2008

On low power test and DFT techniques for test set compaction

Santiago Remersaro
University of Iowa

ON LOW POWER TEST AND DFT TECHNIQUES FOR TEST SET COMPACTION

by

Santiago Remersaro

An Abstract

Of a thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Electrical and Computer Engineering in the
Graduate College of The
University of Iowa

December 2008

Thesis Supervisor: Professor Sudhakar M. Reddy

ABSTRACT

The objective of manufacturing test is to separate the faulty circuits from the good circuits after they have been manufactured. Three problems encompassed by this task will be mentioned here.

First, the reduction of the power consumed during test. The behavior of the circuit during test is modified due to scan insertion and other testing techniques. Due to this, the power consumed during test can be abnormally large, up to several times the power consumed during functional mode. This can result in a good circuit to fail the test or to be damaged due to heating.

Second, how to modify the design so that it is easily testable. Since not every possible digital circuit can be tested properly it is necessary to modify the design to alter its behavior during test. This modification should not alter the functional behavior of the circuit. An example of this is test point insertion, a technique aimed at reducing test time and decreasing the number of faulty circuits that pass the test.

Third, the creation of a test set for a given design that will both properly accomplish the task and require the least amount of time possible to be applied. The precision in separation of faulty circuits from good circuits depends on the application for which the circuit is intended and, if possible, must be maximized. The test application time is should be as low as possible to reduce test cost.

This dissertation contributes to the discipline of manufacturing test and will encompass advances in the afore mentioned areas. First, a method to reduce the power consumed during test is proposed. Second, in the design modification area, a new algorithm to compute test points is proposed. Third, in the test set creation area, a new algorithm to reduce test set application time is introduced. The three algorithms are scalable to current industrial design sizes. Experimental results for

the three methods show their effectiveness.

Abstract Approved: _____
Thesis Supervisor

_____
Title and Department

_____
Date

ON LOW POWER TEST AND DFT TECHNIQUES FOR TEST SET COMPACTION

by

Santiago Remersaro

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Electrical and Computer Engineering in the
Graduate College of The
University of Iowa

December 2008

Thesis Supervisor: Professor Sudhakar M. Reddy

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Santiago Remersaro

has been approved by the Examining Committee
for the thesis requirement for the Doctor of
Philosophy degree in Electrical and Computer Engineering
at the December 2008 graduation.

Thesis Committee: Sudhakar M. Reddy, Thesis Supervisor

Jon G. Kuhl

Zhiqiang Liu

John P. Robinson

Kasturi R. Varadarajan

Janusz Rajski

To My Family

ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Sudhakar M. Reddy and my supervisor, Dr. Janusz Rajski for their support and guidance throughout the years needed to complete this work. I am also grateful to my committee members, their questions during the different evaluations encouraged me to keep thinking about these research topics and gave me other viewpoints to consider.

Also, I want to acknowledge the Semiconductor Research Corporation and Mentor Graphics Corporation for the financial support needed for this research.

I feel the need to also mention the Mentor DFT team and thank them for allowing me to use and modify their tools for my purposes, for explaining the ins and outs of the tools, and for showing interest in my work through several fruitful discussions.

Finally, I would like to thank my family and friends for giving me company. Without them, this would have been a very lonely journey.

ABSTRACT

The objective of manufacturing test is to separate the faulty circuits from the good circuits after they have been manufactured. Three problems encompassed by this task will be mentioned here.

First, the reduction of the power consumed during test. The behavior of the circuit during test is modified due to scan insertion and other testing techniques. Due to this, the power consumed during test can be abnormally large, up to several times the power consumed during functional mode. This can result in a good circuit to fail the test or to be damaged due to heating.

Second, how to modify the design so that it is easily testable. Since not every possible digital circuit can be tested properly it is necessary to modify the design to alter its behavior during test. This modification should not alter the functional behavior of the circuit. An example of this is test point insertion, a technique aimed at reducing test time and decreasing the number of faulty circuits that pass the test.

Third, the creation of a test set for a given design that will both properly accomplish the task and require the least amount of time possible to be applied. The precision in separation of faulty circuits from good circuits depends on the application for which the circuit is intended and, if possible, must be maximized. The test application time is should be as low as possible to reduce test cost.

This dissertation contributes to the discipline of manufacturing test and will encompass advances in the afore mentioned areas. First, a method to reduce the power consumed during test is proposed. Second, in the design modification area, a new algorithm to compute test points is proposed. Third, in the test set creation area, a new algorithm to reduce test set application time is introduced. The three algorithms are scalable to current industrial design sizes. Experimental results for

the three methods show their effectiveness.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

This document is organized as follows. This chapter outlines the purpose of this work and gives a review of different Design for Test (DFT) techniques. Chapter 2 contains the proposed method for generating low power test sets both as part of ATPG (Automatic Test Pattern Generator) and as a post-processing step on a previously created test set. Chapter 3 presents the proposed algorithm for test point insertion that finds observation point locations. Chapter 4 introduces the ATPG heuristics that achieve small test sets in a scalable manner. Finally, Chapter 5 reviews and concludes this thesis.

1.1 Motivation

Every year and a half, according to Moore's law, the transistor count and henceforth the gate count in a typical industrial design becomes doubles. Also, every few years the size of the transistors used shrinks. As these trends continue, several new issues become relevant in the testing of VLSI circuits. Our focus is addressing three of these problems using different heuristics to find solutions to them. Two of the problems share the same objective but through very different means making them separate problems. It is not relevant to find a final or best solution to the three issues addressed since they are NP-hard problems: it is therefore believed that an optimal solution cannot be found without utilizing $O(e^n)$ (where n is the size of the problem, in this case the size of the circuit in the number of gates) computing time. Also, as testing technology changes a usable solution may need to be reviewed and tuned, or may even become obsolete.

The first of the addressed problems is the need for low power testing that appears in some designs. When a response to a test vector is captured by state

elements in scan testing, the switching activity of the circuit under test may be large resulting in abnormal power dissipation and supply current demand. High supply current may cause excessive voltage supply droops leading to larger gate delays which may cause good chips to fail tests. Excessive power dissipation may cause hot spots that could damage the circuit under test. Given this problem, we propose a method that can be used to create a low power test set or to modify in a post-processing step an existing test set to reduce its power.

The second addressed problem regards circuit modification for testability. In this case our objective will be to reduce test pattern count and therefore test cost. As previously explained, the logic gate count of typical industrial designs continually increases and pattern count tends to grow with the size of the design. The increasing size of industrial circuits also impacts the time consumed by an ATPG to create a test set, which also tends to grow with the design size. Six years ago, a technique called test compression started to be used on industrial designs. Test compression may reduce data volume and test application time by a factor that can vary from ten to a couple of thousands but consumes around 1.5X to 5X the time spent in test generation [1]. Given this time overhead, commercial ATPGs started using faster, but less effective (in terms of pattern count) engines based on quicker compaction schemes. These two effects combined to create the need for solutions that address the pattern count increase problem for the next generation of designs. The method for identification of test points should at least work with one existing ATPG set of heuristics, ideally the method would be general to any ATPG, and should identify a different set of test points for each ATPG set of heuristics or a set of test points independent of the ATPG. Two types of test points exist: control points and observation points. In this work we will focus only on observation points because the selection of control points requires different techniques.

The third addressed problem is about pattern count growth. The approach followed is the creation of an ATPG set of heuristics capable of achieving small test sets. The newly developed set of heuristics are more time consuming than the ones previously utilized, but their run time growth normalized to their increase in size is almost constant. This property makes them scalable to industrial designs. Since they are dependent only on ATPG modifications, they are compatible with test point insertion. However, the set of test points computed for the same circuit with different ATPG heuristics may differ because of the properties of the test point computation algorithm.

For the reasons explained at the beginning of this introduction, the three methods proposed do not intend to be final solutions to the problems mentioned. Instead, they are designed to improve on the previously known methods and be relevant during a certain time frame.

1.2 Test of Digital Circuits

Manufacturing test is done after a circuit comes out of the manufacturing line to determine the presence of defects. The possible defects that are often found in a VLSI circuit occur due to random particles, process defects and lithography issues [2][3] that may make some defects design dependent. A representation of the behavior of the circuit in the presence of a defect is used as a surrogate for the actual defect; this representation is called a fault model. Most manufacturing defects are detected using the stuck-at fault model. However, in recent technologies the transition fault model detects a significant fraction of the defects. These models are described in Section 1.2.3.

The manufacturing test of a circuit that is composrised of only combinational logic is a relatively easy task. The circuit inputs can be set to the desired values and the circuit outputs can be observed. The set of zeros and ones applied to the

circuit inputs is called a test vector, and the circuit outputs are called the circuit under test response.

On the other hand, the test of sequential circuits is a more complicated task. The circuit sequential elements need to be set to the desired values, called the circuit state. Then a clock pulse needs to be applied to capture the circuit response for measuring. Scan is a technique applied to the sequential elements of a circuit that enables the setting of the circuit state. This simplifies the task of testing a sequential circuit making it similar to the test of a combinational circuit.

Given the size of present day circuits, the task of creating a test vector set for all possible manufacturing defects that a circuit may have is automated. Powerful ATPGs are employed to this effect. These ATPGs tools are based on different search algorithms devised for this task.

In this Section, some principles of manufacturing test will be introduced along with commonly used Design For Test (DFT) techniques. We will discuss: defects, scan, two of the existing fault models, one of the algorithms to create a test for a given fault (D algorithm), Built-In Self Test (BIST) and test compression.

1.2.1 Defects

While the causes and expressions of defects are various, they have been, in the past, divided into two major categories. *Shorts*, which occur when current conduction is present but not desired and *opens*, which occur when conduction is not present but it is desired. The behavior of these types of defects is determined by its location, whether in the transistor structure or in the interconnect between transistors.

Manufacturing processes that use aluminum tend to have more short [4] defects than open defects. Both extra conductive material and lack of insulating material can cause a short. These two mechanisms can appear due to:

- Photolithographic printing errors
- Conductive particle contamination
- Incomplete etch
- Incomplete metal polish
- Crack in the insulator
- Gate oxide defect causing pinhole

Displaying an opposite behavior to shorts, opens also appear due to the opposite reasons. Which are: missing conductive material or extra insulating material. These can happen due to:

- Photolithographic printing errors
- Step coverage
- Incompletely filled via
- Electromigration
- Silicide agglomeration
- Incomplete via etch or via foreign material
- Insulating particle contamination

A single isolated problem is not always the cause of defective behavior, sometimes a circuit parameter is out of specification across a wide area. This may cause failures or susceptibility to other problems like temperature effects and crosstalk.

In manufacturing processes at the 130nm node and below, other defect mechanisms start appearing with increasing frequency. They are related to the

transition from aluminum to copper that entails a change in the process. Also, to the appearance of optical defects due to layout features being below the wavelength of light used during photolithography and also to design related defects [2].

> Historically, all testing was functional, and asked the question "Does the device do what is supposed to do?"... For digital logic, functional tests became too expensive to develop... As a result, functional tests in production have largely (but not completely) been replaced by structural tests. Structural tests changed the basic questions being asked by test, and expanded the "Does it work?" question into a new question and syllogism: "Are all circuit elements present and working? If so, and the design is correct, then it must work" [2].

Given this new approach to the testing of digital circuits, new techniques have been developed. These include *scan*, *fault models* (as a surrogate for the actual defect), *test generation algorithms* (that target faults instead of defects), *BIST* and *test compression* (to reduce test data volume and test application time).

1.2.2 Scan Based Test

To test a digital circuit several test vectors are applied to its inputs. Then, the circuit under test (CUT) response, which also consists of another stream of zeros and ones, is analyzed. If the CUT responses match the **golden** (fault-free) responses, then the circuit is considered to be functioning properly. The input test vectors and their responses are stored in an Automatic Test Equipment (ATE), which applies the tests to the CUT and analyzes its responses. Figure 1.1 shows a diagram of this process.

To detect a fault, a difference between the golden and the CUT responses must be created. For this purpose, at least one fault must be activated and propagated to an output. As previously stated, ATPG tools are used for this purpose based on

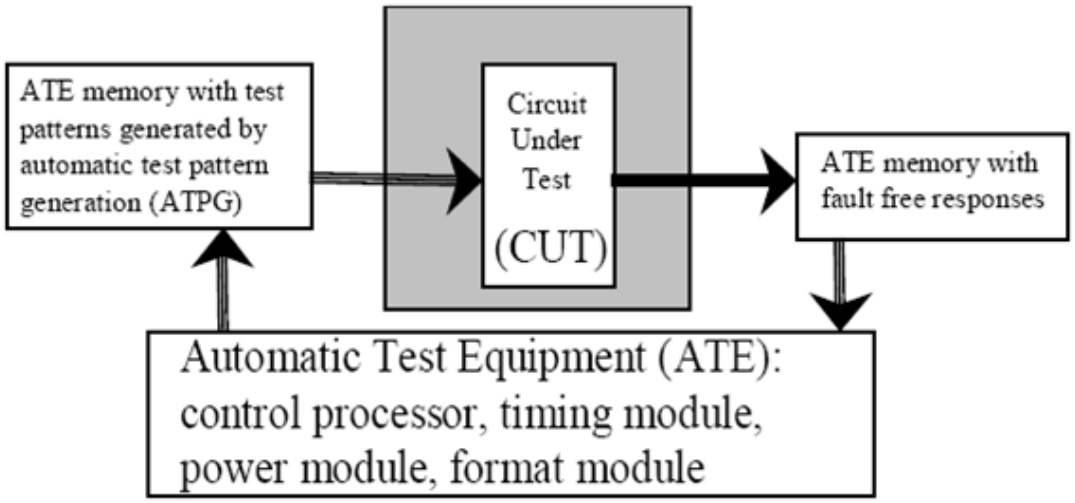

Figure 1.1: Manufacturing test of a circuit. From [5]

different search algorithms. For a given fault, an ATPG will either create a test for it or prove it un-testable (redundant). Some of these are the PODEM, FAN and D algorithms [6]. These algorithms work by searching the space of possible input assignments to find a combination that will detect a given fault. The D-algorithm will be reviewed in more detail in Section 1.2.4.

The CUT may contain sequential elements, namely flip-flops or latches. Sometimes a test vector for a given fault f requires that one or more of the sequential elements be assigned to a specific logic value. To test f when this occurs, the ATPG needs to create another time frame and justify those assignments to circuit inputs, i.e. to assign input values that will set these gates to the desired values. The result of this is that a large percentage of faults now become un-testable due to the sequential elements. To cope with this problem a technique called scan was proposed [7].

In scan, a subset of the circuit sequential elements is replaced with scan cells. Full scan is when all the circuit's sequential elements are replaced. There are many possible implementations of a scan cell; the most common is shown in

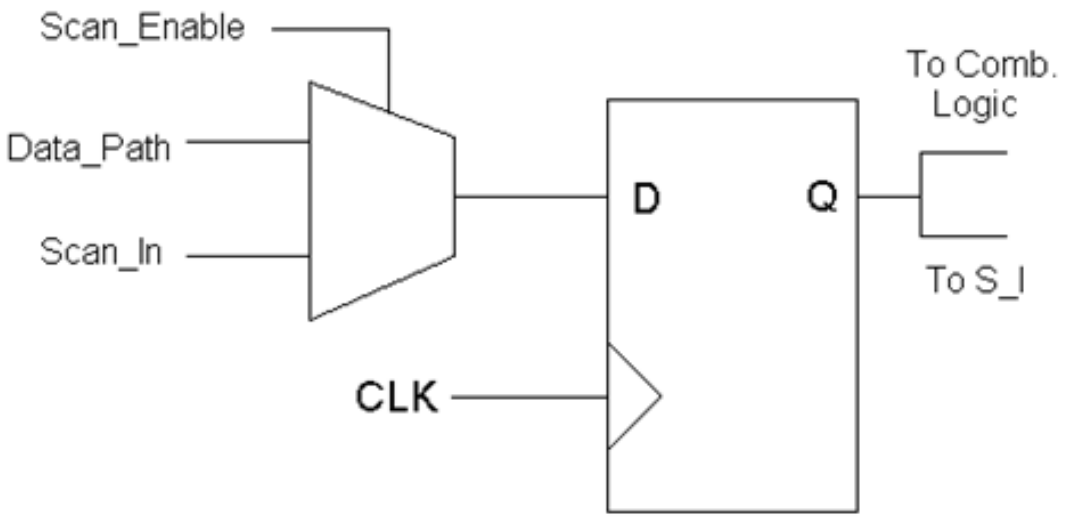

Figure 1.2: A scan cell

Figure 1.2. When using scan, the circuit sequential elements can be set to arbitrary combinations of values by shifting through Scan_In the desired combinations. This enables the test of previously un-testable faults but it may set the circuit into non-functional states.

Through scan, we can control the values that a scan cell will assume by shifting in arbitrary values through the Scan_In input, the scan cell output can now be considered a new circuit input, or pseudo-primary input (PPI), during test mode. Also, since the scan cell can capture the functional CUT response to a test and we can observe this response through shifting out the scan cell values, the Data_Path input of the scan cell can be considered a new circuit output, or pseudo-primary output (PPO), during test mode.

Scan is now a widely adopted technique. Circuits and ATPGs are designed to work under the assumption that this DFT technique will be used. It is important to note that near to full scan is used more than full scan because inserting an extra MUX at a time critical path will reduce the circuit frequency. Thus, even the improved testability of full scan does not have enough payoff for it to be employed,

except in very few designs.

1.2.3 Single Fault Models

In this Section, two fault models will be reviewed: the stuck-at fault model and the transition fault model. The single fault assumption comes from the frequent testing strategy, in which a system is tested often enough to make the probability of a system developing more than one fault very small [6].

1.2.3.1 Stuck-at Fault Model

A stuck-at fault [8] happens when a line in the circuit is stuck at a fixed logic value. One case in which this may occur is when there is a short between ground or power lines and a signal line. Different technologies will be prone to different defects that may cause stuck-at faults. Here, the causes for stuck-at faults in each technology will not be discussed; the focus will be on the model itself. This is the earliest fault model, and still the most common. It was originally proposed for circuits consisting of resistors and vacuum tubes.

