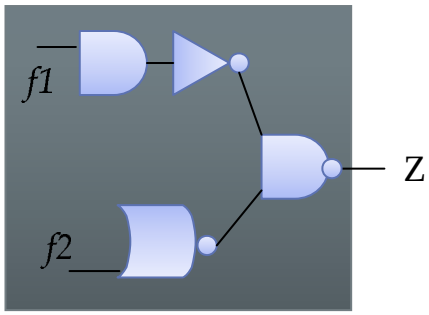


Figure 4.2: Inversion Value Example

I) $f1$ and $f2$ are both $sa-0$. In this case, both the faults if detected will produce the same faulty response at the respective outputs (D). So, $(f1, f2)$ is an indistinguished fault-pair.

II) $f1$ is $sa-0$, $f2$ is $sa-1$. Here, if $f1$ is detected, the faulty response produced would be D whereas if $f2$ is detected, the faulty response would be $\overline{D}$. From this, we can see that the fault-free value at the primary output for detecting $f1$ is different from the fault-free value for detecting $f2$. In other words, a test vector that produces a fault-free value of 1 will detect $f1$ but not $f2$ and a test vector that produces a fault-free value of 0 will detect $f2$ but not $f1$. Thus, we can conclude that $(f1, f2)$ pair is distinguished.

III) $f1$ is $sa-1$, $f2$ is $sa-0$. This is similar to the previous case where $(f1, f2)$ pair is distinguished.

Algorithm 1 InversionFlag(Circuit, g)

Require: $numinversion \leftarrow 0,\ gate \leftarrow g$

1: **if** $g = primary\ output\ po$ **then**
2: **if** $numinversion == odd$ **then**
3: $set\ inversionflag[gate][po]$
4: **else**
5: $reset\ inversionflag[gate][po]$
6: **end if**
7: **else**
8: $numinversion \leftarrow 0$
9: $choose\ a\ fanout\ fout\ of\ g\ easiest\ to\ observe$
10: **if** $fout\ is\ an\ inverted\ gate\ type$ **then**
11: $numinversion++$
12: **end if**
13: InversionFlag(Circuit, fout)
14: **end if**

IV) $f1$ and $f2$ are both $sa-1$. This is same as Case I. The faulty response produced would be $\overline{D}$. Again, $(f1, f2)$ is an indistinguished fault-pair.

From the above analysis, we can conclude that the fault pairs $(f1, f2)$ that produce opposite faulty values on the same primary output are guaranteed to be distinguished. Since the stuck-at values of each fault f are known, we can determine the fault value that f will produce at a primary output o, by keeping track of the number of inversion gates between the fault site and o. So, for each fault f, we define a $inversionflag$ vector. The size of this vector is equal to the number of primary outputs of the CUT. Note that this step is linear to the number of faults (or size) of the circuit. Calculating $inversionflag$ vector for a gate g is shown in Algorithm 1.

4.3.4 Observability Vector - Heuristic $H2$

After using Heuristic $H1$, we are left with fault-pairs that are detected on the same output pins and have the same faulty value when sensitized. The ATPG algorithm for any fault finds a path that is easiest to propagate the fault effect. Essentially, it targets a path of gates that have the least observability values. The CO value for each gate is the cost needed for that particular gate to observe at a specific primary output. The fault can propagate to a bunch of POs, but the CO value is calculated taking into consideration just a single PO where it can propagate. ATPG will try to propagate the fault to the PO which is closest. If the fault effect is masked in between, the algorithm will backtrack and select another path which is the next best. This process continues until the fault effect is/isn't observed at a primary output. So to restate this, each fault has a specific order of primary outputs where it is detected. If we can find this order for each fault, then we can conclude that the fault-pairs that have different PO orders have a chance to be distinguished.

In order to do so, we need to calculate the observabilities of each gate with respect to the POs where they can be observed. For this, we define an observability vector CO of size equal to number of POs for each gate. Again computation of this vector is linear to the size of the circuit which is shown in Algorithm 2.

4.4 Approach

4.4.1 Overall Flow

Figure 4.3 shows the overall flow of this method. Given a full scan circuit, we form a single stuck-at fault list. Using SCOAP values, we find out the nodes with highest observability values i.e. the nodes that are hard to observe and compute $nodeCount_u$ and $nodeRank_u$ values. Then, applying the criterion, Z-set and UR, and heuristics, $H1$ and $H2$, we are left with fault-pairs that are not distinguished. So, for every fault in such pair, we compute the

Algorithm 2 ObservabilityVector(Circuit)

```
1: CO[g][po] ← number of gates + 1 //forall gates g and forall primary outputs po
2: for all gate g //Tracing circuit from output to input do
3:   for all primary output po do
4:     if g is a primary output then
5:       if g == po then
6:         CO[g][po] ← 0
7:       end if
8:     else
9:       for all fanin gate fin of g do
10:        currentvalue ← CO[g][po] + controllability values of other fanins
11:        if currentvalue < CO[fin][po] then
12:          CO[fin][po] ← currentvalue
13:        end if
14:      end for
15:    end if
16:  end for
17: end for
```

set of gates in its fanout cone. The gates that are not common to both the sets are the potential test points that will help in distinguishing such fault-pair. Refer Figure 4.4. So, we update the values of $nodeCount_i$ and $nodeRank_i$ for the identified nodes. The WA metric is used to identify t test points by selecting the nodes with highest WA values.

4.4.2 Architecture

We use the same DFT + DFD architecture as described in Section 3.4.2. So, the maximum number of test points added to the circuit is equal to the number of primary outputs of the circuit. $NOR + XOR$ logic with an extra $TDEnable$ pin are added to the circuit which makes the area overhead same as before.

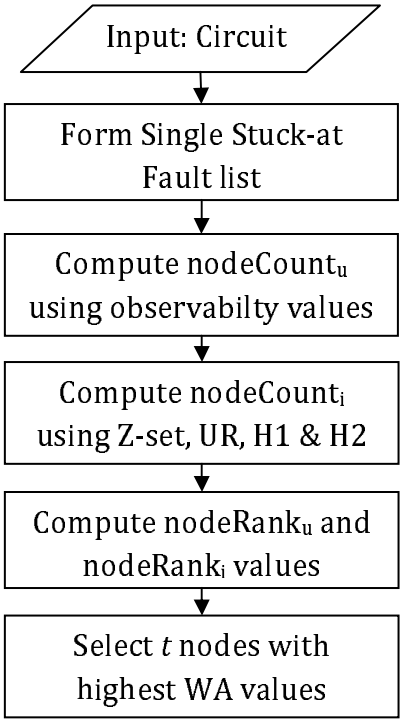


Figure 4.3: Overall Flow

4.5 Experimental Results

The proposed technique was implemented in C++ and experiments were performed on Red Hat Linux workstations with Intel Pentium D 3.0GHz CPU, 8GB memory and 4 cores. We

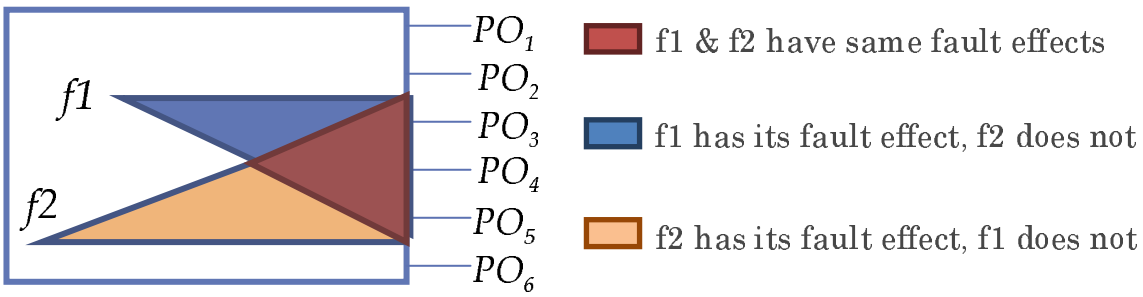


Figure 4.4: Potential nodes for indistinguished fault-pairs

used ISCAS85 and full scan versions of ISCAS89 and ITC99 benchmark circuits for carrying out the experiments. We assumed a single stuck-at-fault model. The ATPG used on the modified circuits was ATOM [Hamzaoglu]. Part I - Section 4.5.1 is first used to show the fault pairs that are left after using heuristic $H1$ and $H2$. Next, we show our results after using a ATPG on the modified circuit (Part II - Section 4.5.2). Finally, we report the results for TMR circuits (Part III - Section 4.5.3) to show the effectiveness of our method.