A stuck-at fault can occur in either a stem or a fanout branch. The differentiation between faults in stem lines and branches of the stem line is to model the case when a gate input is stuck, or under certain circumstances appears to be, at a logic value. This is equivalent to saying that a stuck-at fault can be either at the output of a gate or at its inputs. A notation commonly employed when line l is stuck-at 0 (1) is *l/0(1)*. Most of the defects that a CUT may have are detected using the single stuck-at fault model. In previous technologies, test for stuck-at faults were the only test applied to a system. With the continuously shrinking sizes of the transistors employed in modern designs, other types of defects not covered by the tests for stuck-at faults are beginning to appear in the CUTs. For this reason, tests for other types of faults are being applied, such as the transition

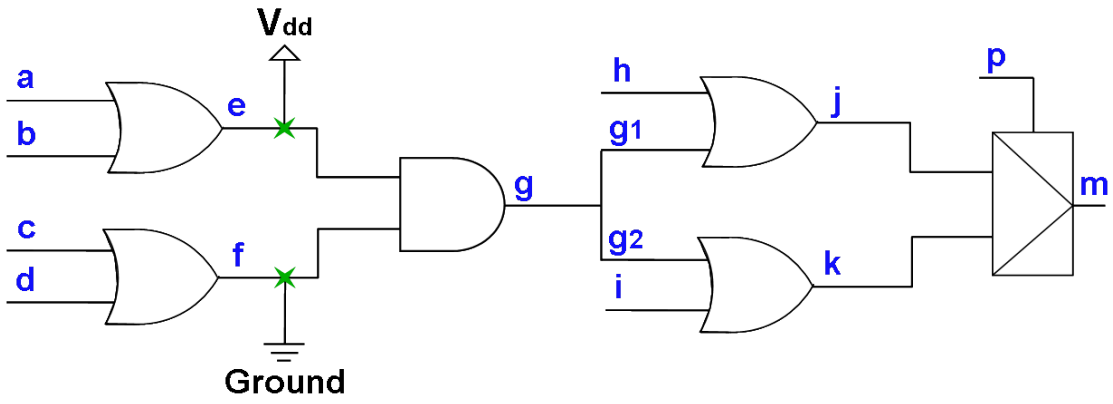

Figure 1.3: Stuck-at faults example

fault model, which is presented next.

Figure 1.3 shows line *f stuck-at '0'* and line *e stuck-at '1'*. Fault *f stuck-at '0'* is represented as a short between line f and ground. Fault *e stuck-at '1'* can be represented as a short between line e and the power supply line V_{dd}. While both faults are represented together in Figure 1.3, it is important to notice that it is rare that both happen in the same CUT at the same time because of the single fault assumption.

1.2.3.2 Transition Fault Model

Certain types of defects in the manufacturing of the transistors that comprise the circuit gates may cause the gate to have a higher than normal delay. This abnormal delay causes the gate to switch at a lower than normal speed when its inputs change. When this delay is large enough the defect is modeled as a transition delay fault [9][10].

A transition delay fault occurs when a gate that should switch its output from 0 (1) to 1 (0), due to a change in its inputs, does it in a longer than normal time. If the delay introduced is large enough so that its effects can be seen at least at one of the circuit primary outputs (POs) or captured in a scan cell, the circuit

cannot operate at its intended clock speed without having a faulty behavior.

A transition fault that occurs when a gate switches from 0 (1) to 1 (0) in an abnormally long time is called a slow-to-rise (slow-to-fall) transition fault. To test a slow-to-rise (slow-to-fall) fault a test vector needs to set the gate output to 0 (1) in the first time frame and, by applying a functional clock, launch a transition in the gate to 1 (0) and propagate the transition to an observation point in the second time frame. Thus, a test for a transition fault should consist of a pair of input vectors to the CUT. One method to produce the second vector in the pair will be reviewed in sub-Section 2.2.1.

In the transition fault model faults are located only at the outputs of gates. The test for this type of fault will be reviewed in Chapter 2 and is called *at-speed test* because of the presence of at-speed clock pulses to launch and capture the faults. This type of fault is being commonly used as a target to test generation for testing the behavior of the CUTs in at-speed environments.

In Figure 1.4 a capacitor to ground in line f will slow the transitions *'0' to '1'* and *'1' to '0'* at the gate output. Every time f changes value the gate driving f needs to charge or discharge the capacitor introducing delay. A slow-to-rise fault in f can be modeled as a big resistor to ground in *f*. A slow-to-fall fault in f can be modeled as a big resistor to the power supply line V_{dd} in *f*.

1.2.4 D-Algorithm

To create a test for a given fault *f*, f must be activated and propagated to an observation point. To activate f is to create the conditions under which there is a difference in behavior between the faulty machine and the good machine. To propagate it is to carry through logic gates the difference in behavior until an observation point is reached.

This problem is called circuit-SAT and is known to be NP-complete. The

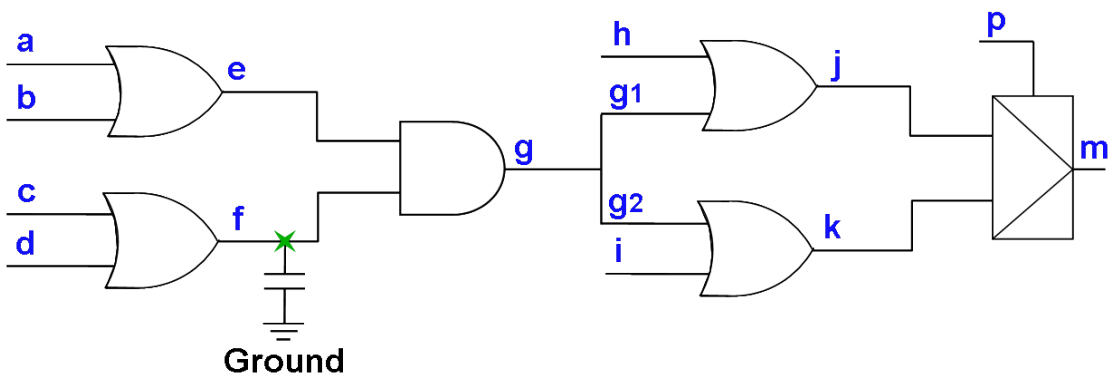

Figure 1.4: Transition delay fault example

algorithms to create a test for f exhaustively search the whole space of possibilities, i.e. the combinations of input (PIs and PPIs) assignments, to find if f can be detected or not. Since this implies exponential time, they try to reduce the search space only to the relevant inputs.

The D-algorithm [11] works by first identifying all the necessary gate output values (assignments) to detect f. After this is done, some lines may have values that are not justified by its inputs because there is more than one possibility in which their values can be justified. We call this set of lines or gate outputs the J-frontier. Also, the fault effect (difference in behavior between the good and faulty machine) may not be propagated to an observation point yet because there is a choice between paths to follow. The gates in which a fault effect is present at an input and could be propagated to the gate output is called the D-frontier.

The D-algorithm maps the decisions to be made in justification and propagation to the gates in which the decisions are to be made, effectively restricting the search space to the relevant portions of the circuit. The order in which the D algorithm makes the decisions choices is based on heuristics, and one possibility is to make them at random. If a decision results in implications that lead to a conflict in values at gates or an empty D-frontier the algorithm encounters a conflict. Then

the algorithm will undo decisions previously made to avoid the conflict and chose another alternative. This is called backtracking and is what guarantees the D algorithm will cover the entire search space. If the D algorithm tries all possible combinations and in each reaches a conflict, then the fault is proven un-testable.

Given that trying all possible combinations may result in an exponential (in the gate count of the CUT) number of operations, an abort limit is placed on the number of possible backtrackings. Since some faults may abort and neither be proven testable by constructing a test or un-testable by exhausting all possibilities, the D algorithm (when abort limit is used) also becomes a heuristic.

Figure 1.5 shows an example of the previously discussed D-algorithm. In Figure 1.5, fault f (a stuck-at fault) is activated through $a = 1$ and $b = 0$, then the fault effect is propagated from e to g by the assignment $f = 1$.

Figure 1.5 shows the state of the circuit at that moment. All necessary assignments for f have been computed, but a test has not yet been created for f. The assignment $f = 1$ is not justified and is in the J-frontier. Also the fault effect has not been propagated to a point where it can be observed because there are two paths to follow: g_1 and g_2. Thus g is in the D-frontier.

The order in which the decisions in the J-frontier and D-frontier are made is based on heuristics and varies in different ATPGs. Moreover, in more complex circuits than the one in Figure 1.5, the resolution of decisions in the J-frontier may lead to new decisions in the J-frontier and conflicts with the D-frontier that can remove gates from the D-frontier and even render it empty, in which case backtracking occurs. The resolution of D-frontier decisions may create both J-frontier and D-frontier decisions. This process continues until all decisions in the J-frontier and D-frontier have been solved, in which case a test cube for f is created, the abort limit in backtracking is reached or the fault is proven redundant.

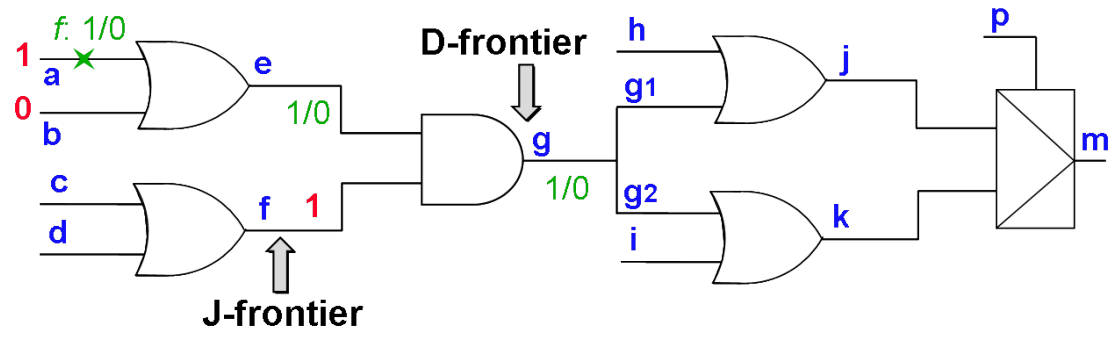

Figure 1.5: Example of the D-algorithm

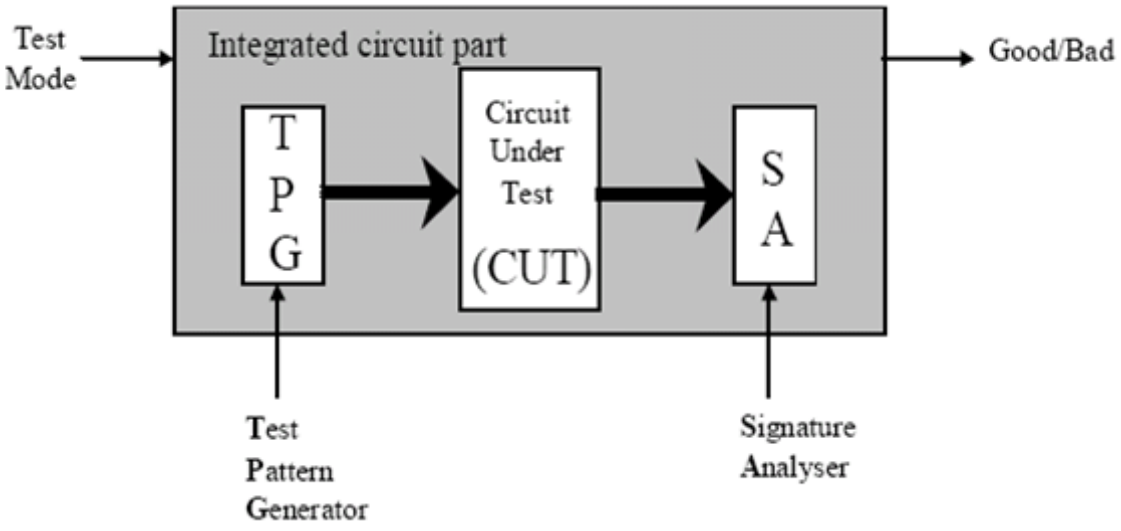

Figure 1.6: High level view of the BIST scheme. From [5]

1.2.5 Built-In Self Test

"BIST is a DFT technique in which test generation and test application is accomplished through built-in hardware features" [12]. Figure 1.6 shows a CUT with BIST. A test pattern generator (TPG) generates patterns that are loaded into the CUT and a signature analyzer (SA) examines the CUT response to the test patterns when the circuit is in Test Mode. The signature analyzer has an output to indicate if the circuit has passed or failed the test.

The TPG can be implemented with one of the following: ROM, linear

feedback shift register (LFSR), cellular automaton or binary counter [12]. The SA can be implemented with one of the following: ROM and comparison logic, LFSR, multiple-input signature register (MISR), cellular automaton, level counter, transition counter or XOR trees [12]. More details of the BIST architecture can be found at [6][12][13].

BIST is a technique based on scan. The TPG loads the vector into the primary inputs (PIs) and scan cells (PPIs); and the SA examines the CUT response from the primary outputs (POs) and scan cells (PPOs).

Most typically used TPGs produce pseudo-random patterns. This results in an increase in the number of patterns needed to test the CUT. To address this problem several techniques are employed [6][12][13], such as: weighted pseudo-random patterns, TPG seeding and test point insertion. These techniques aim at increasing the testability of random pattern resistant faults by either producing different input vector sequences to the CUT or changing the signal probabilities of the circuit lines.

1.2.6 Test Compression

In past years several test compression techniques have been developed to reduce test data volume and test application time, which reduces manufacturing costs. Test compression techniques are based on scan. They segment the scan path into several scan chains and instead of feeding each scan chain with one input, several scan chains are fed with fewer number of inputs called scan channels.

Here the focus will be on one technique called Embedded Deterministic Test [14] (EDT). EDT is based on adding additional hardware at the inputs and outputs of the circuit. A data decompressor is placed at the input. The data decompressor is implemented by a *ring generator* (optimized LFSR) and a *phase shifter*. A response compactor is placed at the output. The circuit scan chains are divided evenly, if

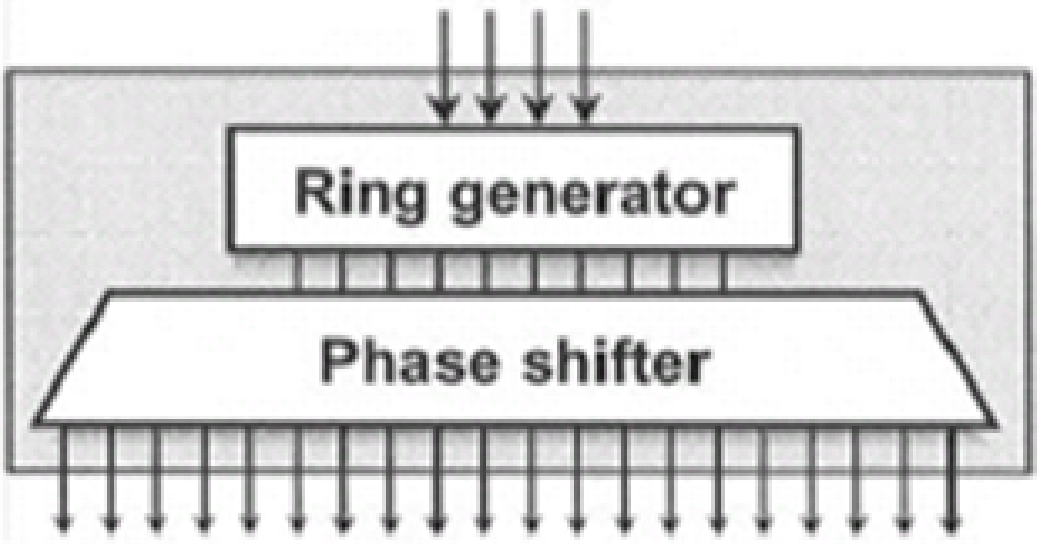

Figure 1.7: An EDT decompressor. From [14]

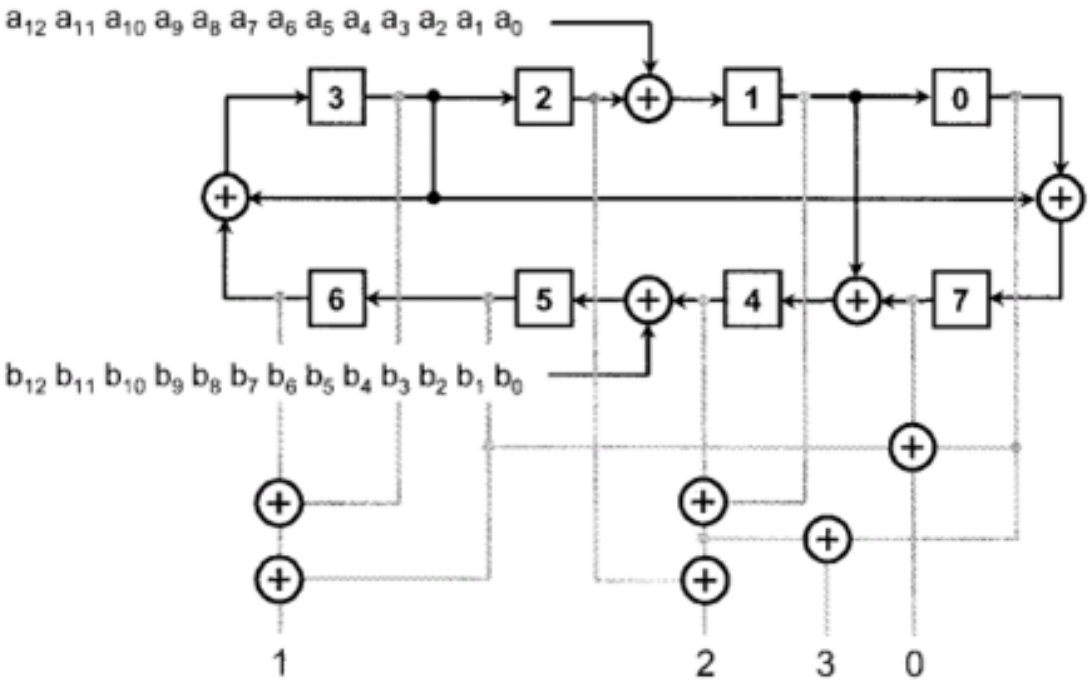

Figure 1.8: A decompressor. From [14]

possible, into several shorter chains. For the purpose of producing the desired output, the ring generator is continuously seeded with data. The ratio between the inputs of the ring generator and the outputs of the phase shifter determines the maximal compression possible. Figure 16 shows a circuit with EDT being tested by an ATE. Figure 1.7 shows the internal schematic of an EDT decompressor and Figure 1.8 shows an implementation of a four-output 8-bit decompressor.