4.5.1 Part I

In this part, we implemented the criterion, Z-set and Unique Requirements, and the heuristics, Inversion Values and Observability Vector, to calculate the number of potential fault-pairs that are not distinguished. Table 4.5.1 shows the % of indistinguished fault-pairs left after applying these criterion and heuristics on the benchmark circuits. The first column is the circuit used, followed by % of pairs left after using Z-set, UR, $H1$ and $H2$. Note that the criterion and heuristics are applied one after the other. So, first Z-set is used to identify indistinguished pairs, then UR is used to further narrow the identified pairs, then heuristic $H1$ is applied to obtain the reduced set and finally heuristic $H2$ gives us the final result.

For most circuits, the % of pairs left is around 3% which is a significant improvement. For bigger circuits, like $s38584$ the % of pairs left is just 0.08%. Similar results follow for $s15850$, $s35932$, and $s38417$. We can see that on an average there is little drop in the number of fault-pairs after just using Z-set and UR. But using $H1$ and $H2$, we were able to achieve a significant improvement. From this, we can see that $H1$ and $H2$ certainly help in narrowing down the set of indistinguished fault-pairs, which facilitates our process of identification of potential test points. The next two parts will focus on how good our heuristics are.

4.5.2 Part II

In this part, we identify the test points using the WA criterion and insert them into the circuit using the proposed DFT + DFD architecture. The number of test points added to the circuits is equal to the number of primary outputs of the circuit. Next, we run an ATPG tool named ATOM [Hamzaoglu] on the modified circuit to get the values of *fault coverage* and *diagnostic resolution*.

Table 4.5.2 shows the result obtained after running the ATPG tool on the modified circuit. The first column shows the circuit used. For each circuit, the table has three sets of results - the first sets represents the number of detected faults, the second set gives the number of unresolved pairs that are left and the third set gives the number of test vectors generated. For each set, the results for original, after random test point insertion and after test point insertion using $H1 + H2$ are given. The table shows that using a smaller test set, we can get improvement in *fault coverage* as well as *diagnostic resolution*. Especially for circuit $s38684$, the number of test vectors reduced from 1031 to 911, the number of detected faults increased from 34797 to 35040 and the number of unresolved pairs left reduced from 2768 to 1871. The other circuits depict a similar outcome. On an average, for a test set reduction of 25.27%, the FC was improved by 0.81% and the DR was improved by 39.81%. There is not a significant improvement because the circuits used are highly testable with a high *fault coverage*, which leaves no or little room for improvement, even though the *diagnostic resolution* has increased considerably. In order to show the effectiveness of our heuristics, we use our method on a hard-to-test circuit in the next part.

4.5.3 Part III

In this part, we use a circuit structure named *triple modular redundancy* (TMR) and use the method developed to come up with test points. Again, ATOM [Hamzaoglu] is used to report the results obtained. But first we give a brief introduction about TMR circuits.

TMR Circuits [Fang]

In many critical applications like aerospace, it is important that the circuit behaves correctly at all the times even when it is affected by some cosmic rays which are capable of flipping a value at a node. So, fault-tolerant circuits were developed to account for such applications. The idea here is to introduce redundancy in the circuits to counterfeit the defects that may occur in the circuit. One of the popular fault-tolerant circuits is the TMR circuit. In TMR, three identical modules are connected in parallel and their outputs are given to a majority voter as shown in the Figure 4.5. So, the TMR circuit will guarantee a correct output response as long as the defects in two or more modules do not affect the same output feeding the majority voter. As a result, TMR significantly reduces the risk of incorrect output but this makes it very difficult to test or diagnose it. Any single stuck-at fault in any of the modules cannot be detected at the output, only the faults in the majority circuit are detected. So, the *fault coverage* of TMR circuits is very low. By applying our approach on the TMR circuits, which are hard-to-test, we can judge how good our heuristics are.

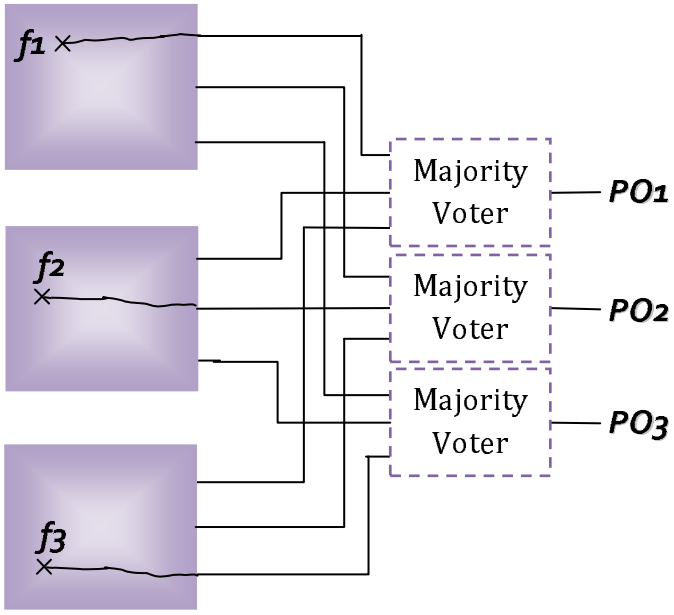


Figure 4.5: TMR Circuit

Table 4.5.3 shows the results obtained by using our approach on TMR circuits. The first column gives the circuit name. For each circuit, the table reports three sets of results - the first set is the fault coverage obtained, the second set represents the diagnostic resolution achieved and the third set is the % increase in the number of fault-pairs as compared to original. Remember that for diagnosis, we consider only the detected faults and form pairs among them. Since, the number of detected faults for TMR circuits increases considerably, the number of fault-pairs increase quadratically which is shown in the last two columns with respect to the original number of fault-pairs. For the first two sets, the results obtained on original, after test point insertion using out approach and after random test point insertion are reported. The table reveals a significant improvement in *fault coverage* using our heuristics. Particularly for $c3540$, the coverage improved from 1.39% to 82.66% as compared to random which increased it to 41.26%. The highlighting point in this case is that the diagnostic resolution also remained similar inspite of the fact that the number of pairs increased by 361369.29%. The results follow a similar trend for other circuits with an average FC improvement of 48% as compared to random which is 29%, while maintaining a similar DR even with an 29748% increase in the number of fault-pairs as compared to random which is 9741%. From the above analysis, we can conclude that the proposed heuristics indeed help in achieving a better FC and DR.

4.6 Summary

In this chapter, we developed novel heuristics to come up with a method that facilitates the selection of intelligent test points. We use the same DFT + DFD architecture described in 3.4.2 to route the TD points. The experimental results show that the test points selected by our method indeed help in increasing *fault coverage* and *diagnostic resolution*. The significant improvements obtained in hard-to-test and hard-to-diagnose circuits (TMR) further bolster our approach. Note that this method is independent of any test set which makes it easy to be incorporated in the VLSI design flow.

Table 4.1: % Indistinguished Fault-pairs left after $H1$ and $H2$

Circuit	Z-set	UR	H1	H2	Circuit	Z-set	UR	H1	H2
c432	97.52	79.41	20.81	11.97	s953	33.43	19.44	9.60	7.05
c499	81.63	80.05	25.84	25.21	s1196	44.65	27.18	10.23	7.38
c880	39.03	35.84	12.15	11.39	s1238	46.13	26.83	9.91	7.15
c1355	86.62	84.02	22.03	21.97	s1423	32.80	30.36	13.62	10.58
c1908	79.04	72.31	22.58	21.40	s1488	16.58	8.76	4.54	4.37
c2670	52.25	48.05	22.15	19.52	s1494	16.52	8.62	4.48	4.32
c3540	86.89	62.07	11.65	7.34	s5378	11.03	9.60	4.26	3.68
c5315	26.50	22.19	10.57	9.32	s9234	14.38	11.84	5.13	4.39
c6288	96.92	96.36	5.93	1.76	s13207	7.41	4.31	2.11	1.63
c7552	26.57	24.63	11.04	8.54	s15850	10.02	2.33	1.07	0.81
s298	22.87	11.78	6.39	6.28	s35932	0.49	0.18	0.09	0.09
s344	27.47	18.40	8.36	7.91	s38417	2.02	0.60	0.24	0.21
s349	26.96	18.01	8.20	7.77	s38584	0.80	0.20	0.09	0.08
s382	19.18	12.17	5.85	5.62	b02	47.62	25.35	13.89	13.59
s386	25.43	13.12	6.33	6.16	b03	22.87	17.02	8.36	7.37
s400	18.75	11.87	5.77	5.49	b04	34.25	23.07	6.77	4.71
s420	36.17	19.57	11.34	10.97	b05	28.26	21.20	7.24	6.18
s444	17.67	11.26	5.55	5.33	b06	33.47	19.58	9.78	8.56
s510	38.30	22.18	11.55	10.11	b07	42.18	35.14	14.91	10.75
s526	17.62	9.85	4.73	4.50	b08	39.60	23.08	8.57	7.61
s641	42.40	34.30	12.30	11.59	b09	41.00	29.80	13.55	12.13
s713	41.70	34.76	12.91	12.39	b10	33.45	20.62	8.55	7.99
s820	19.56	6.25	3.39	3.35	b11	39.76	14.71	6.21	5.35
s832	19.60	6.32	3.43	3.39	b12	13.30	9.26	4.08	3.51
s838	35.49	18.96	11.09	10.72	b13	9.64	6.39	3.11	2.88

Table 4.2: Results without using a test set, $H0 \leftarrow$ Random, $H3 \leftarrow Zset + UR + H1 + H2$

Circuit	# Detected Faults			# Unresolved Pairs			# Test Vectors		
	Original	$H0$	$H3$	Original	$H0$	$H3$	Original	$H0$	$H3$
c432	520	520	523	24	18	14	99	120	111
c499	750	755	752	18	9	15	129	111	117
c880	942	942	942	56	43	54	128	97	100
c1355	1566	1566	1570	793	651	627	224	181	192
c1908	1870	1870	1870	315	307	298	212	228	195
c2670	2630	2640	2678	573	363	261	210	176	128
c3540	3291	3293	3296	568	544	442	255	257	274
c5315	5291	5296	5293	464	397	373	226	176	167
c6288	7710	7710	7710	1053	1028	1017	66	70	82
c7552	7419	7452	7487	1184	974	909	396	330	346
s820	850	850	850	52	46	50	184	178	151
s832	856	859	864	65	45	47	193	179	174
s838	931	931	928	76	73	59	224	203	194
s953	1079	1079	1079	5	2	2	143	130	130
s1196	1242	1242	1242	25	19	14	224	189	162
s1238	1286	1301	1307	55	44	30	240	205	171
s1423	1501	1506	1503	157	111	108	137	117	96
s1494	1494	1438	1499	54	47	32	196	179	160
s5378	4563	4569	4573	554	439	418	358	322	263
s9234	6475	6637	6609	1362	1132	836	663	580	378
s13207	9686	9721	9748	2215	1343	897	710	590	560
s15850	11385	11434	11496	2430	1865	1300	704	596	460
s35932	35110	35754	35758	12898	8524	7787	130	97	94
s38417	31015	31034	31041	4551	2805	2064	1430	1137	662
s38584	34797	34934	35040	2768	2105	1871	1031	1012	911
b04	1666	1676	1676	106	72	68	194	163	130
b05	1928	2018	2009	1021	515	276	192	128	107
b07	1084	1090	1084	32	17	46	101	97	107
b08	451	452	452	109	58	15	103	103	90
b11	1675	1675	1658	304	287	467	182	160	147
b12	2878	2878	2877	59	30	52	264	242	248
b13	826	819	847	57	47	19	75	77	84
% Imp	-	0.63	0.81	-	29.54	39.81	-	12.40	25.27

Table 4.3: Experimental Results for TMR circuits $H3 \leftarrow Zset + UR + H1 + H2$

TMR	Fault Coverage			Diagnostic Resolution			% Increase in Pairs	
Circuit	Original	$H3$	Random	Original	$H3$	Random	$H3$	Random
c880	5.82	49.44	37.16	100.00	99.98	99.97	7677.97	4291.51
c1355	2.94	61.64	63.53	99.99	99.97	99.97	46310.44	49196.79
c1908	2.01	65.07	39.13	100.00	99.99	99.96	109603.52	39572.41
c2670	7.51	71.84	49.61	99.94	100.00	99.98	10120.14	4773.71
c3540	1.39	82.66	41.26	100.00	99.99	99.99	361369.29	89963.82
c5315	3.54	75.08	44.96	99.99	100.00	99.99	47683.89	17034.23
c7552	2.65	68.48	24.32	99.96	100.00	99.99	69325.39	8659.12
s420	6.72	30.83	23.64	99.92	99.90	99.80	2216.37	1261.39
s444	6.08	42.91	27.23	99.77	99.93	99.89	5588.12	2188.00
s526	5.40	49.25	24.17	100.00	99.97	99.89	9289.17	2159.81
s641	9.79	55.39	48.93	99.83	99.96	99.95	3698.02	2864.22
s713	8.30	50.54	46.67	99.83	100.00	99.92	4184.64	3553.48
s820	3.45	37.85	26.27	99.77	99.98	99.96	12959.85	6189.43
s832	3.37	39.08	26.04	99.79	99.97	99.96	14416.15	6343.03
s838	6.41	31.32	23.77	99.96	99.95	99.91	2507.48	1402.31
s953	5.13	74.36	35.16	99.75	100.00	99.94	23435.05	5160.97
s1196	3.18	61.23	35.40	99.94	99.99	99.96	39603.13	13165.82
s1238	2.94	60.07	29.73	99.95	99.99	99.93	44496.19	10822.47
s1423	6.51	53.54	42.76	99.98	99.99	99.97	7529.87	4767.08
s1488	1.67	44.30	19.78	100.00	99.99	99.96	74414.49	14753.15
s1494	1.65	45.96	21.03	100.00	99.99	99.96	82142.86	17104.00
s5378	5.49	70.85	32.88	99.98	100.00	99.99	18499.14	3905.83
s9234	4.30	61.11	29.46	99.98	100.00	99.99	21927.48	5018.42
b04	5.43	59.60	47.97	100.00	100.00	99.98	13287.64	8570.94
b05	2.61	52.31	41.96	100.00	99.97	99.96	43043.20	27659.38
b06	8.92	53.81	39.97	99.10	99.88	99.73	4401.13	2380.84
b07	5.79	58.73	44.59	99.95	99.99	99.98	11550.91	6614.82
b08	7.06	59.19	44.36	99.98	99.94	99.82	7934.55	4411.76
b09	8.15	53.51	43.96	99.82	100.00	99.94	4983.60	3330.67
b10	6.01	60.73	33.30	100.00	99.97	99.89	11358.77	3342.24
b11	2.72	62.24	21.72	99.98	99.98	99.91	55385.29	6652.63
b12	5.29	59.02	35.22	99.94	100.00	99.99	13738.24	4827.74
b13	8.26	45.14	32.46	99.95	99.97	99.95	3412.28	1715.14