In EDT the compactor consist of an XOR tree. Since XORs always propagate fault effects (when no unknown values exist), every scan chain can be observed at the same time using a reduced number of outputs, effectively reducing the response data. Since unknown values can be present in the CUT response to a test, AND gates are placed at the outputs of every scan chain to block these unknown values.

CHAPTER 2
LOW POWER TEST

When the response to a test vector is captured by state elements in scan testing, the switching activity of the circuit may be large resulting in abnormal power dissipation and supply current demand. High supply current may cause excessive power supply droops leading to larger gate delays which may cause good chips to fail tests. This chapter presents a scalable approach called Preferred Fill to reduce average and peak power dissipation during capture cycles. Preferred Fill is introduced also as a post-processing step to modify existings test sets. Also a hybrid technique, based on a combination of Preferred Fill and Adjacent Fill, aiming at reducing both capture and shift power is presented. Experimental results presented for benchmark and industrial circuits demonstrate the effectiveness of the proposed methods.

2.1 Introduction

Scan based test has become the standard method for manufacture test. Earlier it has been observed that scan tests may cause switching activity far exceeding the activity during normal operation of the circuit [15][16][17]. Excessive switching activity is caused by scan tests requiring the circuit under test (CUT) to operate outside of the normal functional operation. Excessive switching activity during the application of scan tests are caused both during scan chain shifts to load tests and unload test responses as well as when the scan cell contents are updated using functional clocks in what are referred to as capture cycles. Abnormal switching activity causes abnormal peak as well as average power dissipation and supply currents. Excessive power dissipation may cause hot spots that could damage the CUT. Excessive peak supply currents may cause supply voltage droops

resulting in increased gate delays during test. Increased gate delays during test may cause good chips to fail at-speed tests causing yield loss [16].

Several methods have been proposed to reduce the switching activity in a CUT during scan based tests. Earlier methods to reduce switching activity during scan shift include adding additional logic [17][18][19][20][21][22], scan chain segmentation [19][23][24][25], ordering of tests [26] and scan elements [27][28], and reduced transition tests [29]. Earlier approaches to reduce switching activity during capture cycles include selectively deactivating some scan chains [30]citeLPTref18, segmented scan with gated clocking [23][24], test generation methods [32][33][34][35][36][37], and methods to fill unspecified values in test cubes [38][39][40][41][42][43][44][45].

The test generation method proposed in [33] that restricts the scanned in states to the set of reachable states insures that the CUT operates in the functional mode only during capture cycles. Thus, such tests not only avoid abnormal switching activity during capture cycles but also avoid detection of faults that do not affect normal functional operation. Methods to generate tests with reachable scanned in states for transition delay faults (TDFs), called functional tests, were proposed in [35]. The requirement that scanned in state of a test is a reachable state may be difficult to ascertain in larger designs. For this reason pseudo-functional tests that attempt to operate a CUT close to its normal functional operation were investigated in [34][36]. Pseudo-functional test generation procedures determine a subset of non-reachable states and generate tests that avoid using any of the non-reachable states as scanned in states. Pseudo-functional tests do not guarantee avoiding non-functional operation that may cause excessive switching activity. Additionally finding a sufficiently large set of un-reachable states in large designs and specially those with multiple clock domains may not be practical.

The methods to fill unspecified entries in test cubes to reduce circuit switching

activity have the advantage that they require only minimal changes to the existing ATPGs. However, in order to make such methods scalable, the specific procedure used to fill the unspecified entries must achieve substantial reduction in switching activity while not increasing the run times of ATPGs substantially. In this work we propose a scalable and effective method, called *preferred fill*, to fill unspecified entries in test cubes such that the switching activity during capture cycles of the tests is reduced. The proposed method is applied to generate launch off capture tests for TDFs. However, the method can be applied to generate tests for other fault models such as stuck-at and path delay faults.

2.2 Preliminaries

In this section we first briefly review launch off capture (LOC) also called broadside test method [46] used to detect delay faults in standard scan designs. We discuss the issues related to supply current demands and power dissipation during the application of LOC tests. Next we review earlier works that proposed methods to fill unspecified values in test cubes to reduce supply current and power dissipation during test application. We also discuss the shortcomings of the known methods that are addressed by the method proposed in this work and described in the following sections. Since our initial focus in this document TDFs, initially all discussions will assume this fault model. Later in the document, the proposed method will be also applied to stuck-at faults (SAF), when this is the case it will be explicitly stated.

2.2.1 Launch off Capture Tests

For standard scan designs illustrated in Figure 2.1 two methods have been used to test TDFs. One is called launch off capture (LOC) [46] and the other is called launch off shift [47]. Both methods use a two-pattern test $< V_1, V_2 >$ to detect

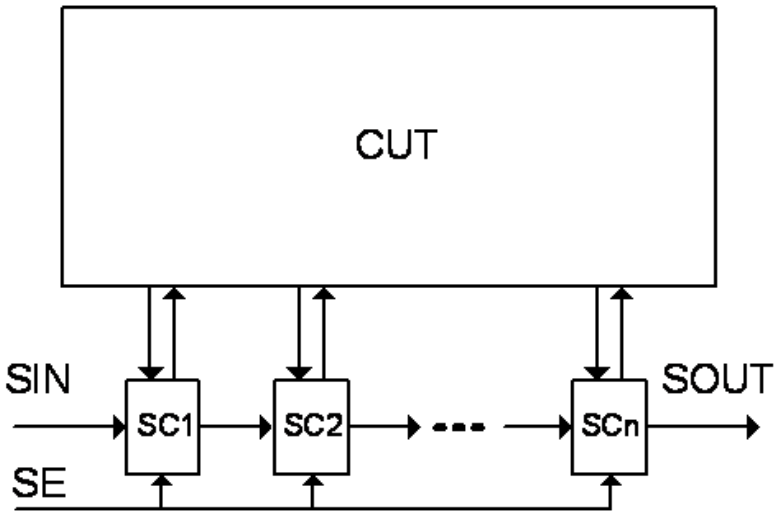

Figure 2.1: Standard scan

a targeted TDF. In both methods, the first pattern V_1, called the initialization pattern, is scanned in, with the scan enable (SE) signal asserted. For a slow to rise (slow to fall) TDF, V_1 sets the fault site to 0(1). The second pattern V_2 is generated differently in the two methods. In LOC the second pattern is generated through the combinational logic of the circuit under test (CUT) by applying a clock pulse with SE de-asserted. This is referred to as the launch cycle and is illustrated in Figure 2.2 for a scan chain of length n. Application of V_2 activates the fault by launching a transition at the fault site and also propagates the fault effect to an observed output (primary output or a scan cell). For a slow to rise (slow to fall) TDF V_2 is a test for a stuck-at-0 (stuck-at-1) fault at the fault site. Following the application of V_2, another clock pulse is applied with SE still de-asserted to capture the CUT response to the test. This is also shown in Figure 2.2 where the corresponding clock pulse is labeled C for capture. Often the two clock pulses used to launch a transition and capture test responses are both called capture cycles. Thus a standard LOC test uses two capture cycles.

During the application of LOC tests, circuit nodes switch states due to scan

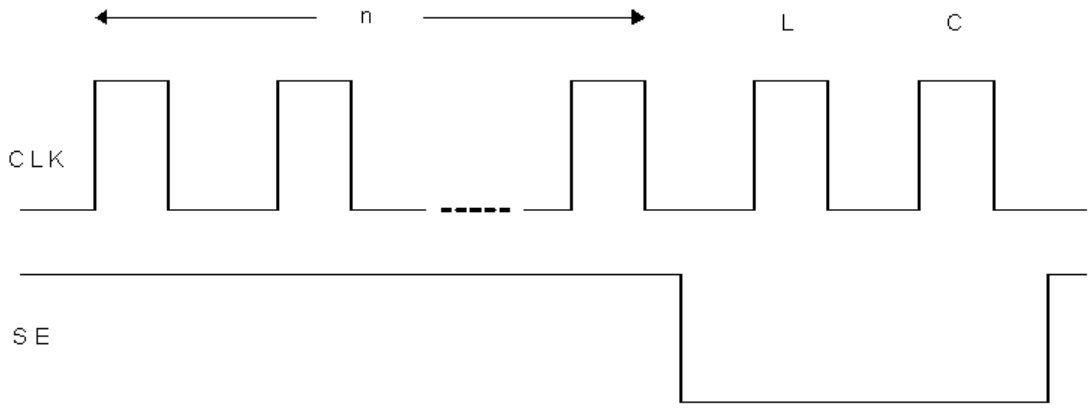

Figure 2.2: Timing diagram of LOC Tests

shifts as well as capture cycles. The switching activity caused by the changes in the circuit inputs (primary inputs as well as the scan cells) initiated by the capture cycle clocks may be considerably higher than during normal circuit operation. High supply current demand may cause supply voltage to droop which tends to increase signal propagation delays of effected gates. Increased delay due to supply voltage droops may lead to capturing faulty responses, especially during the second capture cycle. This causes good chips to fail tests leading to yield loss [16]. Thus to reduce the potential yield loss it is critically important to reduce peak switching activity caused by the first capture cycle changing the state of the scan cells. It is also important to reduce the switching activity caused by the second capture cycle to prevent excessive power dissipation.

In this work we use *Weighted Switching Activity* (WSA) defined next. WSA was also used to represent instantaneous power and current in earlier works [18]. The *weighted switching activity* (WSA) of a node is the number of state changes at the node multiplied by (1+node fan-out). The WSA of the entire circuit is obtained by summing the WSA of all the nodes in the circuit.

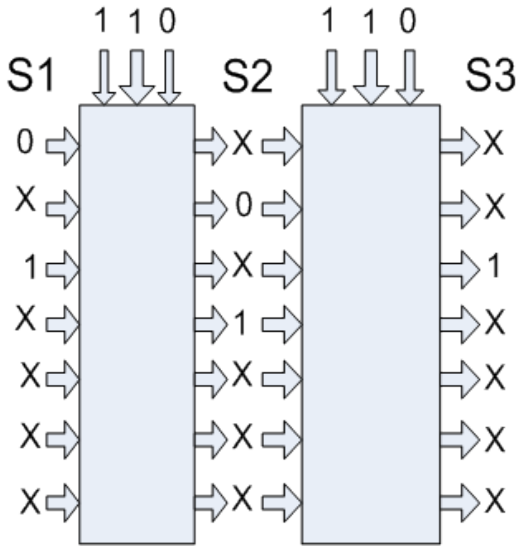

Figure 2.3: A LOC test

2.2.2 Related Earlier Works

A test pattern for a target fault or a set of target faults typically contains many unspecified entries. The unspecified values can be arbitrarily specified to binary values. This is often referred to as filling of unspecified values or simply fills. Among the works that target reduced WSA some target reducing WSA during scan shifts [39] and others target reducing WSA during capture cycles [42][43][44]. Experimental results in [39] showed that a method called adjacent fill not only reduces WSA during scan shift but also the average WSA (averaged over all tests) caused by capture cycles. However, it was observed that this method may increases the peak or maximum value of WSA [43]. As noted earlier increased peak WSA during capture cycles increases the possibility of failing good chips due to increased supply voltage droops [16]. For this reason recent works proposed methods to reduce peak WSA during capture cycles of LOC tests [42][43][44]. The objective of this work is also to reduce peak WSA and average WSA caused during capture cycles in LOC tests.

Next we briefly describe the methods used in [43][44] to fill unspecified entries

in LOC test cubes. The two methods are essentially the same with some differences in details. We discuss the steps used in [44]. Consider the two-pattern test illustrated in Figure 2.3. In Figure 2.3 we show two frames. The vertical inputs at each time frame are primary inputs and the horizontal inputs and outputs are the present and the next states of the CUT, respectively. A vector in a two-pattern test has two components. One component corresponds to the primary inputs and the other component corresponds to the state variables. In Figure 2.3, S_1 (S_2) is the state part of the initializing (test) vector V_1 (V_2) of the two-pattern test $< V_1, V_2 >$. S_3 is the state captured by the second capture cycle. If the state and the primary inputs of V_1 and the primary input part of V_2 are fully specified then S_1, S_2 and S_3 will all be fully specified. Further more S_2 and S_3 are determined entirely by S_1 and the primary inputs. Thus the unspecified values available to fill to reduce WSA are only in S_1. For the sake of simplicity of explanation, assume that the primary input values are held constant during the application of the two-pattern test and are also fully specified. Then the switching activity in the circuit is caused by the changes in the state from S_1 to S_2 after the first capture cycle and next due to the state change from S_2 to S_3 after the second capture cycle. The method proposed in [44] fills the unspecified values in S_1 such that S_1 and S_2 as well as S_2 and S_3 differ in as few places as possible. In other words the method tries to fill the unspecified values in S_1 such that the Hamming distances between S_1 and S_2 and S_2 and S_3 are as small as possible. The procedure called LCP-fill from [44] to fill unspecified values in S_1 to reduce WSA caused by capture cycles of LOC tests is briefly reviewed next.

Consider the example illustrated in Figure 2.4 which shows a test cube with S_1 = (0, X, 1, X, X, X, 1) which results in S_2 = (X, 1, X, 0, X, X, 0). Notice that the first and the third elements of S_1 are specified and the corresponding elements of S_2 are unspecified. Similarly the second and the fourth components of S_2 are

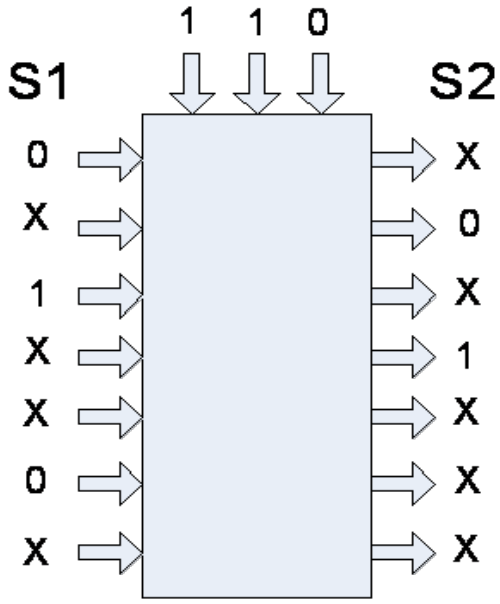

Figure 2.4: LCP filling

specified while the corresponding elements of S_1 are unspecified. In order to reduce the places in which S_1 and S_2 differ or equivalently to keep the Hamming distance between S_1 and S_2 small, the procedure of [44] iteratively specifies few selected unspecified entries in S_1 at a time. In the example being considered, one can specify the second and the fourth elements of S_1 to 1 and 0, respectively, to match the specified values in the corresponding elements of S_2. One can also attempt to set the first and the third elements of S_2 to 0 and 1, respectively, to match the values in the corresponding elements of S_1, through implication and line justification procedures of ATPGs. These steps will specify some of the originally unspecified values in S_1. S_1 is simulated with the newly specified values to obtain S_2 and the procedure is iterated until all components of S_1 are specified. One can also fill the unspecified values in S_1 such that the Hamming distance between S_2 and S_3 is small also. The procedure in [44] attempts to balance between the two objectives: keeping the Hamming distances between S_1 and S_2 small and the Hamming distance between S_2 and S_3 small. The LCP-fill procedure leads to considerable reduction in peak

and average WSA of LOC tests compared to random fill of the unspecified values in the test cubes. However the run time of this procedure could be potentially high. The reasons for this are the repeated simulations of incrementally updated test cubes and use of implications and line justification steps as part of the procedure.

In this work we investigate a simple and scalable procedure to fill unspecified values to reduce WSA during capture cycles of LOC tests. The procedure fills all the unspecified values in the initialization vector of a two-pattern test in one step rather than iterative incremental fill and simulation. It also does not use implications and line justification procedures. Experimental results presented for benchmark circuits show that the reductions in the peak and average WSA using the proposed procedure are similar to those obtained using LCP-fill while the run times are substantially smaller.

2.3 The Proposed Method

In this Section the proposed method called *preferred fill* (PF) to fill unspecified values in test cubes is described. For the sake of simplicity we first describe the basics of the method in the context of circuits with 100% scan and present experimental results on ISCAS-89 benchmark circuits. In the next Section we discuss extensions to the basic method used for industrial designs.

2.3.1 Preferred Fill

Consider a two-pattern LOC test $< V_1, V_2 >$ for TDFs. It can be written as $V_1 = (PI_1, S_1)$ and $V_2 = (PI_2, S_2)$, where PI_1 and S_1 correspond to the PI values and the PPI values in the initialization vector and PI_2 corresponds to the PI values in the test vector and S_2 is the PPO values implied by V_1. Our goal, as in [42][43][44] is to reduce the Hamming distance between V_1 and V_2 by reducing the Hamming distance between PI_1 and PI_2 as well as between S_1 and S_2.

The Hamming distance between PI_1 and PI_2 can be minimized in a straightforward manner as done in [43]. Where ever possible we first fill the unspecified values in PI_1 (PI_2) to match the specified values in PI_2 (PI_1). This step is not needed in case the primary inputs are held constant over the two time frames during test generation. After this step, all the remaining unspecified values in PI_1 and PI_2 will be in the same positions. We randomly fill these values to have the same specified value. To illustrate filling of the values in PIs consider the following. In a two pattern test cube let $PI_1 = (1XXX01X)$ and $PI_2 = (01X0XXX)$. We fill the unspecified values in PI_1 (PI_2) in positions 2 and 4 (5 and 6) with 1 and 0 (0 and 1) to match the specified entries in these positions in PI_2 (PI_1). After this step we get $PI_1 = (11X001X)$ and $PI_2 = (01X001X)$. Next we fill positions 3 and 7 in PI_1 and PI_2 randomly say by 0 and 1, respectively and get (1100011) and $PI_2 = (0100011)$. The Hamming distance between PI_1 and PI_2 after the proposed fill is 1.

It should be pointed out that instead of randomly filling unspecified values in PI_1 and PI_2 in the second step one can use a better procedure. For example we can determine preferred values as we do for PPIs which is discussed next.

Next we describe the procedure we used to reduce the Hamming distance between S_1 and S_2 of a two-pattern test. Recall that we can arbitrarily fill the unspecified values in S_1 since it is the state that is scanned in. Let $S_1 = (s_{11}, s_{12}, s_{13}, ..., s_{1n})$ and $S_2 = (s_{21}, s_{22}, s_{23}, ..., s_{2n})$. If s_{1j} is unspecified then we should fill it with 1 (0) if the probability of s_{2j} taking the value 1 (0) is higher than it taking the value 0 (1). In other words we should fill s_{1j} with a value that is more likely to be held in the j^{th} scan cell. With this in mind, we define $HP_j(v)$, $v \in 0, 1$, as the probability of the j^{th} flip-flop holding the value v under the assumption that all PIs and PPIs other than the j^{th} PPI are applied random inputs. Note that $HP_j(v)$ is simply the conditional probability that the j^{th} scan cell will hold the

value v if it is loaded with v and all other inputs are applied random inputs.

Definition: 0 (1) is the preferred value for the j_th scan cell (i.e. s_{1j}) if $HP_j(0) > HP_j(1)$ ($HP_j(1) > HP_j(0)$). Note that if $HP_j(0) = HP_j(1)$ then s_{1j} does not have a preferred value.

The procedure we used to fill the unspecified values in S_1 is the following. First, in every position of S_1 in which it has unspecified value and S_2 has a specified value we fill the unspecified value in S_1 by the specified value in S_2. After this step in every position in which S_1 has an unspecified value S_2 will also have unspecified value (S_2 may have additional unspecified values). This method is called *dynamic preferred fill*. We illustrate below this step using an example.

Let $S_1 = (1XXX01X)$ and $S_2 = (01X0XXX)$. We fill the second and the fourth position of S_1 to match the specified values 1 and 0 in S_2 and obtain $S_1 = (11X001X)$ and $S_2 = (01X0XXX)$. Notice that now every place in which S_1 is unspecified S_2 is also unspecified. In these positions, S_1 is filled with the preferred values if they exist. Otherwise, they are filled randomly.

The hold probabilities defined above or equivalently the conditional probabilities as discussed above can be computed in two ways. One is to use symbolic methods to accurately compute probabilities and the other is to estimate the probabilities by simulating a large number of random patterns. The first method is known to be NP-hard and the second method may also require a large computation effort for modern designs.

A simpler but less accurate method is to use the standard signal probability calculation procedures ignoring statistical correlation between gate inputs [6]. However even this approach may take large run times if the number of scan cells in a circuit is large as is the case with modern VLSI designs. This is because the conditional probabilities must be computed separately for each PPI. For this reason we used a much simpler procedure. It uses simpler procedures to compute signal

probabilities ignoring correlation between gate inputs together with a simplifying assumption. The simplifying assumption we used is that the probability of s_{2j}, the j^{th} component of S_2, assuming the value 1 or 0 is independent of the state of s_{1j}. Under this assumption, we get, $HP_j(1) > HP_j(0)$ ($HP_j(0) > HP_j(1)$) if and only if $P_j(1) > P_j(0)$ ($P_j(1) > P_j(0)$), where $P_j(1)$ ($P_j(0)$) is the probability that s_{2j} takes the value 1 (0). Thus one can determine the preferred values for s_{1j} by simply computing the signal probabilities of s_{2j}. Signal probabilities of all s_{2j}, $1 \leq j \leq n$, can be simultaneously found in one pass using the standard signal probability calculation procedures [6] ignoring the correlation between gate inputs. In our experiments we used this simplified method of determining preferred values for s_{1j}.

Next we give an example to illustrate how we computed signal probabilities and preferred values for s_{1j}, $1 \leq j \leq n$. Consider the circuit in Figure 2.5. On each line we show two numbers. The first number is the probability of the line having the value 0 and the second number is the probability of the line having value 1. The circuit has two PPIs s_{11} and s_{12} and one PI a. We assign to these inputs equal probabilities for 0 and 1 as shown. Next we compute the probabilities for the other lines using the standard formulae [6] and ignoring the correlation between gate inputs. For example the two inputs to the OR gate of the circuit are correlated. The output of the circuit is the PPO s_{21}. The probability of the scan cell holding a 1 (0) is computed by computing the conditional probability of a corresponding PPO being at 1 (0) given that the corresponding PPI is at 0 (1). This is illustrated in Figure 2.6 for the circuit of Figure 2.5. In Figure 2.6(a) we set the 1 probability of PPI s_{11} to 1.0 and the 0 probability to 0.0. The probability of s_{21} being 1 under this condition is 0.75 and hence $HP_1(1) = 0.75$. We can compute $HP_1(0) = 1.0$ in a similar manner as illustrated in Figure 2.6(b). In Figure 2.6(b) we set the 0 probability of s_{11} to be 1.0 and its 1 probability to be 0.0. Since $HP_1(0) > HP_1(1)$

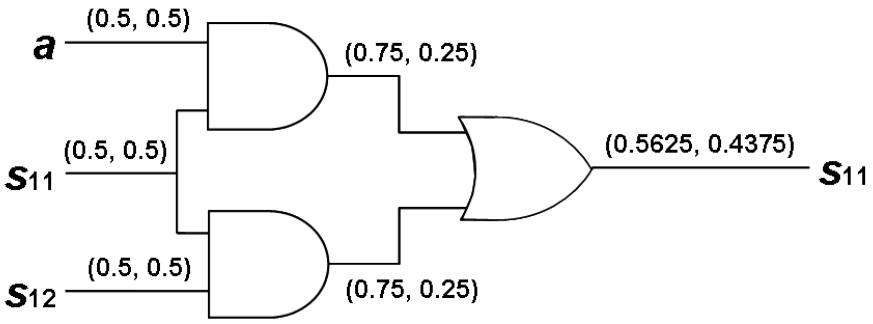

Figure 2.5: Signal probabilities

the preferred value for s_{11} is 0. As this example illustrates computing $HP_j(0)$ and $HP_j(1)$ requires two passes through the circuit for each j, $1 \leq j \leq n$. Instead, as discussed above, we use the signal probabilities $P_j(1)$ and $P_j(0)$ of the j^{th} PPO being at 1 and 0, respectively, to determine the preferred value for the j^{th} PPI. For the example under consideration, we have from Figure 2.5 $P_1(0) = 0.5625$ and $P_1(1) = 0.4375$. $P_1(0) > P_1(1)$ we conclude that the preferred value for s_{11} is 0. In this example the preferred value found for $s11$ using hold probabilities as well as signal probabilities turns out to be the same. However in general it may be different.

We performed extensive experiments on benchmark circuits using all the methods discussed above to determine preferred values for s_{1j}. All the methods to determine preferred values yielded essentially the same results for the reduction of WSA. For this reason we used the computationally less demanding procedure of computing the signal probabilities for s_2j, $1 \leq j \leq n$, illustrated in Figure 2.5.

2.3.2 Experimental Results for Benchmark Circuits

The proposed method was implemented in C language and two experiments were conducted on ISCAS-89 benchmark circuits using Pentium 4 2.8 GHz PC with 1 GB RAM using Linux. For the first experiment we chose an experimental set up

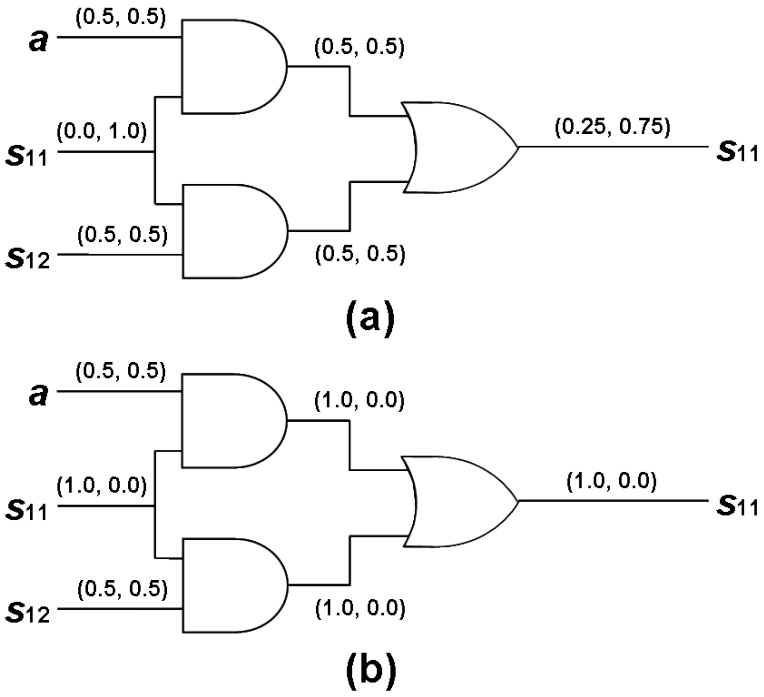

Figure 2.6: Conditional probabilities

for the results on the benchmarks so as to be identical to the one used in [44] with which we compare the obtained results. Test cubes were obtained using an academic ATPG. The test cubes were generated using dynamic compaction procedures to detect all detectable TDFs using LOC tests. The dynamic compaction procedure was allowed to use up to 20% of the unspecified values in the test cube generated for the primary target fault. For each test cube the state captured after the application of the first capture cycle (*i.e.* S_2) was also provided by the ATPG. The unspecified values were filled using the preferred fill proposed in this paper.

In Table 2.1 we give the test generation results. After the circuit name we give the reduction percentages in average WSA and in peak WSA in the CUT caused by the first and the second capture cycle using LCP-fill of [44] and using the preferred fill proposed in this paper, respectively. Both reduction percentages are relative to the case when the unspecified values are filled randomly. From the

Circuit	Average WSA Reduction				Peak WSA Reduction			
	1st Capture		2nd Capture		1st Capture		2nd Capture	
	LCP	Pref.	LCP	Pref.	LCP	Pref.	LCP	Pref.
s1423	19.37	37.95	27.75	27.06	18.49	40.78	6.65	16.69
s5378	49.88	45.07	23.35	46.27	36.89	30.08	22.84	26.30
s9234	18.07	34.59	8.56	15.20	12.25	19.13	2.25	-1.07
s13207	23.95	45.77	9.40	29.55	15.42	26.62	4.51	22.81
s15850	45.82	41.82	31.29	31.56	31.68	35.57	6.78	21.90
s35932	60.38	30.78	53.22	28.56	25.05	23.50	19.88	11.83
s38417	20.07	20.75	18.20	15.86	13.02	29.24	10.96	21.88
s38584	43.96	31.56	37.97	18.57	50.17	27.63	51.29	35.68
Average	35.19	36.04	26.22	26.58	25.37	29.07	15.65	19.50

Table 2.1: WSA reduction results for ISCAS-89 circuits

table, it can be seen that both the preferred fill and the LCP-fill achieve similar reduction percentages in peak and in average WSA compared to random fill.

In Table 2.3.2, we show the run times for the LCP-fill and the preferred fill under the column *CPU time*. The run time given for LCP-fill is for only the fill procedure used in [44] and was provided to us by its authors [48]. The run times reported for preferred fill includes computation of the signal probabilities. It can be seen that the run times for preferred fill were un-measurably small.

Direct comparison of test set sizes will not be meaningful since the test pattern generators used are not the same. However for the sake of completeness we report the test set sizes for LCP-fill from [44] and preferred fill under the column *#Patterns*. We also give the pattern count when the unspecified entries are filled randomly and the dynamic compaction procedure is allowed to use all unspecified values to target detection of additional faults. These pattern counts are given in

Circuit	CPU time		#Patterns		
	LCP	Pref.	LCP	Pref.	Rand.
s1423	0.10	0	135	112	82
s5378	0.70	0	248	206	167
s9234	2.50	0	350	392	328
s13207	13.10	0	356	385	377
s15850	5.90	0	220	202	183
s35932	27.60	0	72	96	40
s38417	87.30	0	227	324	222
s38584	302.40	0	444	320	292
Average	54.95	0	257	255	211

Table 2.2: CPU and pattern increase results for ISCAS-89 circuits

the last column of Table 2.3.2. It can be seen that over all the circuits the number of test patterns are similar as can be seen by the average numbers reported in the last row of Table 2.3.2. Also when preferred fill is used to fill unspecified entries in test cubes on the average the number of tests increases by approximately 20% to 25% from 211.

In the next experiment we used a post-processing step applied to (completely specified) test vectors generated by an ATPG. For this experiment we used completely specified test vectors generated using dynamic compaction with no limit thus obtaining test sets of minimal size. In the post-processing step we relaxed the given test vectors one at a time after ordering them by the WSA value of the first capture.

Relaxing a fully specified test unspecifies some specified entries in it without decreasing fault coverage. We adapted the test vector relaxation method of [49] for

Circuit	Original # of Patterns	Final # of Patterns	% Reduction in WSA				Time
			1st Capture		2nd Capture		
			Peak	Average	Peak	Average	
s1423	82	82	28.65	33.59	18.01	26.60	8.02
s5378	167	166	34.81	45.42	23.50	45.49	7.50
s9234	328	325	36.68	33.19	11.47	16.99	36.87
s13207	377	367	28.51	47.29	28.01	31.36	63.27
s15850	183	181	43.13	43.05	3.48	35.34	37.43
s35932	40	40	19.53	22.12	19.84	18.57	40.07
s38417	222	222	25.44	19.40	6.34	16.36	106.27
s38584	292	292	38.65	33.46	33.21	19.42	165.47
Average	211.4	209.4	31.92	34.69	17.98	26.32	45.58

Table 2.3: Post-Processing results for ISCAS-89 Circuits

the two pattern TDF tests. The unspecified entries in the relaxed tests were then filled using preferred fill values. Results of this experiment are given in Table 2.3.

In Table 2.3 after the circuit name, the number of test patterns in the original test set is given followed by the number of tests after the post-processing step. In the next four columns, the percentage reduction in peak and average WSA in the first and the second capture cycles are given. In the last column the run times for the procedure are given. Comparing the percentage reduction in WSA in Tables 1 and 2 we conclude that test relaxation followed by preferred fill achieves WSA reduction similar to those achieved by filling test cubes during test generation. However the advantage of post-processing approach is that the number of test patterns does not increase. As a matter of fact for some circuits the number of patterns can decrease as shown in the second column of Table 2.3. A disadvantage of post-processing is that it requires additional run times beyond test generation

times.

2.4 Applications to Industrial Designs

In this Section, we give a brief sketch of the modifications we made to the basic preferred fill procedure for industrial designs.

2.4.1 Signal Probability Calculations

An industrial design often contains non-scan cells, RAMs, ROMs, undriven pins, unmodeled cores, buses, bidirectional pins, and combinational loops. Additionally we need to accommodate a larger set of signal values to include 0, 1, U and Z, where U represents unknown value and Z represents high impedance state. Let *P(v)* be the probability that a signal line takes the value $v \leq 0, 1, U, Z$.

Undriven pins and unmodeled core outputs are treated as U and their signal probabilities are set as $P(0) = P(1) = P(Z) = 0$ and $P(U) = 1$. For the PIs which can take Z value, the initial signal probabilities are set as $P(0) = P(1) = P(Z) = 1/3$ and $P(U) = 0$. The signal probabilities at other signal lines are determined in a manner similar to the case when only Boolean gates and signal values 0 and 1 are needed. Now we need to compute probabilities of all four signals.