Chapter 5

Enhancing Diagnosability of BIST Architectures

5.1 Motivation

BIST architectures are often employed in digital circuitries over external testing because

- They enable at-speed testing to give a high throughput.
- They give a high defect coverage. They are able to detect faults which were not specifically targeted.
- The architecture can be reused when the circuit is in field [Wunderlich].
- Low cost testers can be used instead of expensive ATEs for application of test vectors.

Despite several advantages, BIST architectures suffer from very low diagnostic ability. This is largely because of the output response analyzer which compacts the output responses obtained to a single signature. This signature is sufficient enough to tell whether the chip is

faulty or not, but it does not give any information about the fault location i.e. it does not aid diagnosis. This chapter focuses on improving the diagnostic resolution of BIST architectures.

5.2 Problem Statement

Given a full scan circuit C with BIST architecture, develop an approach which improves the diagnosability with minimum area overhead and processing.

5.3 Related Work

The diagnosability enhancement of BIST architectures started with [Aitken, Zorian] in which a modification of the BIST architecture was proposed to incorporate the diagnosis data as shown in the Figure 5.1. As shown, additional components - Response Generation and Fail Memory are added to the architecture. Response Generation is typically a ROM or PLA which generates responses in accordance with the input test vectors. These responses act as reference responses and are compared with the actual responses obtained from the chip. If the responses mismatch, the corresponding actual response is stored in the fail memory. The responses stored in the fail memory are then used to identify the actual fault by using a separate diagnosis approach.

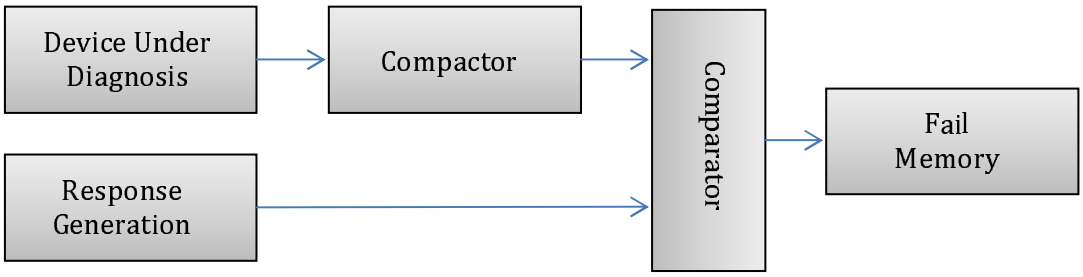


Figure 5.1: Modified BIST Circuit for Aiding Diagnosis

Later, two phase tests were developed for diagnosis. The first phase would screen out the defective chips which subsequently would be tested again with different scan chains or different scan cells [Bayraktaroglu], with different test patterns [Wohl] or with different response compactor [Leininger]. Next, multiple signatures were used to get more diagnostic data either by observing the storage elements data [Rajski] or by using different test patterns [Cheng]. In [Elm], the authors proposed a novel Built-in Self Diagnosis (BISD) architecture, a slight modification of the BIST model as shown in Figure 5.2. It compacts the data on-chip using space compaction by XORing the outputs of scan chains and time compaction by feeding the space compacted outputs to a single input signature register (SISR). It also needs a modification to the ATPG algorithm to generate deterministic patterns. All of these methods somehow or the other needed a response memory so as to store the failing responses into the fail memory.

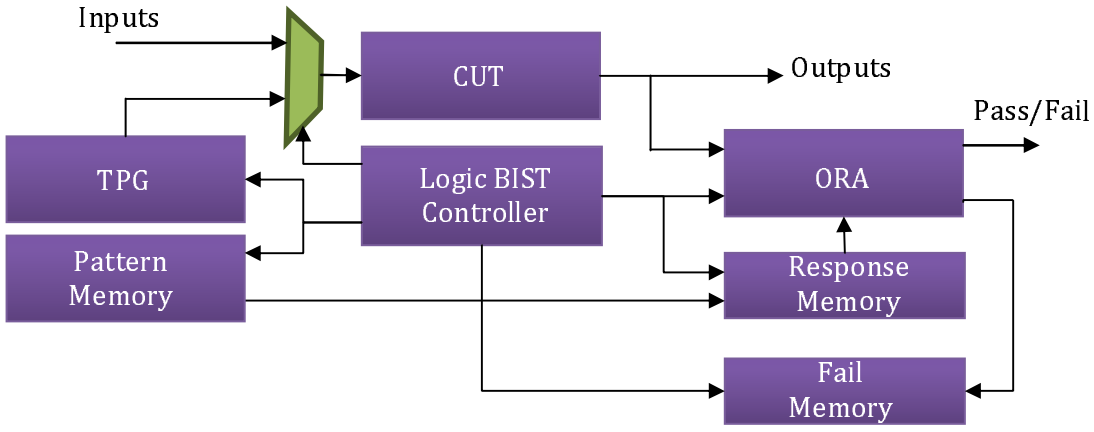


Figure 5.2: BISD Architecture proposed in [Elm]

5.4 Key Idea

We propose the following modifications on the BISD architecture proposed in [Elm].

5.4.1 Eliminating Response Memory

[Elm] uses a response memory, usually a ROM or PLA, to store the responses of each and every test pattern generated by the TPG. The compare unit is used to compare this reference response to the actual response obtained from the circuit. If a mismatch is found, the corresponding incorrect response along with the test pattern index is stored into the fail memory. The major factor that increases the area overhead for this procedure is the response memory block. Removing this block altogether will result in significant savings in area. This will also eliminate the need of comparator giving additional area savings.

If this is done, then there are no reference signatures present to compare the actual output responses of the circuit in which case we will not be able to decide what all responses are incorrect that are needed to be stored in the fail memory. One way to get around this would be to store each and every response and run a subsequent diagnosis procedure on the collected data. But this will require the memory size to be the same as the response memory which will not save any area and also the subsequent diagnosis procedure would mean additional test session. From this analysis, we can conclude that if the response memory has to be eliminated, the output responses of selected small subset of test patterns must be stored.

Now to select a small subset, it is important that the selected vectors be of high quality in the sense that they should not compromise either on detectability or diagnosability. So, in other words, the FC and DR achieved by the selected test set should be equal to that obtained by the whole test set. We can achieve this by using test set compaction methods.

Experiments carried out on benchmark circuits using this criterion show a reduction in the test set size by 35% as shown in Section 5.6. So, this indicates that responses for 65% of the test vectors need to be stored which would mean increase in the size of the on-chip memory. In order to avoid this, we can target to select the subset such that only the FC obtained is same as the original set. Experiments conducted using this criterion leaves us with a test set size of around 5% which is quite small and acceptable as shown in Section 5.6. The diagnostic resolution decreases in this case which is expected because of decrease

in the number of vectors. Now, we can use our approach developed in Chapter 3 to select test points using the WA metric and insert them using the architecture proposed in Section 5.5.2.

In summary, we use a memory that stores the output responses of a few selected test vectors $t \subset T$. These test vectors are selected such that they give the same FC as achievable by the original test set. The DR of t could be equal to or less than that of T. To improve the diagnostic resolution of t, we use our approach described before in Chapter 3 by computing $nodeCount_i$ and selecting the TD points. In this case, since we just need to improve upon DR, we perform DFD by setting α to 0 in Equation 3.3

5.4.2 Eliminating Diagnosis on Fail Memory

The authors in [Elm] propose that once the test session has finished, the data collected in the fail memory is downloaded to an external tester where the data is subjected to a diagnosis procedure. This has to be performed since the responses downloaded to the fail memory are the compacted signatures and the test pattern index which need additional processing to diagnose and decode respectively. This processing can be saved if the output responses of the primary outputs are stored instead of the lossy signatures and the pattern index. Since the number of primary outputs can be large, it is important to store responses of very few test patterns so that on-chip memory does not eat up a lot of area. Since we select only the subset of test vectors that have the same FC as that of original set, we can do the above without incurring a severe area penalty.

5.4.3 Eliminating Extreme Compaction

The proposed extreme compaction in [Elm] uses both space and time compaction technique and is named XP-SISR. The data in different scan chains is space compacted to generate a stream of data which is then time compacted using single input signature register (SISR)

to generate the signature. Such a scheme compresses the output responses to generate a lossy signature. In order to reduce aliasing of signatures, some additional deterministic patterns are added on-chip which are generated by using the modified ATPG. Since, our above discussion concludes that only few test responses are sufficient to aid testing and diagnosis, we can remove the compaction scheme and store the primary outputs along with the storage elements states on the embedded memory. Removing the compaction scheme reduces the area further and also cancels out the aliasing issue. This also eliminates the need to modify ATPG performed in the original BISD architecture.

5.5 Approach

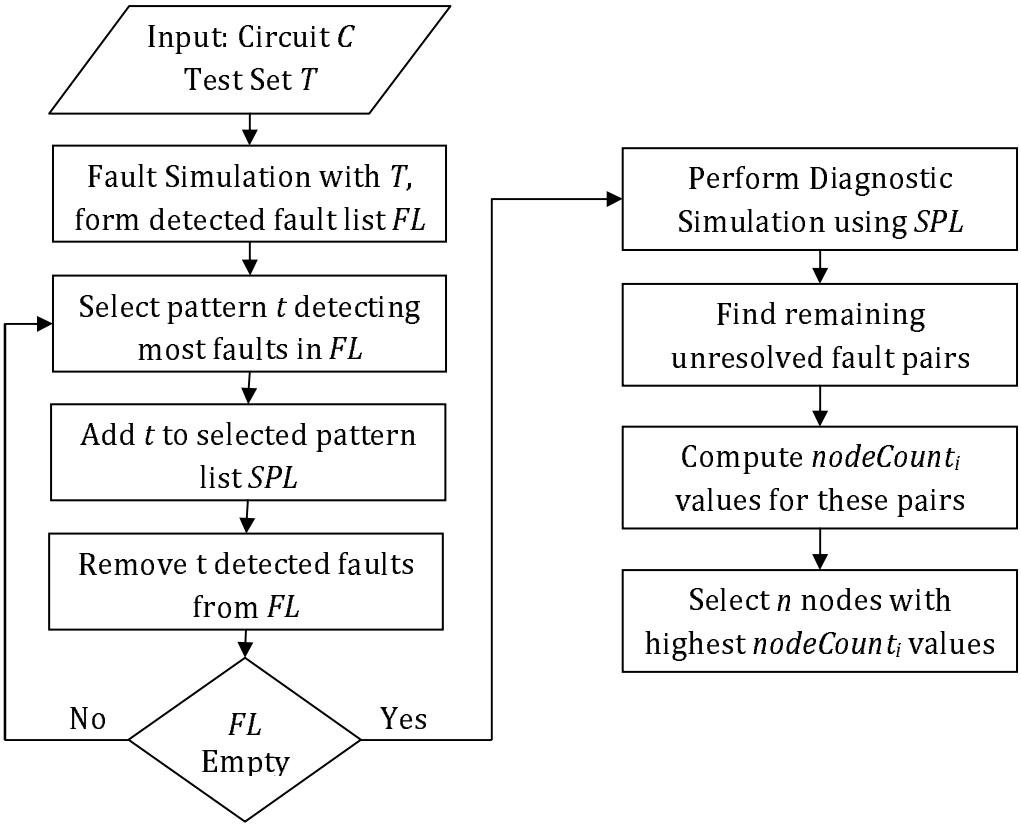


Figure 5.3: Overall Flow for our BISD architecture

5.5.1 Overall Flow

Figure 5.3 shows the overall flow of the method. We start off with the fault simulation of the available test vectors. Next we select the test vector which detects the most number of faults. The faults detected by this vector are then removed from the fault list and the next vector which detects the most faults from the remaining list is chosen. This process is repeated until the fault list is empty. This gives us a small subset of test vectors that give the same FC as obtained by using all the test vectors. In other words, a greedy algorithm is used to compact the test set. Now, we need to store the output responses of only these selected test vectors. So, this step decides the size of the on-chip memory to be used. The next step is to perform diagnostic simulation using the selected vectors to find out the unresolved pairs. Also, during this step, the $nodeCount_i$ values are computed for the identified unresolved pairs. The $nodeCount_i$ values are arranged in descending order and n nodes with highest values are selected as test points. The value of n is set to be equal to the number of primary outputs in the circuit.

5.5.2 BISD Architecture

Figure 5.4 shows the architecture that we propose for improving diagnosis of BIST. In addition to the components required in BIST, we just add an extra block of memory. This memory will store the output responses of the circuit for a small subset of test patterns generated by the TPG. Since this memory will be embedded on-chip, it is important that the size of the memory should be small enough not to cause a huge area overhead and at the same time, should be sufficient enough for diagnosis. Since, we would like to store the output responses of the circuit, the size of each row in the memory would be equal to the number of primary outputs in the circuit. If $t \subset T$ are the test vectors that are chosen and p are the number of primary outputs of the circuit, then the size of memory would be $t \times p$. To insert the test points, we employ the same $NOR + XOR$ logic described in Section 3.4.2.

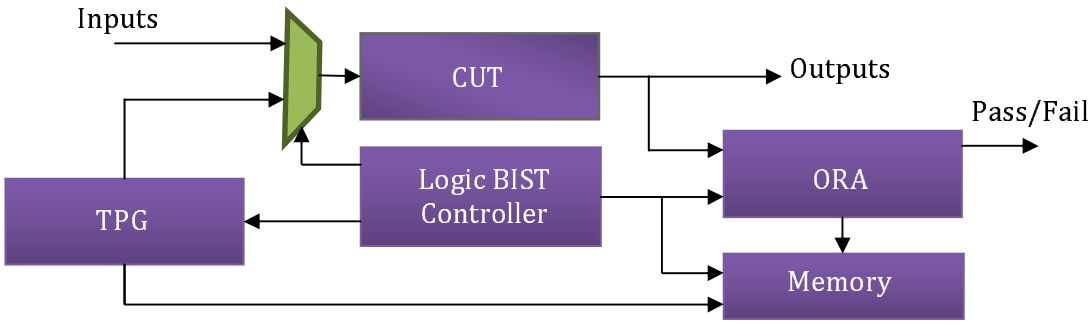


Figure 5.4: Proposed BISD Architecture

5.6 Experimental Results

The proposed technique was implemented in C++ and experiments were performed on Red Hat Linux workstations with Intel Pentium D 3.0GHz CPU, 8GB memory and 4 cores. We used ISCAS85 and full scan versions of ISCAS89 and ITC99 benchmark circuits for carrying out the experiments. We assumed a single stuck-at-fault model. The test set T used was generated by a random function to depict the characteristics of LFSR. We conduct the experiments in two sets. Set I shows the results for compacted test set $t1$ obtained with the criterion that FC and DR achieved by $t1$ is equal to that of T. Set II shows results for compacted test set $t2$ obtained with the criterion that $t2$ and T have the same *fault coverage*.