To determine the signal probabilities of the gates involved in a combinational loop, we need to calculate them iteratively. For example, the signal probabilities at gates G_1 and G_2 in Figure 2.7 are shown by using the evaluation order G_1 followed by G_2. For each line we show three signal probabilities corresponding to $P(0)$, $P(1)$ and $P(U)$. We illustrate the computations for six iterations. However, the calculation results give optimistic estimates for $P(0)$ and $P(1)$ when sequential behavior of the loop is not allowed during test generation. In our implementation, ATPG does not consider the sequential behavior of the loop. To reduce the optimistic calculation of $P(0)$ and $P(1)$, we apply the procedure given in Figure

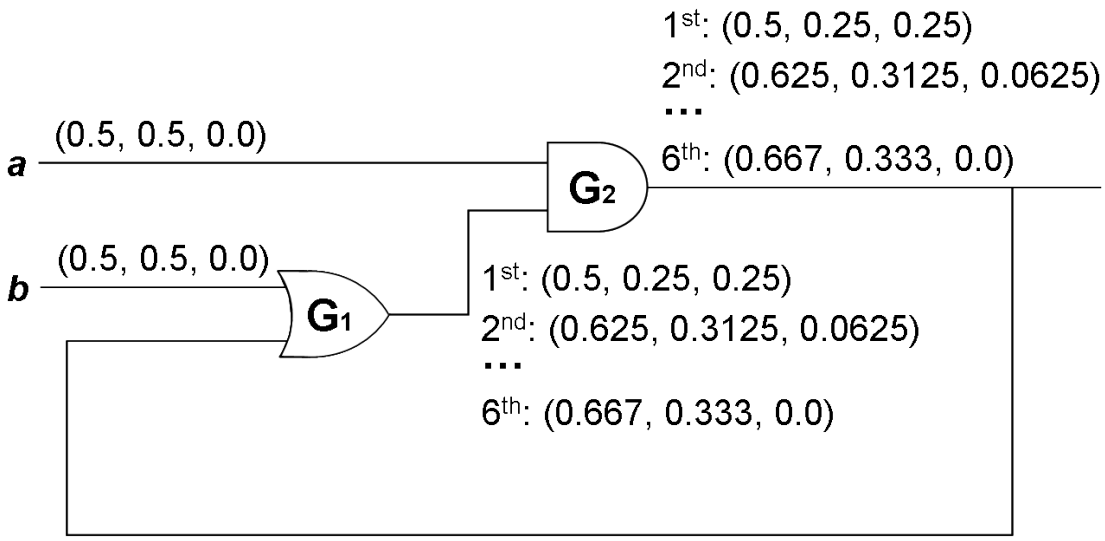

Figure 2.7: Calculate signal probabilities for a combinational loop

1. Find minimum cuts to break the loops. A cut is a signal line that breaks the feedback connections among gates.
2. Levelize the gates in the loop by assuming that the cut points are at the first level.
3. Assign probability $(P(0), P(1), P(U)) = (0.0, 0.0, 1.0)$ to all cut points.
4. Initialize two change flags cf_0 and cf_1 to be false for every cut point.
5. Calculate the probabilities iteratively:
   (a) Evaluate the probabilities of every gate in the loop according to their level order.
   (b) For each cut point: If cf_v, where $v \leq 0, 1$, is true, do not update the probability at the cut point for value v. Otherwise, if the probability $P(v)$ at the cut point's driving gate is greater than 0.0, set the probability at the cut point to be $P(v)$ and set cf_v to be true.
   (c) Stop iteration if there is no change in any cut point.

Figure 2.8: Procedure *estimate_probabilities_in_loop()*

2.8 below to estimate the signal probabilities of the gates in a loop.

For example, let us assume that the cut point for the loop shown in

Figure 2.7 is at the connection from G_2 to G_1. After the first iteration, the probabilities at the cut point are $(0.5, 0.25, 0.25)$ and both change flags are set. The second iteration sets the probabilities at G_1 and G_2 to be $(0.25, 0.625, 0.125)$ and $(0.625, 0.3125, 0.0625)$ and the calculation stops.

2.4.2 Preferred Fill for Industrial Designs

For the benchmark circuits an unspecified value in the j^{th} PPI s_{1j} was filled with 1 if $P_j(1) > P_j(0)$ and filled with 0 if $P_j(0) > P_j(1)$, where $P_j(0)$ ($P_j(1)$) is the probability that PPO s_{1j} is 1 (0). For industrial designs we filled an unspecified value in the j^{th} PPI, s_{1j} with 1 if $P_j(1) > P_j(0) + \epsilon$ and filled with 0 if $Pj(0) > Pj(1) + \epsilon$, where ϵ was set to 0.025 in our experiments.

As noted earlier methods to fill unspecified values in test cubes to reduce WSA typically increases the test pattern counts compared to the case when the unspecified values are randomly filled [42][43][44]. The same was observed when preferred fill is used for industrial designs. In order to reduce the increase in test pattern count using preferred fill we experimented with first filling some percentage of unspecified values randomly followed by preferred fill. We call this technique *limited preferred fill*.

2.4.3 Use of Signal Probabilities for Test Generation

We also experimented with the use of signal probabilities to guide the ATPG to generate tests that cause lower WSA during capture cycles. We used signal probabilities to guide line justification step of the ATPG. Among the signals which when set to a value justifies a desired line value we pick the signal with the highest probability. The idea behind this heuristic is similar to the idea behind preferred fill. We report the results of using the test generation procedure with this modification.

2.4.4 Calculation of WSA with Unknown Values

The X-sources in the industrial designs make some gates have unknown values during good machine simulation. The unknown values can potentially create transitions at the internal gates. Ignoring them completely will underestimate the power dissipation in capture cycles. In this work, we computed WSA at a gate g using the formula given below.

$$WSA_g = \alpha * (1 + fanout\_number_g)$$

- When g has unknown values in both the previous time frame and the current time frame, α is equal to 0.125.
- When the gate g has unknown value in the previous time frame and known value in the current time frame, α is equal to 0.0625.
- When the gate g has known value in the previous time frame and unknown value in the current time frame, α is equal to 0.0625.
- If the gate has known values in both the time frames that create a transition then α is equal to 1.

2.4.5 Experimental Results

The proposed techniques were integrated into a commercial ATPG tool and several industrial designs were used to evaluate their effectiveness.

In Table 2.4, we show the test generation results using random fill during test generation. This is the standard test generation flow. After the circuit names, we show the numbers of transition faults in millions and the test pattern counts under the columns *#Flts* and *#Pat*, respectively. The average WSA and the peak WSA for the two capture cycles are given under the columns $1^{st}Capture$ and $2^{nd}Capture$.

The results using preferred fill are shown in Table 2.5. Two sets of data

are given in Table 2.5. The first set of data, given under *Preferred Fill* columns, corresponds to the case when preferred fill alone is used without modifying the test generation procedure. The second set of data, given under *Preferred Fill + ATPG* columns, is for the case when test generation procedure is modified by employing signal probabilities to guide line justification step of the test generation procedure. In Table 2.5 we give the results for pattern counts and WSA as a percentage increase or decrease relative to their respective values when random fill is used and given in Table 2.4. In column *Pat. Inc.* we give the percentage increase in pattern count. The percentage decreases in average WSA and peak WSA for each capture cycle are given under the columns *Ave.* and *Peak*, respectively. It can be seen that, on average, when only preferred fill is used, the average WSA and the peak WSA are reduced by 64.29% and 58.43% in the first capture cycle and by 46.90% and 53.55% in the second capture cycle. The test pattern count is increased by 144.83% on average. These results illustrate that random fill helps hold down test pattern counts, but it creates a larger amount of internal gate switching activity leading to much higher power dissipation during test. *Preferred fill* reduces the switching activity dramatically. However pattern counts are increased substantially.

The data in Table 2.5 for the case using signal probabilities in test generation together with preferred fill given under Preferred Fill + ATPG columns show that both peak and average WSA can be further reduced if signal probabilities are used to guide ATPG. Additionally pattern counts do not increase as much as when only preferred fill is used.

To moderate the pattern count increase when preferred fill is used, we apply the limited preferred fill where we first fill a certain percentage of unspecified values of PPIs randomly and fill the remainder of unspecified values using preferred fill. We report the results of these experiments in Table 2.6 for the case when up to 10% of the unspecified values are filled using random fill. The data in Table 5

Circuit	#Flts (Millions)	#Pat.	1st Capture		2nd Capture	
			Average	Peak	Average	Peak
ckt1	0.473	15034	143754	266527	99387	219534
ckt2	0.745	17447	70620	228801	34697	130912
ckt3	1.317	6391	293044	673924	245466	574517
ckt4	2.207	21665	638073	1216694	512557	920930
ckt5	2.284	60748	171778	950634	138212	894014
ckt6	3.689	12666	704288	828964	441437	603281
ckt7	5.202	50720	764327	1550568	751004	1816088

Table 2.4: WSA for ATPG with random fill

Circuit	Preferred Fill					Preferred Fill + ATPG				
	Pat. Inc.	1st Capture		2nd Capture		Pat. Inc.	1st Capture		2nd Capture	
		Ave.	Peak	Ave.	Peak		Ave.	Peak	Ave.	Peak
ckt1	66.61	45.77	36.74	25.81	32.42	80.07	47.87	33.51	27.80	24.99
ckt2	133.13	60.04	62.47	33.80	46.16	29.20	58.31	66.65	38.20	44.07
ckt3	158.07	70.16	56.51	69.10	61.82	81.99	74.00	44.29	69.72	55.47
ckt4	170.83	79.35	66.95	43.32	51.95	182.20	80.61	66.41	44.60	55.13
ckt5	219.12	67.21	66.38	50.69	66.88	303.39	63.32	66.65	44.26	68.67
ckt6	95.57	59.94	51.63	47.56	44.69	181.33	71.62	63.20	52.76	46.88
ckt7	170.51	67.56	68.32	58.04	70.94	25.88	69.54	71.96	58.62	72.95
Average	144.83	64.29	58.43	46.90	53.55	126.29	66.47	58.95	47.99	52.59

Table 2.5: WSA reductions with preferred fill and ATPG

Circuit	Preferred Fill + 10% Random Fill					Pref. Fill + ATPG + 10% Rand. Fill				
	Pat. Inc.	1st Capture		2nd Capture		Pat. Inc.	1st Capture		2nd Capture	
		Ave.	Peak	Ave.	Peak		Ave.	Peak	Ave.	Peak
ckt1	23.99	36.87	26.49	21.61	25.28	34.18	41.78	26.46	25.86	24.14
ckt2	28.10	38.40	40.10	4.31	22.11	10.11	36.46	43.96	17.36	20.21
ckt3	25.63	58.50	41.42	54.20	48.31	27.15	62.43	38.30	60.36	49.47
ckt4	36.23	59.05	52.27	27.79	43.69	90.80	73.28	52.06	39.37	52.95
ckt5	59.76	40.72	51.90	38.52	54.44	105.85	48.45	54.45	39.32	58.86
ckt6	51.03	51.75	44.63	42.55	35.78	111.25	66.56	61.57	50.39	43.58
ckt7	38.00	49.19	52.63	47.85	65.85	-14.53	57.41	60.88	51.42	67.66
Average	37.53	47.78	44.21	33.83	42.21	52.12	55.20	48.24	40.58	45.27

Table 2.6: WSA reductions with limited preferred fill

is arranged in a manner identical to that in Table 2.5. From Table 2.6 one can notice that using limited preferred fill only the test pattern counts increase, on average, by only 37.53% instead of 144.83% when preferred fill is used without limited random fill, the average percentage reduction in average WSA for the first capture cycle is however reduced to 47.78% from 64.29%. Similar reductions in the percentage reductions of peak as well as the percentage reductions of WSA during second capture cycle can be noted. From the second set of data in Table 2.6, one can also note that the percentage increase in pattern counts is moderated when limited preferred fill is used with modified ATPG procedure. These results show that a good tradeoff between power dissipation and test pattern count is achieved using *limited preferred fill*.

2.5 A Post-Processing procedure for low power test

In this SubSection details how to apply *preferred fill* as a post-processing method are given. *Preferred Fill* will be combined with Adjacent fill to reduce

both capture and shift power. Experimental results for both SAFs and TDFs for the post-processing procedure will be shown both for benchmark and industrial circuits. This will prove the effectiveness of *preferred fill* for different fault models.

2.5.1 Details of the Proposed Procedure

The procedure takes a set of fully specified tests T and modifies the tests in T to obtain a new test set T' with reduced WSA during scan shift as well as during capture cycles. The procedure has the following steps:

Step 1: Order the tests in a given set of fully specified tests.

Step 2: Fault simulate the ordered set of tests and note the faults detected by a test.

Step 3: Relax, fill (using PF) and fault simulate the tests in the order they appear in the ordered test set.

Next we give details of how the steps given above were performed.

Step 1: The tests in a given set of tests T are ordered in decreasing order of WSA caused by the first capture cycle of the test application. Note that for stuck-at tests there is only one capture cycle and for delay tests (we used LOC tests) there are two capture cycles and we use the WSA for the first capture cycle in this case for ordering tests. Let the ordered test set be T_0.

Step 2: The tests in T_0 are simulated in reverse order using fault dropping. When a test t_i is simulated the set of faults F_i detected by it are recorded. F_i is called the target fault set of t_i.

Step 3: This step has three procedures which are iterated. Next we give these as Steps 3.1, 3.2 and 3.3. We also discuss the choices we made in each step.

Step 3.1: Remove the test t_i from the top of the ordered test set T_0 and relax it by un-specifying some of the specified entries in the test. When we say relaxing a test we mean relaxing the initialization vector of a two-pattern test for TDFs or the

test for stuck-at faults. Methods to relax fully specified tests are given in [49][50]. We used the procedure of [49] in our work. The original procedures in [49] and [50] considered relaxing stuck-at tests. We extended the procedure in [49] to relax the initialization vector V_1 of a two-pattern LOC test (V_1, V_2).

Relaxation of tests must be done such that the fault coverage by the modified tests is not lower than that of the original test set. This is achieved by providing a target fault set, which must be detected by a relaxed test. In [51] the target fault set for a test is determined by using what is called double detection fault simulation [51]. Double detection fault simulation drops faults only after they are detected two times. In [49] the set of target faults for a test are determined by simulating a fully specified test when it is picked for relaxation. In our procedure, initially the set of target faults for each test are F_i found in Step 2 given above. Reverse order simulation of an ordered test as done in Step 2 to determine sets of target faults has several beneficial effects as discussed in Observation 1 given in this section after describing Step 3.2 and Step 3.3.

Step 3.2: After relaxing a test we fill 50% of the unspecified values of PPIs using their preferred fill values and the remaining are filled next using adjacent fill. The PPIs whose values are filled with preferred values are selected randomly. If the WSA of the (first) capture cycle of the modified (LOC) test is higher than that for the unmodified test the modified test is discarded and the corresponding original test is retained in its place.

Step 3.3: Fault simulate the test filled in Step 3.2 and drop all detected faults from the target fault sets for each test in T_0. The test set T_0 is updated by deleting tests with empty target fault sets. Steps 3.1 to 3.3 are repeated if T_0 is not empty.

Observation 1: The rationale behind the test ordering we use in Step 1 above is the following. In Step 3.1 when a test is relaxed it is clearly advantageous to relax maximum number of entries in it. The extent to which this is done depends

on the order in which the tests are processed and the sets of target faults of tests. For a test t_i that appears earlier in the ordered test the number of faults in its target set F_i are smaller since the target faults are determined by reverse order fault dropping fault simulation in Step 2. Hence the tests at the top of the ordered list which originally have high WSA have smaller sets of target faults. Thus, during relaxation we expect to maximally relax these tests. Additionally as we continue relaxing tests from the top of the ordered test set the sets of target faults for the remaining tests keeps reducing (in Step 3.3) which again allows for maximal relaxation of the tests.

Observation 2: In Step 3.2, one can fill different proportions of unspecified values with preferred and adjacent fills. For the ISCAS-89 benchmark circuits equal proportions for preferred fill and adjacent fill gave the best results in simultaneously reducing WSA both during scan shift and capture cycles of stuck-at and TDF tests.

Observation 3: In the procedure given above some tests in the original test set T may be dropped. A test t_j is dropped if in Step 3.3 its set of target faults is found to be empty during updating of test set T_0.

2.5.2 Experimental Results

The proposed method was implemented in C++ language and experiments were conducted on a Pentium XEON 2.8 GHz PC with 1.5 GB of RAM. For ISCAS-89 benchmark circuits, the original fully specified stuck-at test sets are from [52] and the original LOC test sets for TDFs were generated using an academic tool.

In Table 2.7, we give the results of applying the proposed procedure to stuck-at fault test sets for ISCAS-89 circuits. After the circuit name we give the number of tests that remain after the proposed procedure is used. Then we give the WSA reductions in peak capture and in average scan shift under the columns *Peak Capture* and *Average Shift*, respectively. The run time is shown under the

Circuit	#Tests	WSA Reduction %		CPU (Sec.)
		Peak Capture	Average Shift	
s1423	24	22.40	12.39	0.15
s5378	100	35.26	26.00	3.45
s9234	111	16.56	26.79	7.40
s13207	235	41.78	27.46	21.63
s15850	97	32.31	36.24	12.8
s35932	12	0.15	15.44	7.13
s38417	87	23.74	29.40	24.34
s38584	114	48.79	29.47	38.10
Average	97.5	27.62	25.40	14.38

Table 2.7: WSA reduction for stuck-at tests

column *CPU* in seconds. In Table 2.8, we give similar data as in Table 2.7 for LOC transition fault tests. The reductions in peak capture WSA for each of the two capture cycles are given under the columns 1^{st} and 2^{nd}, respectively. In the second column of Table 2.8, the numbers shown in the parenthesis are the differences in test pattern counts between the tests after applying the proposed procedure and the original tests.

From Tables 2.7 and 2.8, it can be seen that the proposed procedure reduces both the peak WSA during capture cycles and the average WSA during scan shifts for all the circuits. For stuck-at faults, on average, the reductions in the peak WSA of capture cycles and the average WSA of scan shifts are 27.62% and 25.4%, respectively.

As we discussed earlier, it is important to reduce average switching activity during scan shift and the peak switching activity caused by capture cycles for scan tests. However, for the sake of completeness we computed the reduction in the

Circuit	#Tests	WSA Reduction			CPU (Sec.)
		Peak Capture		Average Shift	
		1st	2nd		
s1423	82	28.69	25.52	34.71	7.17
s5378	165(-2)	26.97	25.89	16.19	7.42
s9234	324(-4)	35.79	14.10	45.84	38.15
s13207	368(-9)	31.07	26.64	32.28	66.57
s15850	181(-2)	39.49	8.15	54.40	40.37
s35932	40	16.24	21.85	31.53	41.48
s38417	222	26.81	14.80	44.35	109.12
s38584	292	36.65	37.78	51.11	173.82
Average	209.5	30.21	21.84	38.80	60.51

Table 2.8: WSA reduction for TDF tests

average WSA of capture cycles and the peak WSA of scan shifts for TDF tests. This data is given in Table 2.5.2. It can be seen that both the average WSA of capture cycles and peak WSA of scan shifts also reduced when LOC tests are modified using the proposed procedure.