Table 5.1 shows the result for our approach. The first column is the circuit used followed by the number of faults detected in the second column. The third column represents the number of test vectors generated by the LFSR to test the circuit. The next column has the number of unresolved fault-pairs left if all the responses of all the test vectors would be stored in the fail memory. The subsequent columns contain the number and percentage of selected test vectors that detect the same number of faults as before. The number of fault pairs remaining after selecting these vectors is shown in the next column. The following column denotes the number of unresolved pairs remaining after adding test points. The test points are selected here based on the selected vectors present in fifth column. The next two

columns represent the number and percentage of selected test vectors that give the same FC and DR as before.

The table shows promising results. Using a small subset of available patterns and adding test points, a good improvement in *diagnostic resolution* is seen. Note that the number of vectors are selected based on the greedy algorithm as explained in Flowchart 5.5.1. The algorithm makes sure that the *fault coverage* remains same for the selected vectors. So, the column # Flts representing the faults detected remains the same throughout. The number of unresolved pairs increase after selecting a subset of vectors as expected. After adding test points based on our procedure, we are able to decrease this number. Especially for circuit $c2670$, using a subset of 1.82% vectors, we were able to reduce the number of unresolved pairs from 733 to just 55. Some other circuits like $c3540$, $s1423$, $b05$, $b07$, $b13$ follow similar trend. On an average, by using just 5% of the total test patterns generated by the TPG, we were able to achieve an excellent diagnostic resolution. So, the on-chip memory is required to store the responses of these selected patterns only. Also, by adding test points by using our architecture, we further improved the DR by around 60%. These results show that by using our proposed architecture and the selected test points, the BIST architecture now helps diagnosis. The new architecture not only will term a chip as faulty through signature comparison but also give enough output response data to narrow in on the fault location in the circuit.

Also as evident from the numbers in the last column, the percentage of patterns needed for keeping FC and DR same as before is large. Circuits $c880$, $c3540$, $s1196$, $b12$ need more than 85% of the available test vectors to achieve the same original result. The outcomes are similar for other benchmark circuits too with an average of 65% of test vectors required. If we store the output response of each of these test patterns, the size of memory needed would end up being high and would defeat our purpose of saving the on-chip area overhead. So, the criterion of maintaining same FC and DR as original set is not used for our approach.

Table 5.1: Results for Selected Patterns with same FC and with same $FC + DR$

Circuit	# Flts	Original		Same FC			# Pairs after test points	Same $FC + DR$	
		# Vec	# Pairs	# Vec	% Vec	# Pairs		# Vec	% Vec
c432	520	524	28	46	8.78	105	66	353	67.37
c499	750	758	12	52	6.86	26	0	373	49.21
c880	931	942	57	45	4.78	86	15	861	91.40
c1355	1561	1574	747	82	5.21	763	599	765	48.60
c1908	1838	1879	375	102	5.43	435	265	624	33.21
c2670	2316	2747	445	50	1.82	733	55	1953	71.10
c3540	3284	3428	541	129	3.76	624	342	2981	86.96
s298	305	308	21	32	10.39	28	2	215	69.81
s344	342	342	5	17	4.97	35	4	170	49.71
s349	348	350	10	18	5.14	41	3	179	51.14
s382	399	399	23	29	7.27	38	1	185	46.37
s400	420	426	46	29	6.81	61	5	190	44.60
s420	324	455	113	30	6.59	125	47	145	31.87
s510	560	564	3	56	9.93	22	0	329	58.33
s526	520	555	57	48	8.65	84	19	385	69.37
s641	436	467	15	38	8.14	19	2	357	76.45
s713	514	581	178	42	7.23	184	60	387	66.61
s953	905	1079	60	60	5.56	155	18	506	46.90
s1196	1148	1242	75	105	8.45	114	19	1077	86.71
s1238	1176	1355	106	110	8.12	192	38	1116	82.36
s1423	1481	1515	170	52	3.43	215	8	1071	70.69
s1494	1467	1506	38	108	7.17	94	16	683	45.35
b05	1886	2470	1116	64	2.59	1226	132	1131	45.79
b06	140	140	2	14	10.00	8	1	64	45.71
b07	1019	1090	85	35	3.21	144	38	684	62.75
b08	399	452	89	29	6.42	259	4	192	42.48
b10	513	517	21	43	8.32	50	2	378	73.11
b11	1624	1740	311	72	4.14	427	150	1116	64.14
b12	2723	2878	121	117	4.07	157	33	2585	89.82
b13	810	852	53	28	3.29	73	5	681	79.93
AVG	-	-	-	-	5.08	-	60.41	-	65.60

5.7 Summary

In this chapter, we proposed a new way of improving the diagnostic resolution of BIST architectures. By storing the output responses of only a small subset of test patterns, the size of fail memory is reduced. The other area overhead would be the $NOR + XOR$ logic which is shown to be quite less. The big advantage is the absence of response memory which saves a lot of area. Experimental results show that using just a subset of around 5% of test patterns, quality diagnosis data can be obtained.

Chapter 6

Conclusions and Future Work

In this thesis, we targeted the problem of synergistic enhancement of *fault coverage* and *diagnostic resolution* of a digital circuit. The test point insertion for DFT as well as DFD was utilized with focus on selecting intelligent observe points called as Testability-Diagnosability (TD) points. We also developed a new DFT + DFD architecture to insert the selected test points into the circuit. Logic BIST architectures were also studied with the aim to increase their *diagnostic resolution*. This was achieved by modifying the BIST architecture and is called as BISD architecture.

Chapter 1 gave a general overview of the problem statement and motivation for solving this problem. It also highlighted the contributions of the thesis.

Chapter 2 introduced the preliminaries and concepts used in the thesis. The various techniques used for Testability Analysis, Design for Testability and Design for Diagnosis were also described. An overview of existing BIST architecture was outlined. It also gave a summary about the background work ongoing in this domain.

In *Chapter* 3, we proposed a novel low-cost method and a new DFT + DFD architecture to increase *fault coverage* as well as *diagnostic resolution* by adding appropriate *TD* points. Previously, test points were added only from the testability point of view, while our method

considers both testability and diagnosability. An efficient method for *TD* points selection has been described with an architecture that does not add any output pins. The experimental results indicate a 4× improvement in *diagnostic resolution* while maintaining the same *fault coverage*. Also, the area overhead was around 5% which is relatively small considering the benefits obtained. The method of finding test points using our metric is computationally inexpensive. Once the fault and diagnostic simulation are done, the computation of TD points takes less than 0.1 sec. The routing issues for our architecture would be mainly due to *TD Enable* pin. However, the issues will be less significant as compared to the routing issues for other DFT methodologies.