Next we compare the peak WSA reduction of capture cycles for stuck-at tests achieved by the proposed method and the method in [37]. Both methods use post-processing steps on tests obtained a normal run of ATPG using random fill of unspecified values in test cubes. The original test sets used by the post-processing steps in both cases are the highly compact test sets from [52]. It should be pointed out that the method in [37] may increase the test pattern count and it focuses on reducing the peak WSA of capture cycles. Instead, the proposed method simultaneously reduces both the WSA during capture and the WSA during scan shift without increasing test pattern count. Moreover, the method in [37] measures

Circuit	Ave. Capture WSA %Red.		%Red. in Peak WSA of Scan Shift
	1st	2nd	
s1423	32.96	30.97	10.98
s5378	42.31	44.59	6.86
s9234	33.38	16.66	14.08
s13207	41.76	26.21	22.65
s15850	40.99	34.72	3.40
s35932	18.61	13.58	6.93
s38417	24.02	19.95	26.00
s38584	33.11	21.08	25.51
Average	33.39	25.97	14.55

Table 2.9: Reduction in average WSA of capture cycle and peak WSA of scan shift for TDF tests

the power reduction during capture cycles by the number of state element changes in the capture cycles. For this reason we also computed the numbers of state element changes when the proposed method is used. In Table 2.10, we give the results from [37] and the results by using the proposed method with 50% preferred fill and 50% adjacent fill of unspecified values in relaxed tests. After the circuit name, we give the number of test patterns, the reduction in the peak number of the state element changes during capture cycles and the run times in seconds. The results from [30] are given under column heading with [37] and the results for the proposed method are given under *PF+AF*. We highlight the entries for the number of test patterns and the reduction percentage of the number of state element changes when they are the larger of the two methods. It can be seen that the proposed method achieves better reduction of the number of state element changes for five out of eight circuits and always have smaller or equal test pattern count for all the circuits. On average,

Circuit	#Tests		%Red. of Peak SC		CPU (Sec.)	
	[30]	PF+AF	[30]	PF+AF	[30]	PF+AF
s1423	27	24	40.82	32.65	7.6	0.15
s5378	106	100	16.67	36.27	20.3	3.45
s9234	133	111	20.16	16.94	62.7	7.40
s13207	235	235	24.74	44.21	66.9	21.63
s15850	101	97	40.07	59.57	114.2	12.80
s35932	13	12	40.44	18.86	114.4	7.13
s38417	91	87	16.78	26.61	224.9	24.34
s38584	121	114	50.38	58.27	479.5	38.10
Average	103	98	31.26	36.67	136.3	14.38

Table 2.10: Comparison between [30] and the proposed method

the proposed method reduces the peak state element change by 36.67% while the method of [37] reduces it by 31.26%. Furthermore, the run times for the proposed method are much lower than those for the method in [37].

The results after applying the proposed method to industrial circuits are given in Table 2.11. The results are given when 50% preferred fill and 50% adjacent fills are used for stuck-at fault tests. After the circuit name we give the numbers of faults and scan cells in the circuits. Next we give the percentage reduction in average and peak WSA of capture cycles. Due to extremely high run times to compute WSA during scan shift operation, the number of changes in the states of scan cells or equivalently the number of transitions of states in scan cells during scan shift is counted instead. We report the percentage reductions in the average and peak number of state transitions during scan shift in the last two columns. From Table 2.11, it can be seen that the peak WSA during capture is reduced by

Circuit	#Flts (Mill.)	#Scan Cells	50% PF + 50% AF			
			Capture WSA Reduction %		Scan Shift Transition Reduction	
			Average	Peak	Average	Peak
ckt1	0.63	20.2K	39.67	52.95	96.77	86.28
ckt2	1.85	70.3K	46.46	48.37	97.49	49.69
ckt3	2.84	44.9K	51.00	41.96	90.51	81.21
ckt4	4.17	133.7K	54.51	49.20	95.35	68.24
Average			47.91	48.12	95.03	71.36

Table 2.11: Results for industrial circuits

48.12% on average while the average percentage reduction in the number of state transitions of scan cells during scan shift is reduced by 95.03% on average. These results demonstrate the effectiveness of the proposed method to large industrial designs.

2.6 Conclusions

In Section 2.3 a new technique called *preferred fill* is presented to address the problem of larger than normal peak current and power dissipation during the fast capture cycles of broadside delay fault testing. Preferred fill uses circuit signal probabilities to fill unspecified values in test cubes. Since the signal probabilities can be computed once in a preprocessing step preferred fill fills all the unspecified values in a test cube simultaneously. The time to compute signal probabilities and hence preferred fill are negligible and hence the method is scalable to large designs. Since preferred fill is used only to fill unspecified values in test cubes, achievable fault coverage is not affected. *Preferred fill* is shown to achieve substantial reductions in peak and average power dissipation in benchmark and industrial

circuits.

Preferred Fill was proven to be applicable to the stuck-at fault model in Section 2.5. Also when used in the post-processing scheme it does not increase test set size. The method in its two forms requires no hardware overhead. When combined with adjacent fill it can provide reductions in both capture and shift power. In both its forms it can be easily integrated to existing ATPG flows.

The main trade-off between the potential forms of using *preferred fill* is that if it is used to replace random fill it requires no extra computing effort but pattern count increases. On the other hand if it isused as a post-processing method computing effort increases substantially due to the required relaxation step but pattern count is held constant or even reduced.

Further research in this topic should focus on how to compatibilize *preferred fill* with existing compression techniques like EDT or broadcast scan.

CHAPTER 3
TEST POINT INSERTION

As digital circuits grow in gate count so does the data volume required for manufacturing test. To address this problem several test compression techniques have been developed. This paper presents a novel and scalable technique for inserting observation points to aid compression by reducing pattern count and data volume. Experimental results presented for industrial circuits demonstrate the effectiveness of the method.

3.1 Introduction

Test point insertion has been studied in the literature for a long period of time. Most of the previous work done on test point insertion (TPI) has been in the area of logic Built-In Self Test (BIST) [12][13]. The goal of BIST oriented TPI is to decrease the pattern count needed to achieve the desired fault coverage by increasing the testability of hard to detect random pattern resistant faults. Mainly, two techniques have been used to identify target sites for test point insertion: exact fault simulation [53][54] and approximate testability measures [55][56][57].

In recent years, with the invention and commercialization of test compression techniques such as EDT [14] and Broadcast Scan [58], logic BIST has become less utilized and deterministic test gained more importance.

When compression, along with ATPG, is utilized the relevant variables that determine its performance are the fill rate (percent of specified bits in test patterns) of the patterns created by the ATPG engine, the number of patterns and the total data volume, defined here as the sum of the specified positions before X-fill in every test cube in the test set. These parameters determine to what extent the scan chains should be segmented for compression and this in turn determines the maximum

achievable compression ratio. The usual result after applying compression is that the pattern count is similar to ATPG without compression, but the data volume is reduced and hence tester time needed for the circuit under test (CUT) decreases.

Thus, two objectives gain relevance for test point insertion in deterministic test: enhanced compaction (pattern count reduction) and data volume reduction. Pattern count reduction impacts test cost by directly reducing test time. Each pattern takes a fixed amount of time to be loaded from the tester to the CUT, depending on the maximum length of the scan chains. If fewer test vectors are needed, less tester time will be used. Fill rate reduction, when pattern count remains fixed, allows higher compression ratios by properly adjusting the maximum scan chain length. This in turn results in shorter test application time per pattern and, again, a reduction in tester time utilization. The objective of our work is to develop a technique that will identify locations for observation points (OPs) that enhance pattern compaction, i.e. reduce the pattern count, and reduce the specified bits prior to X-fill.

The remainder of the chapter is organized in the following manner: Section 3.2 reviews the related existing literature in TPI. Section 3.3 explains how to find the possible locations for OPs and describes the proposed method. Section 3.4 gives experimental results based on industrial designs, Section 3.5 gives experimental results on large ISCAS circuits and Section 3.6 concludes the paper.

3.2 Previous works

Previously, some papers have been published that insert test points (TP) with the objective of enhancing compaction to reduce test application time [59][60][61][62][63]. They will be described in this section.

In [59], a method was proposed that combines approximate testability measures from Controllability/Observability Program (COP) and Sandia Control-

lability/Observability analysis program (SCOAP) and Test Counts, which is the number of times a line must become zero (one) during the application of a test set. Three methods are derived and compared, one based on COP, another on SCOAP and a third that integrates Test Counts. The type of test point inserted is a transparent scan cell at the output of a gate. A transparent scan cell behaves as a buffer during the circuit normal mode and as a scan cell in test mode. The authors ran experiments for the three methods on ISCAS circuits and for the COP and Test Count methods on small industrial designs using the single stuck-at fault model, inserting in industrial designs around one TP per thousand gates (more TPs in the smaller circuits). For the industrial designs an average reduction in pattern count of 36.39% for the COP method and 39.26% for the Test Count method was obtained. The average time required to compute the Test Count test points was 59.49% of the ATPG time required for the baseline (pre-TP insertion) ATPG run, but the ATPG time after TP was reduced to 43.58% of the original. In [60] an extension of the methods in [59] to work with transition faults is given by the same authors. Results are obtained for ISCAS circuits and a subset of the industrial designs of [59].

In [61], a method for inserting OPs to enhance test compaction for the single stuck-at fault model is presented. The method starts from a compact test set and reduces it by combining two of its test vectors (τ_0, τ_1) into one. It achieves this by targeting the faults detected by both τ_0 and τ_1 with a new test pattern $\tau_{0,1}$ and adding OPs to detect the activated but unobserved faults. If every fault detected by τ_0 and τ_1 is detected by $\tau_{0,1}$ after inserting observation points, the two initial tests are replaced by $\tau_{0,1}$. Results are given for some ISCAS benchmark circuits.

In [62] and [62] the authors introduce a method based on fault detection probabilities and value assignment probabilities to enhance compaction of patterns. The authors propose algorithms to search the circuit lines for test point locations,

both control and observe, based on these metrics. In both works, a set of three industrial designs are used to prove the effectiveness of the method. In [63] the designs are broken into their constituent blocks and test points are inserted in the blocks with the highest pattern count. The main purpose was to reduce tester time, measured as a function of test pattern count, achieving a reduction of 33.62% in this metric.

Another work [64], does not share the same objective of improving compaction in deterministic test, it is designed for BIST but it introduces several concepts relevant for TPI. Also, it is implemented in a commercial tool which is available and will be used for comparison purposes. Given these facts, and for the sake of completeness, some of its key new concepts are discussed here. Multi-Phase Test Point Insertion (MTPI) utilizes the AND-OR type of control point. It introduces the control points at the outputs of gates to control the value of a stem and in this way improve its controllability. In MTPI several control points are controlled by the same input. This reduces the extra logic that needs to be inserted. The test patterns are divided into a small number of phases. During each phase a subset of control points are enabled, increasing the testability of certain faults (possibly blocking others), while the rest are disabled. OPs on the other hand, are always enabled. To reduce the number of scan cells needed to capture the OPs values, several observation points are wired together through XOR gates (this concept is not novel to MTPI). To find the test point locations MTPI simulates a small number of random patterns and based on them it calculates signals with low controllability as potential control points and signals with low observability as potential OPs.

3.3 The proposed method

The principal step in OP insertion is the addition of a branch to the output of a gate. This branch in turn feeds an extra output of the circuit, preferably a scan

cell and not a primary output. The addition of a scan cell to the circuit introduces an area penalty. When the number of scan cells added to a circuit becomes large this area penalty may become unacceptable due to design constraints. Due to this there is a limit on the number of OPs that can be inserted, and not all gates can be chosen. This calls for careful selection of the observation points. Since an XOR gate takes less area than a scan cell, two OPs can be observed together using an XOR gate. In this way more observation points can be inserted with similar area overhead. We will not discuss the problem of how many OPs can be inserted because it depends on design constraints. Instead, we will focus on finding a heuristic that successfully identifies the best observation points for a given circuit in an orderly manner. The first identified OPs should give the greatest benefit in compaction and data volume reduction. Progressively adding more OPs would add diminishing returns.

OPs cannot be used to reduce pattern count for the path delay or the transition fault models. For path delay faults, an OP will observe only the first part of the path. The industry currently uses the transition fault model to also detect small delay defects. This requires propagating the fault effect (FE) through a path that has the smallest slack. An OP will reduce the combined delay of the fault and the gates that the FE traverses, increasing in this way the minimum delay that the test set can detect. This makes the insertion of OPs risky for these fault models, therefore we will not analyze the effectiveness of OPs for them.

In this work we will focus on identifying OPs for the single stuck-at fault model. We will target the SAF fault list of a circuit using combinational patterns. The remainder of this paper assumes stuck-at faults and combinational test patterns in the discussion.

1. For each fault targeted as a primary objective do:
   (a) Compute Propagation Path from PO to fault site
   (b) Compute Essential Input Assignments
   (c) Compute Propagation Input Assignments per gate in the Propagation Path
   (d) Store in database

   When ATPG/data gathering phase is complete:

2. For the desired number of observation points do:
   (a) Check each gate cost function
   (b) Select OP as the gate output with higher cost function
   (c) Update database removing assignments after OP

Figure 3.1: OP selection process flow

3.3.1 Overview of the proposed procedure

The proposed procedure first runs ATPG on the circuit. When a cube is created for a target fault it is immediately analyzed to extract one of the FE propagation paths. Then the input assignments needed to propagate the FE through the propagation path are identified. These input assignments are divided into two categories: essential and propagation, defined later. With this information a database is created. After the ATPG is done with test generation a post-processing step on the database is done to select OPs. After each OP selection the database is updated to reflect the presence of the newly identified OP. When the desired number of OPs is identified they are inserted into the circuit and ATPG is run again to create the final test set. Figure 3.1 shows an overview of the flow of this process.

If the ATPG heuristics that choose the propagation path and the gate input to justify a required value are changed, the information contained in the database

will change and the OPs inserted will be different.

3.3.2 Creating a database from the ATPG run

The goal for this step is to create a database of input assignments for each test cube that is created when ATPG is targeting a fault as a primary objective. Each input assignment will be associated with a gate in the FE propagation path. This database will be used to select the OPs.

Every time a fault is targeted by ATPG as a primary target and a cube is successfully created for it, the propagation path of the fault is traced backward from the detection gate, PO or pseudo-primary output (PPO) of a scan cell, to the fault site. Only the first encountered propagation path is recorded, even if the fault has multiple paths. The tracing operation is linear in the circuit depth.

Once one FE propagation path is found we record for every gate in it the input, meaning both primary inputs and pseudo-primary inputs, assignments necessary to propagate the fault effect to the gate output. In the case of the fault site, and if the fault is at one of the gate inputs, we combine the activation assignments with the assignments made to propagate the fault to the gate output. The reason behind this is that a fault should not be observed at its fault site. The earliest point of observation is the output of g. This is because the only way to insert an OP is at the output of a gate, if we try to insert an OP at a branch a new fault will appear between the OP and the gate input which is fed by the branch, which is equivalent to say that the OP is at the stem. We will call the set consisting in the union of the activation input assignments and the propagation input assignments needed to propagate the FE to the output of the fault site *essential* input assignments. The rest of the input assignments will be called *propagation* input assignments and the will be associated with a gate in the FE path.

Observation 1: Since we need to propagate a FE to a gate output to observe it, *essential* input assignments cannot be avoided with OPs. On the other hand, *propagation* input assignments can be eliminated with OPs, since the OP eliminates the need to sensitize part of the observation path.

To find the input assignments that a gate needs to propagate a FE, we trace every input of a gate in the identified propagation path backwards. If a gate has a controlling value at its output one or more of its inputs is set to a controlling value. In this case, we will mark as required the first input that is set to a controlling value. If during this process we encounter a gate g that was already transited to justify a value for a gate that appears earlier in the propagation path, we stop tracing. In this case the inputs assigned to justify a value in g are needed to propagate the FE through multiple gates but we only assign those inputs to the gate in the propagation path of the FE that has the lower gate level. We do this because if an observation point is added in the propagation path to remove some of the input assignments the ones belonging to a gate with a lower level will still be necessary even if they are not needed in the upper level gate.

Observation 2: By ending the tracing at a previously traced gate, the operation becomes linear in the number of gates of the input cone of the detection gate.

Observation 3: If a fault has multiple propagation paths, every propagation path after the first can be treated as assignments needed in the faulty circuit that will map into assignments needed in the good circuit eventually.

In Figure 3.2 we illustrate the process of mapping a faulty circuit value into required good circuit values. In this figure, required values are underlined and the first propagation path of the fault effect is in italics. First it is important to observe that to activate f we need a zero in one of the inputs of g_1, this is the *essential* assignment for f. We can consider this a case in which a faulty circuit value (0/1 at the output of g_1) maps into a required good circuit value (0 at one

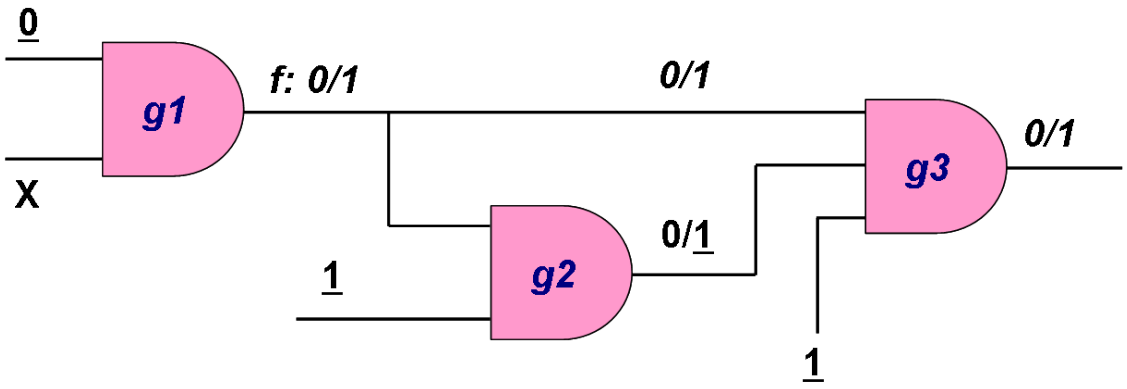

Figure 3.2: Mapping faulty circuit to good circuit

of its inputs). We consider the propagation path $< g_1, g_3 >$ as the primary and the other ($< g_1, g_2, g_3 >$) as assignments in faulty circuit that will map by tracing back into assignments in good circuit or paths that will end up in the fault site. If we analyze Figure 3.2 we see that to propagate the fault effect through g_3 only the one in the fault circuit is needed at the output of g_2, and that the zero in the good circuit is unnecessary. Then if we trace back the one in the faulty circuit at the output of g_2 we find the fault site and a one in good circuit. This process can always be applied since the presence of any faulty circuit value can be explained by the presence of the fault and the input assignments, which are good circuit assignments.

After this process is complete no faults remain in the fault list, because either they were targeted by ATPG or dropped by fault simulation. For every fault that was targeted as a primary objective and a cube was created for it we have one of its propagation paths and for every gate g in the FE propagation path we have the inputs that were assigned to a logic value to propagate through g.

Figure 3.3 illustrates the process of creating a database from the cubes created by the ATPG. Fault f is located at input a of gate G_1. The sets of input assignments A and B of the cones feeding a and b cannot be avoided by inserting OPs. Thus the

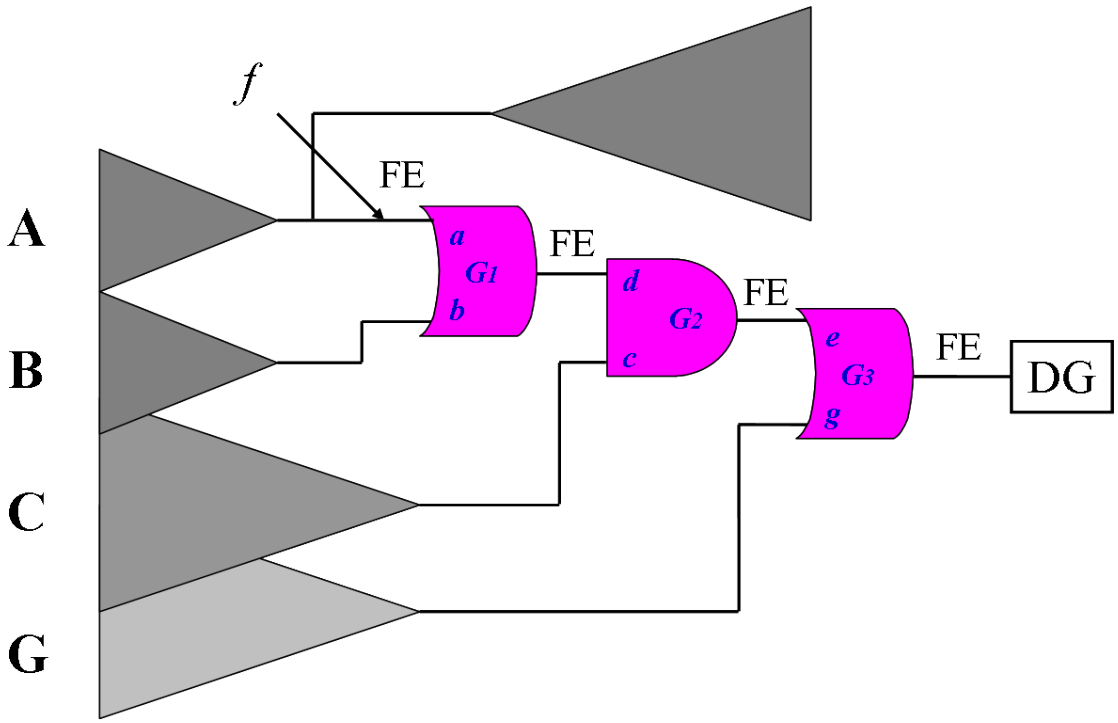

Figure 3.3: Input assignment identification process

sets of inputs assignments A and B will be identified as *essential* to f and the sets C and G will be classified as *propagation* input assignments. The set of assignments $C - A \cup B$ will be associated with G_2 since it is the required set of propagation input assignments to propagate through G_2. The set of assignments $G - A \cup B \cup C$ is associated to G_3 for the same reason. If we observe the output of gate G_2, we will not need $G - A \cup B \cup C$, the set of inputs assignments to set g to a zero that are not required for setting a value in a gate of lower level in the FE path. Also, if we observe the output of gate G_1 the input assignments $G \cup C - A \cup B$ are not needed to detect the fault. In this example the best OP will be the output of gate G1 but given the presence of multiple faults and reconvergent fanout in the circuit several contributions coming from faults in different parts of the circuit changes the outcome.

Next we introduce two methods to locate OPs. One is designed to reduce

pattern count and the other to reduce data volume, defined as the sum of the specified inputs in all test cubes before random fill.

3.3.3 Conflict reduction (CR) oriented method

Two faults cannot be targeted with a single test when a conflict exists in the assignments needed to activate and propagate them to an output. This inter-fault conflict differs from the usual meaning of the term conflict, which is between assignments to create a test for a target fault. Here the word conflict is used to refer to the inter-fault conflict.

Given a circuit input and a set of test cubes for every circuit fault (the database), the number of compatibility conflicts created by the cubes in it is $\min E_0 + P_0, E_1 + P_1$, where E_0 is the number of times the input was set to 0 to excite the fault and to propagate it to the output of the gate which the fault is associated with *i.e.* the number of the cubes with zero essential input assignments in that input, and P_0 is the number of times the input was set to 0 to propagate through a gate other than the one at which the fault is located *i.e.* the number of cubes with zero propagation input assignments. All avoidable conflicts can be eliminated by observing every gate output in the circuit, and then the remaining conflicts will be $\min E_0, E_1$. Then for a given input the number of avoidable conflicts will be $\min E_0 + P_0, E_1 + P_1 - minE_0, E_1$.

The database records the extra input assignments needed to propagate the FE through every gate in the FE propagation path. If an observation point is inserted at the output of a gate g, all the input assignments that are needed to propagate a FE in every cube through an upper level gate are not needed any more. We will consider that a conflict is removed only if the input assignment removed is of the same logic value as $\min E_0, E_1$ because this is the minimum achievable conflicts for that input.

Now we can compute for every gate in the circuit how many conflicts will be removed if its output is observed. We will select the output of a gate g that will remove the most conflicts for every circuit input as our first OP. After this we will search through the FE propagation path of every cube and remove all the input assignments of gates in the propagation paths that come after g. Then we will repeat this in the actualized list of FE propagation paths and inputs assignments until we have identified the desired number of OPs. In this work we identified up to one OP per five hundred gates in the circuit.

3.3.4 Data volume reduction (DVR) oriented method

Using the same information from the FE propagation paths we derived a second technique oriented at reducing the number of specified positions in the test cubes before random fill. After collecting the information from the cubes we will assign to each gate output a cost function. This cost function is simply the sum over all the cubes of the number of input assignments that will become unnecessary if that gate output were to be observed. The gate output with the highest cost function will be selected as the first OP and the input assignments that come after that gate will be removed from the cubes. After that the second iteration will select the second best OP and so on.

Both methods yielded very similar results due to the high correlation in the sets of OPs identified. The more input assignments that an observation point will remove the more likely that those removed input assignments will remove input conflicts. The data volume reduction (DVR) method runs faster ($\tilde{4}$X) than the conflict reduction (CR) method. This is the only significant difference in the results for the two methods.

3.4 Experimental results on industrial designs

All the results in this section are using the single stuck-at fault model tested with combinational patterns. In all instances Test Coverage (TC) was the same, or slightly higher, than without TPs. The insertion of OPs never reduced TC and any increase, when it happened, although beneficial, is not reported because it is not the target in this work. Only relative run times are given since absolute times vary from one computer to another. Also the relative run time results have some noise due to the varying loads on the computers running the experiments. All results shown were obtained using a commercial ATPG tool.

In Table 3.1, the circuit name is given in the first column. The gate count in the circuit under test rounded to the nearest thousand is given in the second column. After this we show the ratio between time spent in observation point identification for the CR method and the ATPG time. The other two columns show respectively the original pattern count and data volume, measured in bits before filling the unspecified values, needed to test the circuits. It can be seen from Table 3.1 that the OP identification process will take approximately twice the time of a regular ATPG run. In the last row N/A means non-applicable.

In Table 3.2, one of the circuits for which the ATPG runs faster (due to its small size) is used to analyze the effectiveness of the method and how much additional benefit we can obtain from inserting progressively more OPs. In Table 3.2, the first column has the number of observation points, and then under the test case name there are two columns: the first is the number of patterns that the ATPG produced and the second is the specified bits before X-filling of the entire test set. Here we employed the CR method to identify OPs. It can be seen from Table 2 that as more OPs are inserted further reduction in pattern count and data volume (sum of specified bits) is achieved. There are very few exceptions in which the curve is not monotonic. This behavior can be attributed to the ATPG

Circuit	KGates	OPI/ATPG (time)	Pattern Count	Data Volume
C_210	210	0.49	1002	733330
C_260	260	0.37	8745	4357725
C_419	419	0.15	4613	3082955
C_845	845	0.47	1029	749927
C_2007	2007	1.11	2325	2766406
C_2225	2225	1.49	28038	9843423
C_2500	2500	1.07	7525	5457533
C_2508	2508	1.52	4729	7621035
Average	N/A	0.83	N/A	N/A

Table 3.1: Observation Point Selection time

heuristics that order the fault list producing noise when a few new faults (due to the observation points) are added to the fault list.

In Table 3.3, a comparison between the CR and DVR methods is conducted for the eight circuits in the experiment. In Table 3.3, the first column shows the number of OPs inserted relative to the circuit gate count, the next column *PC Red%* shows the average percentage pattern count reduction for the eight circuits in the experiment for both the CR and DVR methods. Last, under the column labeled *DV Red%*, the same information is shown for the average percentage data volume reduction. It can be seen from the experimental data that both methods perform similarly for the same number of OPs. The difference is minimal and can be attributed to the fault order.

In Table 3.4, a comparison between our method and MTPI [64] is done for the circuits whose netlists were available in Verilog format. This was necessary to run MTPI on them. While MTPI was devised for BIST, we used if for comparison in deterministic test due to its availability. In Table 3.4, after the circuit name we

# of OP Inserted	C_260	
	# Patterns	Data Volume
None	8745	4357725
1	8130	4156080
2	7454	3948161
3	6889	3768812
4	6101	3536561
5	5532	3364807
6	4774	3144804
7	4133	2967506
8	3727	2782390
9	3726	2602537
10	3681	2415063
11	3649	2253949
12	3661	2173013
1 / 40 KGates	4774	3144804
1 / 20 KGates	3660	2170960
1 / 10 KGates	3654	2135085
1 / 5 KGates	3666	2074099
1 / 2 KGates	3673	1929533
1 / 1 KGate	3675	1734588
1 / 500 Gates	3667	1534205

Table 3.2: Results for C_260

# of OPs /# Gates	PC Red %		DV Red %	
	CR	DVR	CR	DVR
1/ 10000	28.06	30.38	17.90	19.44
1/ 5000	32.22	32.42	21.18	22.53
1/ 1000	37.84	37.47	29.51	30.74
1/ 500	41.44	41.55	34.50	34.40

Table 3.3: CR vs. DVR

Circuit	PC Red %		DV Red %	
	DVR	MTPI	DVR	MTPI
C_260	58.50	5.22	62.29	5.60
C_419	68.01	53.00	54.31	13.39
C_2007	16.42	7.52	56.73	18.31
C_2225	13.45	6.69	32.58	2.91
C_2508	57.04	37.97	9.07	13.11
Average	42.68	22.08	43.00	10.66

Table 3.4: DVR vs. MTPI

show the pattern count reduction both for the DVR method and MTPI. In the last column we show the data volume reduction for both methods. It can be seen from the data that DVR clearly outperforms MTPI when applied to deterministic test.

In Table 3.5, results obtained using an industrial compression tool based on EDT [14] are shown for the circuits using the DVR method to compute OPs. In Table 3.5, after the circuit name, under the column *Original*, we show the original pattern count using EDT. Under the column *Red%* we show the percentage pattern count reduction when inserting one OP per 500 circuit gates after the tool was done

Circuit	Original	Red %	@ Same TC	Scan Chains
C_210	1122	47.27	47.27	5
C_260	9051	57.65	60.74	4
C_419	7509	51.44	67.16	19
C_845	1096	30.02	30.02	16
C_2007	2997	22.06	42.01	8
C_2225	27825	14.34	15.42	14
C_2500	6359	12.19	14.62	64
C_2508	5750	53.55	57.60	32
Average	N/A	36.06	41.85	N/A

Table 3.5: Results with EDT

generating patterns. Then, under the column *@ Same TC*, we show the percentage pattern count reduction at the same test coverage that was achieved without OPs. Last, under the column *Chains*, we show the number of scan chains that each circuit has. Since the number of scan chains of every circuit in the sample was small we only use one scan channel for each circuit. Due to the fact that we were not interested in benchmarking compression but to compare results with and without OPs we did not reconfigure the scan chains to obtain a better compression ratio. Since compression is being used, we do not report data volume which is proportional to pattern count. From the large reductions obtained it can be seen that the insertion of the OPs under compression greatly reduces test set length.

3.5 Experimental results on benchmark circuits

In this section results for the CR and DVR methods for ISCAS benchmark circuits are given for the sake of completeness. Both methods fail to provide with pattern count reduction for these circuits as it can be seen in Tables 3.6, 3.7, 3.8

and 3.9. It is important to notice the reasons so a discussion will be provided next.

Both methods analyze the resulting cubes of a ATPG run performed on the original circuit and based on that they compute observation locations. This is done under several assumptions:

1. There is enough cubes created by the ATPG so that the database created has enough statistical information.

2. A large enough number of the faults targeted by ATPG are hard to detect faults (in the sense that random fill has small chance to detect them).

3. Enough faults targeted in the original ATPG run are targeted again after observation point insertion by ATPG.

4. Several retargeted faults when targeted again use the same decision order in the *J-frontier* and in the *D-frontier*.

These assumptions can be summarized into two. First, the ATPG runs pre observation point insertion and post observation point insertion are similar. Second, pattern count is high. While pattern count depends on the CUT design, DFT techniques applied to it and the ATPG algorithm used to create the test set it can be observed a correlation between pattern count and circuit size (and number of faults too).

For ISCAS benchmark circuits this assumptions do not hold because they are small circuits. They have a very low gate count and number of faults. The test sets created by ATPG have a very low pattern count with only one case above two hundred patterns. Then the database does not have enough information to properly locate observation points. Nor there is many hard to detect faults targeted, nor the pre-insertion and post-insertion ATPG runs are similar.

For these reasons the effect of the extra logic inserted (as observation points) does not reduce the pattern count in the same percentages than in industrial

Circuit	Original	10K	5K	1K	500
c2670	71	-	74	69	75(-8)
c3540	144	-	149	139	130
c5315	96	96	92	87	85
c6288	36	35	34	38	39
c7552	125	128	119	113(-18)	116(-18)

Table 3.6: CR pattern count results for ISCAS85

circuits. In some cases pattern count increases due to the extra faults added to the fault list. This extra faults belong to the inserted logic. The effect that they have is resorting the fault list, changing the ATPG run.

It is also important to notice that as observation points were added a few redundant faults became testable. Since the original pattern count was very low, the extra patterns due to these newly detectable faults can impact severely on pattern count.

In Table 3.6 pattern count results are presented for the CR method for the largest five ISCAS85 benchmark circuits. In Table 3.6 after the circuit name, under column *Original*, we show the original pattern count without any observation points. Then, under columns *10K*, *5K*, *1K* and *500* we show pattern count results when one observation point is inserted every ten thousand, five thousand, one thousand and five hundred faults in the original fault list respectively. The quotient is rounded up to obtain an integer number of observation points. When the number of redundant faults was reduced we show this reduction in parenthesis in the same table cell after the pattern count result. The original number of redundant faults is not shown but in can be found in [65].

Tables 3.7, 3.8 and 3.9 are organized in the same manner as Table 3.6. Table 3.7 shows results for the DVR method on the same subset of ISCAS85 circuits

Circuit	Original	10K	5K	1K	500
c2670	71	-	73	75	66
c3540	144	-	149	134	134
c5315	96	96	92	90	82
c6288	36	35	37	36	37
c7552	125	128	119	109(-18)	116(-18)

Table 3.7: DVR pattern count results for ISCAS85

Circuit	Original	10K	5K	1K	500
s5378	132	133	133	133	125
s9234	192	184	162(-8)	142(-48)	136(-48)
s13207	291	283	274	265(-2)	245(-26)
s15850	159	155	149	136	131(-22)
s35932	45	45	45(-6)	49(-38)	48(-69)
s38417	136	137	142	115	112(-1)
s38584	186	189	180	170	166(-1)

Table 3.8: CR pattern count results for ISCAS89

than Table 3.6. Table 3.8 shows results for the CR mehtod on the largest seven ISCAS89 circuits. Table 3.9 shows results for the DVR method on the same subset of ISCAS89 circuits than Table 3.8.

3.6 Conclusions

In this paper we presented a new observation point insertion technique to enhance compaction and reduce data volume for scan based test. The method analyzes the cubes created via ATPG and based on them inserts the test points.