Chapter 4 describes an approach for test point insertion using just the structural characteristics of the circuit. Two heuristics were proposed to come up with the test points. The significant improvements achieved by using our approach on hard-to-test circuits bolstered the heuristics developed. Since the method is independent of any test set, simulation time is saved and makes the method easily portable. The computation of heuristics is linear to the size of the circuit and the computation complexity of test points remains same as before. For benchmark circuits, the average improvement in FC was 0.81%, in DR was 39.81% with a test set reduction of 25.27%. Implementing this method on some hard-to-test circuits increased FC by 48% maintaining a similar DR even though the number of fault pairs increased by a staggering 29748%.

In the future, the combination of *TD* points selection and partial scan, rather than full scan, can be explored further to improve the utilization of the approach. Also the number of test points added to the circuit can be increased by inserting an additional XOR gate in the $NOR + XOR$ logic as shown in the Figure 6.1. The more the test points, the more the benefit achieved in FC and DR. One can incrementally add the XOR gates to form a XOR tree as long as the area overhead incurred is acceptable.

Chapter 5 talked about the methods to enhance the diagnosability of BIST architectures. BIST architectures are known for poor DR. We targeted this problem to come up with

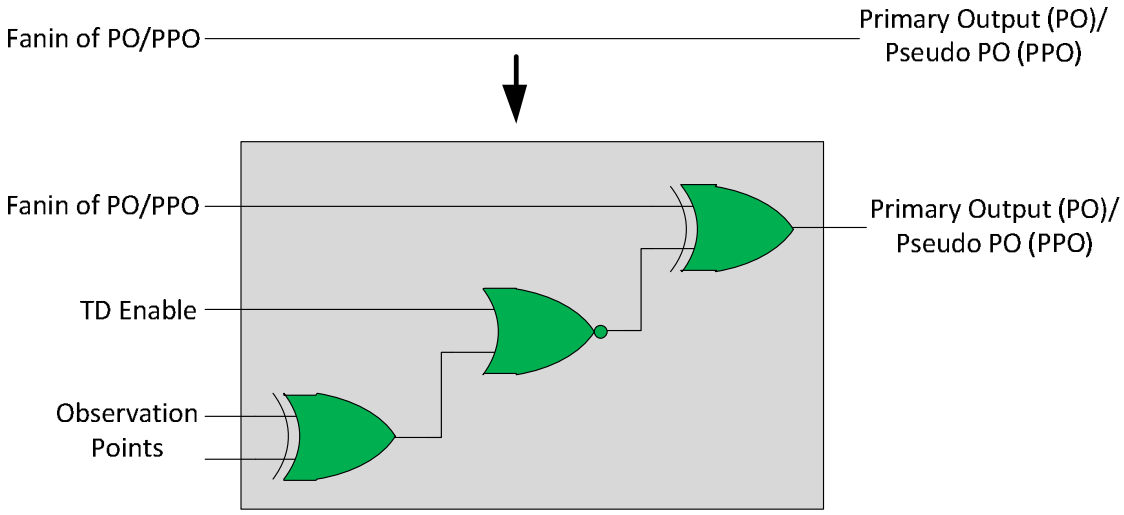


Figure 6.1: Architecture for adding 2× Test Points

an architecture that needs minimal on-chip memory to store the output responses of the circuit for some specific small subset of test vectors. Experimental results indicate that implementing our architecture and by adding appropriate test points, the DR increased by 60% as compared to DR achievable by external testing using just 5% of the total test vectors.

In future, for the proposed architecture in BIST, instead of storing the responses of all the primary outputs, a subset of outputs can be chosen. This way one can further reduce the on-chip memory size required and save area. It will be interesting if we can constrain the faults to be detected only at a subset of outputs and use our proposed architecture to see the benefits obtained. Even if some deterministic patterns are added to detect some additional faults, our approach can be applied again to come up with a reduced compact test set which will help diagnose the faults.

Bibliography

[Aitken] R. Aitken and V. Agarwal, "A Diagnosis Method Using Pseudo-Random Vectors Without Intermediate Signatures," *Digest of Technical Papers of the IEEE International Conference on Computer-Aided Design*, pp. 574-577, 1989

[Balakrishnan] K. J. Balakrishnan and L. Fang, "RTL Test Point Insertion to Reduce Delay Test Volume," *VLSI Test Symposium*, pp. 325-332, 2007

[Bayraktaroglu] I. Bayraktaroglu and A. Orailoglu, "Cost-effective deterministic partitioning for Rapid Diagnosis in Scan-based BIST," *IEEE Design and Test of Computers*, vol. 19, pp.42-53, Jan/Feb 2002

[Chandrasekar] M. Chandrasekar and M. Hsiao, "Diagnostic Test Generation for Silicon Diagnosis with an Incremental Learning Framework based on Search State Compatibility," *High Level Design Validation and Test Workshop*, pp. 68-75, 2009

[Cheng] W.-T. Cheng, M. Sharma, T. Rinderknecht, L. Lai, and C. Hill, "Signature Based Diagnosis for Logic BIST," *International Test Conference*, pp. 1-9, 2006

[Elm] M. Elm and H.-J. Wunderlich, "BISD: Scan-Based Built-In Self-Diagnosis," *Design, Automation and Test in Europe Conference and Exhibition*, pp. 1243-1248, 2010

[Fang] L. Fang and M. S. Hsiao, "Bilateral Testing of Nano-scale Fault-tolerant Circuits," *21st IEEE Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 309-317, 2006

[Geuzebroek 00] M. J. Geuzebroek, J.T. van der Linden, and A. J. van de Goor, "Test Point Insertion for Compact Test Sets," *International Test Conference*, pp. 292-301, 2000

[Geuzebroek 03] M. Geuzebroek and A. J. van de Goor, "TPI for PR Fault Coverage of Boolean and Three-State Circuits," *IEEE European Test Workshop*, pp. 3-8, 2003

[Goldstein] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia Controllability/Observability Analysis Program," *Design Automation Conference*, pp. 190-196, 1980

[Hamzaoglu] I. Hamzaoglu and J. H. Patel, "New Techniques for Deterministic Test Pattern Generation," *In Proceedings of 16th IEEE VLSI Test Symposium*, pp. 446-452, 1998

[Hayes Detection] J. Hayes, and A. D. Friedman, "Test Point Placement to Simplify Fault Detection," *IEEE Transactions on Computers*, vol C-23, no. 7, pp. 727-735, 1974

[Hayes Diagnosability] J. Hayes, "On Modifying Logic Networks to Improve Their Diagnosability," *IEEE Transactions On Computers*, vol. C-23, no. 1, pp. 56-62, Jan 1974

[Hsiao] M. S. Hsiao and M. Banga, "Kiss the Scan Goodbye: A Non-Scan Architecture for High Coverage, Low Test Data Volume, and Low Test Application Time," *Asian Test Symposium*, pp. 225-230, 2009

[Iyer] M. A. Iyer and M. Abramovici, "FIRE: A Fault Independent Combinational Redundancy Identification Algorithm," *IEEE Transactions on VLSI Systems*, vol. 4, no. 2, pp. 295-301, June 1996

[Krishnamurthy] B. Krishnamurthy, "A Dynamic Programming Approach to the Test Point Insertion Problem," *Design Automation Conference*, pp. 695-704, Jun 1987

[Leininger] A. Leininger, M. Goessel, and P. Muhmenthaler, "Diagnosis of Scan-chains by Use of a Configurable Signature Register and Error- correcting Codes," *Design, Automation and Test in Europe*, vol. 2, pp. 1302-1307, 2004

[Lin] C-C. Lin, M. M.-Sadowska, K-T. Cheng, and M.T.-C. Lee, "Test-Point Insertion: Scan Paths Through Functional Logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, no. 9, pp. 838-851, Sep 1998

[Marinissen] E. J. Marinissen, and N. Nicolici, "Editorial: Silicon Debug and Diagnosis," *IET Computers and Digital Techniques*, vol. 1, no. 6, pp. 659-660, Nov 2007

[Nakao] M. Nakao, S. Kobayashi, K. Hatayama, K. Iijima, and S. Terada, "Low Overhead Test Point Insertion For Scan-based BIST," *In Proceedings of International Test Conference*, pp. 348-357, 1999

[Pomeranz] I. Pomeranz, S. Venkataraman, S. M. Reddy, and B. Seshadri, "Z-sets and Z-detections: Circuit Characteristics that Simplify Fault Diagnosis," *Design, Automation and Test in Europe*, pp. 68-73, Mar 2004

[Rajski] J. Rajski, J. Tyszer, C. Wang, and S. Reddy, "Finite Memory Test Response Compactors for Embedded Test Applications," *IEEE Transactions on computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 622-634, 2005

[Ravikumar] C. P. Ravikumar, M. Sharma, and R. K. Patney, "Improving the Diagnosibility of Digital Circuits," *In Proceedings of 12th International Conference on VLSI Design*, pp. 629-634, 1999

[Reddy] S. M. Reddy, "Easily Testable Realizations for Logic Functions," *IEEE Transactions on Computers*, vol. C-21, no. 11, pp. 1183-1188, Nov 1972

[Remersaro] S. Remersaro, J. Rajski, T. Rinderknecht, S. M. Reddy, and I. Pomeranz, "ATPG Heuristics Dependent Observation Point Insertion for Enhanced Compaction and Data Volume Reduction," *IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, pp. 385-393, 2008

[Sankaralingam] R. Sankaralingam and N. A. Touba, "Inserting Test Points to Control Peak Power During Scan Testing," *In Proceedings of 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 138-146, 2002

[Seiss] B. Seiss, P. Trouborst, and M. Schalz, "Test Point Insertion for Scan-based BIST," *In Proceedings of the European Test Conference*, pp. 253-262, Apr 1991

[Sethuram] R. Sethuram, S. Wang, S. T. Chakradhar, and M. L. Bushnell, "Zero Cost Test Point Insertion Technique to Reduce Test Set Size and Test Generation Time for Structured ASICs," *Asian Test Symposium*, pp. 339-348, 2006

[SWang] S. Wang and S. T. Chakradhar, "A scalable scan-path test point insertion technique to enhance delay fault coverage for standard scan designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 8, pp. 1555-1564, 2006

[Synopsys] http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Pages/ DCUltra.aspx

[Tamarapalli] N. Tamarapalli and J. Rajski, "Constructive Multi Phase Test Point Insertion for Scan-based BIST," *International Test Conference*, pp. 649-658, 1996

[Touba] N. A. Touba and E. J. McCluskey, "Test Point Insertion Based on Path Tracing," *In Proceedings of IEEE VLSI Test Symposium*, pp. 2-8, 1996

[Venkataraman] S. Venkataraman, I. Hartanto, W. Fuchs, E. Rudnick, S. Chakravarty, and J. Patel, "Rapid Diagnostic Fault Simulation of Stuck-at Faults in Sequential Circuits using Compact Lists," *Design Automation Conference*, pp.133-138, 1995

[Vimjam] V. C. Vimjam and M. S. Hsiao, "Efficient Fault Collapsing via Generalized Dominance Relations," *IEEE VLSI Test Symposium*, pp. 258-265, 2006

[VTVT] http://www.vtvt.ece.vt.edu/vlsidesign/cell.php

[Wang SOC] L-T. Wang, C. E. Stroud, and N. A. Touba, "System on Chip Test Architectures," Morgan Kaufmann Publishers, 2008

[Wang VLSI] L-T. Wang, C-W. Wu, and X. Wen, "VLSI Test Principles and Architectures," Morgan Kaufmann Publishers, 2006

[Wohl] P. Wohl, J. Waicukauski, S. Patel, and G. Matson, "Effective Diagnostics Through Interval Unloads in a BIST Environment," *Design Automation Conference*, pp. 249-254, 2002

[Wunderlich] H. Wunderlich, "BIST for System-on-a-Chip," *INTEGRATION, The VLSI Journal*, vol. 26, no. 1-2, pp. 55-78, 1998

[Yoshimura] M. Yoshimura, T. Hosokawa, and M. Ohta, "A Test Point Insertion Method to Reduce the Number of Test Patterns," *In Proceedings of 11th IEEE Asian Test Symposium*, pp. 298-304, 2002

[Zorian] Y. Zorian and V. Agarwal, "A General Scheme to Optimize Error Masking in Built-in Self-testing," *In Proceedings of 16th International Symposium on Fault-Tolerant Computing Systems*, pp. 410-415, July 1986

Appendix A

Why XOR Gate?

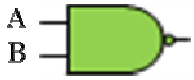
Figure A.1 shows the truth tables for three 2-input gates: XOR, NAND and NOR. For each table, we have shown three columns. The first two columns represent the logic values for the two inputs (A and B) to the gate considered. The third column is the output logic value obtained for the logic values applied at the gate's inputs.

We only consider the cases where there is at least one fault effect (D or $\overline{D}$) on any one of the inputs of the gates. Out of the 32 combinations of $\{0, 1, X, D, \overline{D}\}$ logic values for 2-input gates, we are left with 16 cases to consider.

For the XOR gate, we can see that at the output, the fault effect is observed for 8 cases in total. The NAND gate propagates the fault effect at its output for a total of 6 cases. Similarly, the fault effect is observed at NOR gate's output for 6 cases in all. From this, it can be concluded that the probability of propagation of fault effect is more for XOR gate as compared to NAND and NOR gates. Since NAND and NOR are universal gates, all the primitive gates have less probability of fault effect propagation as compared to XOR.

So, we have used XOR gate in our approach so as to exploit the aforementioned benefit.

Input A	Input B	Output
0	D	D
0	$\overline{D}$	$\overline{D}$
1	D	$\overline{D}$
1	$\overline{D}$	D
D	0	D
$\overline{D}$	0	$\overline{D}$
D	1	$\overline{D}$
$\overline{D}$	1	D
$\overline{D}$	$\overline{D}$	0
D	D	0
$\overline{D}$	D	1
D	$\overline{D}$	1
D	X	X
$\overline{D}$	X	X
X	D	X
X	$\overline{D}$	X



Input A	Input B	Output
0	D	1
0	$\overline{D}$	1
1	D	$\overline{D}$
1	$\overline{D}$	D
D	0	1
$\overline{D}$	0	1
D	1	$\overline{D}$
$\overline{D}$	1	D
$\overline{D}$	$\overline{D}$	D
D	D	$\overline{D}$
$\overline{D}$	D	1
D	$\overline{D}$	1
D	X	X
$\overline{D}$	X	X
X	D	X
X	$\overline{D}$	X

Input A	Input B	Output
0	D	D
0	$\overline{D}$	$\overline{D}$
1	D	0
1	$\overline{D}$	0
D	0	$\overline{D}$
$\overline{D}$	0	D
D	1	0
$\overline{D}$	1	0
$\overline{D}$	$\overline{D}$	D
D	D	$\overline{D}$
$\overline{D}$	D	0
D	$\overline{D}$	0
D	X	X
$\overline{D}$	X	X
X	D	X
X	$\overline{D}$	X

Figure A.1: Truth Tables for XOR, NAND and NOR gates