Circuit	Original	10K	5K	1K	500
s5378	132	140	133	129	122
s9234	192	184	162(-8)	142(-48)	137(-48)
s13207	291	294(-16)	283(-16)	265(-21)	246(-24)
s15850	159	163	153	139(-45)	141(-48)
s35932	45	45	45(-6)	42(-24)	47(-60)
s38417	136	139	137	120	110(-1)
s38584	186	179	182	169(-18)	157(-19)

Table 3.9: DVR pattern count results for ISCAS89

Because of this the method is ATPG heuristics dependent. Since the method run times do not exceed those of test pattern creation the technique is scalable to large industrial designs. Experimental results with industrial designs showed substantial reductions in pattern count and total specified positions before X-filling. Comparisons with other test point insertion methods demonstrate the advantages of the method. Experiments with industrial test compression tools show that under compression environments the technique is applicable and achieves large benefits in test duration and therefore test cost.

CHAPTER 4
A SCALABLE METHOD FOR THE GENERATION OF SMALL TEST SETS

This Chapter presents a scalable method to generate close to minimal size test pattern sets for stuck-at faults in scan based circuits. The method creates sets of potentially compatible faults based on necessary assignments. It guides the justification and propagation decisions to create patterns that will accommodate most targeted faults. The technique presented achieves close to minimal test pattern sets for ISCAS circuits. For industrial circuits it achieves much smaller test pattern sets than other methods in designs sensitive to decision order used in ATPG.

4.1 Introduction

For scan based circuits the test application time is proportional to the test set size and the length of the longest scan chain. Hence, it is important to reduce test set size by generating compact test sets in order to reduce test cost. In recent years, with the introduction of test compression techniques, test cost has been reduced. Test compression techniques introduce test generation time overhead. This overhead has been initially compensated by the utilization of faster Automatic Test Pattern Generator (ATPG) engines. However, faster ATPG engines may create abnormally large test sets for some circuits under test (CUTs). This weakness in the ATPG heuristics must be overcome to avoid abnormal pattern counts for some CUTs. At the same time test generation must be completed in a reasonable time to cope with large industrial designs.

The techniques to obtain a compact test set can be classified into static and dynamic compaction methods. Static compaction methods are applied to already generated test sets to further reduce their size by removing redundant

tests. Some of these methods do not alter the tests in the set [6][66]. Others, after relaxing the test vectors in the set, attempt to target faults detected by a vector so that they will be detected by other vectors in the set, rendering the initial vector redundant [50][67]. Dynamic compaction methods attempt different heuristics to accommodate detection of more faults by a test pattern while it is being created [68]. The basic principle of dynamic compaction is to create a test cube for a fault, called *primary* or *parent* fault, and use the resulting specified positions as constraints to target other faults, called the *secondary* or *child* faults. The best results in test set size are obtained by methods using both static and dynamic compaction techniques.

This work focuses on opportunistically achieving close to minimal test sets using dynamic compaction techniques. The objective is not to create minimal test sets but to get consistently close to minimal test counts with a fast algorithm that can be applied to large industrial designs. An ATPG engine based on the D-algorithm using new ways to guide decisions in order to accommodate detection of more faults by the same test vector is developed. The guidance is based on a preprocessing step that computes cliques of faults based on some of their necessary assignments. After the preprocessing step, the faults in a clique are targeted for test generation. Each time a decision is to be made by the ATPG, an attempt to avoid violating the necessary assignments of the remaining faults in the clique is made.

The rest of this chapter is organized in the following manner. Section 4.2 describes earlier works on compact test set generation related to the present work and others that achieve related results. Section 4.3 presents the dynamic compaction algorithm proposed. Experimental results are given in Section 4.4. Section 4.5 concludes the chapter.

4.2 Earlier Works

Terms related to test generation procedures and used later in the paper are defined below [6].

Definition 1: J-frontier is the set of all gates whose output value is known but is not implied by its input values.

Definition 2: D-frontier is the set of all gates whose output value is unknown and they have one or more error signals on their inputs.

In [69], the authors combine dynamic and static techniques to generate minimal or close to minimal test sets. *Dynamic fault ordering*, a technique to sort the fault list during test generation, based on the computation of *independent fault sets* (IFS) is used. IFS are sets of faults in which no two faults can be detected by the same test vector. In [69], after the generation of a test vector the largest IFS remaining is placed on top of the fault list and a new fault is selected as a target for test generation from the IFS. Another heuristic, *rotating backtrace*, based on the rotation of gate inputs selected to justify values in the J-frontier, is employed to facilitate detection of yet undetected faults. *Double detection*, which requires each fault to be detected twice before being dropped from the fault list, is used as a dynamic compaction technique to create test vectors that detect earlier detected faults [51]. This facilitates dropping of tests generated earlier by performing reverse order fault simulation. Extremely compact test sets are created in [51] at the expense of large computational times. This reduces the method's applicability to larger designs. In [52], a similar trade-off between test set generation time and test set size is proposed. Using similarly oriented techniques, smaller test set sizes are achieved at the expense of additional computing time. We will compare the results obtained with the proposed method to those obtained in [52] and [51].

In [70] a method, called SCOAP, to guide ATPG decisions is introduced. SCOAP is aimed at measuring the complexity of justifying a line value and

observing a gate. It does this by creating scalar estimates of how many circuit inputs are necessary to control and observe each gate. Its complexity is linear in the circuit gate count. SCOAP has the disadvantage that it fails in the presence of reconvergent fanout by either underestimating or overestimating the scalar measures. Also it is static in the sense that it does not take into account yet undetected faults for ATPG guidance.

The work in [71] uses necessary assignments (NA) for sensitizing selected paths in order to create compact test sets to detect transition faults through longest testable paths. A NA *clique* of faults, is a subset of faults such that necessary assignments of any pair of paths in the clique do not conflict. In [71], the collection of NAs for all paths in a clique are called clique assignments (CA). Next, tests to satisfy all necessary assignments of CA are derived. These tests sensitize all paths and detect the corresponding transition faults in a clique. The paths chosen are first verified to be sensitizable and the target transition faults are detected when the paths are sensitized. So, the problem of creating a test for these faults is reduced to justifying the necessary assignments for sensitizing the faults. In [71] it is pointed out that once a clique C is formed, it is very rare that the necessary assignments in the corresponding CA cannot be simultaneously justified.

In this work the focus is on stuck-at faults. We also form cliques of faults based on necessary assignments that do not conflict. However, not all necessary assignments to detect a fault can be used. We use a limited set of necessary assignments for the activation and propagation of faults in forming cliques of faults. The collection of the necessary assignments for faults in the clique CA are used to guide J-frontier and D-frontier decisions by the ATPG when generating a test for a fault. In this way, we attempt to avoid conflicts with CA. Thus, it is possible that the ATPG will violate some necessary assignments in CA in searching for a test. This will happen when the ATPG cannot create a test for a given fault in the

restricted search space compatible with CA.

4.3 The proposed method

The objective of both static and dynamic compaction is to produce smaller test sets. As discussed before in Section 4.2 some compaction procedures attempt to find optimal test sets with computationally intensive techniques.

This section describes two scalable methods we propose for generating test sets of near minimal size. Section 4.3.1 describes the necessary assignments of a fault and which ones are considered by our methods. Section 4.3.2 describes the *single detections* (SDA) algorithm. Section 4.3.3 provides an example of the SDA method and Section 4.3.4 describes the proposed *extra detections* (EDA) method.

4.3.1 Necessary assignments

For a fault f, necessary assignments (NA) are every line value necessary for the detection of the fault. This includes values to activate f and propagate its effect to a fanout stem or an output. The number of necessary assignments to detect a stuck-at fault can be increased, for example, by determining dominators [72] and using learning techniques [73][74].

We use necessary assignments obtained by using simple forward and backward implications on the fault-free circuit only. Specifically, we do not use dominator analysis and learning techniques. Thus, implications stop at the first fanout stem or output reached from the fault site and the backward implications stop when inputs of a gate are not uniquely implied by its output. We illustrate the necessary assignments we use by determining them for the fault f in Figure 4.1. The necessary assignments we use for fault f are shown in bold and are underlined. For fault f, many additional necessary assignments can be found using dominators and learning. These are shown in italics without underlines. It should be noted that not all

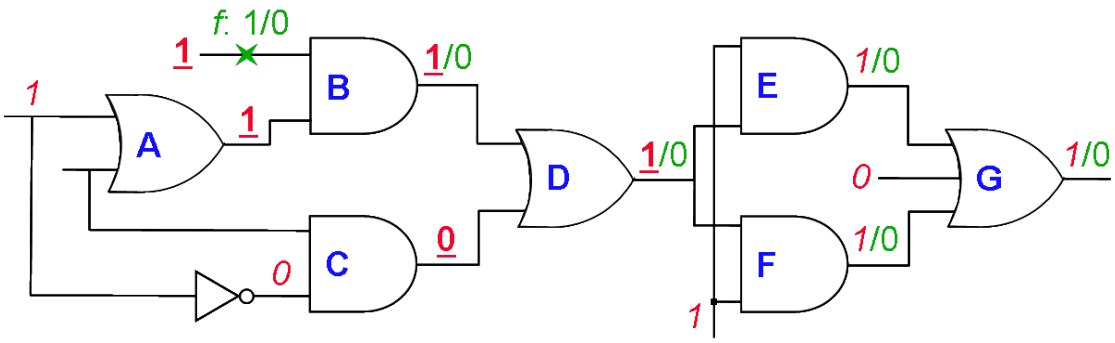

Figure 4.1: Necessary assignments for f

1. Compute NA-estimates ($NAest$).
2. Sort the fault list F by decreasing order of $NAest$.
3. Pick first untargeted fault as parent fault f_p and compute NA.
4. Add the NA of f_p to CA and f_p to C.
5. $\forall f \in F$:
   (a) If f cannot be activated: continue.
   (b) If f does not have an X-$path$: continue.
   (c) Compute NA for f with CA as constraints.
   (d) If NO conflict: add NA to CA and f to C.
6. Generate a test for f_p guided by CA.
7. If success: Target every $f \in C$ guided by CA.
8. Random fill and fault simulate using fault dropping the created test pattern.
9. If F not empty: goto 3.

Figure 4.2: A pseudocode of the SDA method

underlined assignments in bold are normally regarded as necessary assignments. For example the outputs of gates B and D. In our method we use these values as necessary assignments for fault F.

4.3.2 Single detections algorithm (SDA)

The key concept in the *single detections algorithm* (SDA) is the addition of a pre-processing step to test generation. This pre-processing step screens faults to determine which ones should be targeted for detection by a test and determines how ATPG decisions are made. It does this by computing NAs for the faults. If two faults have conflicting NAs they cannot be detected with the same pattern. If they have compatible NAs they will be targeted with the same pattern and the *J-frontier* and *D-frontier* decisions will be guided, using the computed NA, in an attempt to facilitate detecting the faults with a single test.

SDA works in the following way. It orders the faults placing the ones with larger number of NAs first. However, instead of ordering on actual number of NAs per fault we use an easily computed approximation of the NA count. We call this *NAest*. This is done to avoid determining the NA of all faults upfront since during test generation only the NA of faults that remain undetected are needed. The *NAest* computation time for the entire fault list is linear in the gate count of the circuit. The motivation for ordering the faults in decreasing order of *NAest* is that faults with larger *NAest* will tend to have more conflicts with other faults and restrict to a larger degree the number of additional faults that can be detected by a single test. After this preprocessing step test generation is started.

The first untargeted fault f_p from the ordered fault list is selected as a *parent* fault and added to a clique C. The NA of f_p are computed and added to the set of clique assignments CA. Next the NA of a yet undetected fault f is checked for compatibility with CA. If f can be activated and has an X-path under CA, NA of f with CA as constraints are computed. If there is a conflict CA is restored and f discarded. Else f is added to C and CA is updated with the computed NA of f.

After C is formed, every fault except f_p in C is sorted. The sorting criterion is the number of previous times that f was targeted as a secondary target fault. If f

was targeted more times it is placed higher in the list. If two faults were attempted the same number of times, sorting places first the fault whose *NAest* value is larger. In this way more restrictive faults are placed first in the list to be targeted.

After C is sorted f_p is targeted by the ATPG. Every time a J-frontier decision or a D-frontier decision has to be made by the ATPG we attempt to maximize the compatibility of the decision with CA. If a test cube is formed for f_p, every $f \in C$ will be targeted as secondary fault for detection by the same test in order of appearance in the sorted clique C.

Once all the faults in C are targeted the unspecified values in the final test cube are randomly filled, the resulting test vector is fault simulated and all the detected faults are dropped. This process is repeated until the fault list F has no more faults which were not targeted as parent faults. In Figure 4.2, we give a pseudocode of the ATPG flow.

4.3.3 An example of SDA method

Consider the circuit of Figure 4.3 with f_1, f_2 and f_3 being the only faults remaining in the fault list F. The gate whose output is labeled **m** is a multiplexer. Lets us assume that the faults are originally ordered in ascending order of their indices. Next we discuss the effect of applying SCOAP and random decision order as well as the SDA algorithm in generating tests for the faults.

If SCOAP based guidance is applied to a test for f_1 a D-algorithm based ATPG will face two decisions. One in the J-frontier when justifying line **f**=1 and the other in the D-frontier when propagating **g**=D through **g1** or **g2**. Since the circuit is symmetrical the choices could be **c**=1 for justifying **f**=1 and **g1** to propagate the error value on **g**. This will result in a test vector that cannot accommodate detection of any of the other two faults. This is because f_2 is blocked since **c** is set to 1 and f_3 is blocked because the select input **p** of the multiplexer will be 0 to

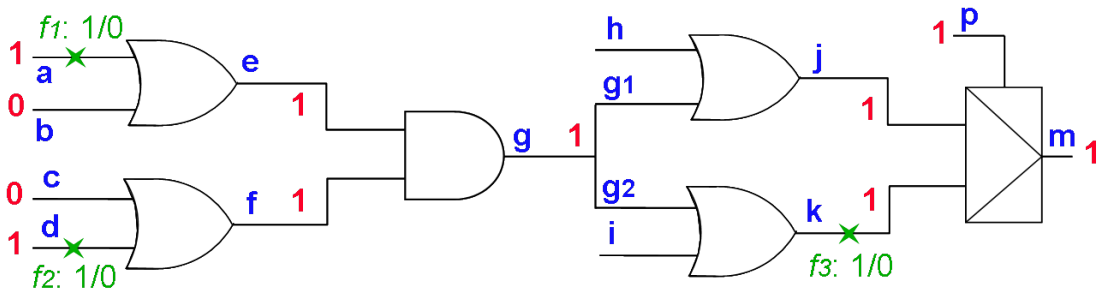

Figure 4.3: An example illustrating SDA method

propagate the error value on **g1**. This is illustrated in Figure 4.4. When targeting f_2, the same choice in the D-frontier will be faced and SCOAP based guidance will again lead to the same result. For these faults SCOAP based guidance will lead to three test patterns.

Instead if random decision order is used, the D-algorithm, when faced with the same decisions has a 50% chance to decide among two choices for each decision. Computing the probabilities for different sets of tests to detect the faults using random decisions, with probability 0.25 a single pattern to detect all faults will be obtained, with probability 0.625 two patterns to detect all faults will be created and with probability 0.125 three patterns will be created.

If the SDA algorithm is used, NAs for the faults in F will be computed. As there is no conflict in the necessary assignments a clique C including all the faults in F will be formed. The necessary assignments for all the faults in F are displayed in Figure 4.3. When the J-frontier decision on **f** is to be made the SDA algorithm will choose **d** to set **f**=1 to avoid conflict with **c**=0 which is part of the CA of the fault clique C. When the D-frontier decision in **g** is to be made, SDA algorithm will choose branch **g2** to avoid conflicting with the assignment **p**=1, which is also part of the CA. The pattern created in this way can detect faults f_2 and f_3. Thus the SDA algorithm will produce one test vector to detect all the faults.

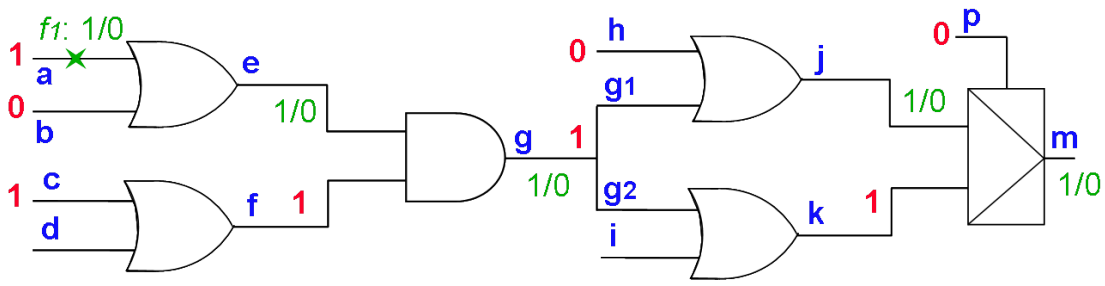

Figure 4.4: Test for f_1 using SCOAP

4.3.4 Extra detections algorithm (EDA)

In order to reduce the pattern count obtained using the SDA procedure given above we used the following observation in deriving a modified procedure called *extra detections algorithm* (EDA).

As test generation proceeds, the number of yet undetected faults decreases and the sizes of the cliques of compatible faults constructed from the undetected faults decreases. We can increase the sizes of cliques by adding faults that have been detected by tests generated earlier. This causes some faults to be detected several times without increasing the size of the test set compared to that obtained by using the SDA algorithm. Multiple detection of faults improves the quality of tests by increasing the probability of detection of unmodeled faults. This observation was the motivation behind the recent work called Embedded Multi-Detect ATPG [75]. Additionally, as observed in [51], the extra detections of earlier detected faults causes some of the tests generated earlier to become unnecessary and they can be dropped using static compaction techniques [51][66].

We modify the formation of cliques in the SDA algorithm to obtain the EDA algorithm. In the SDA algorithm the fault cliques are formed from yet undetected faults. In the EDA algorithm after considering the yet undetected faults to form the cliques we consider faults already detected in increasing order of the number of times they are detected. In order to increase the probability of dropping earlier

generated tests we use the following heuristics in processing the already detected faults. We only keep the faults that are detected less than ten times, dropping the faults at the tenth detection. For each fault we record the first pattern that detects it and the number of faults detected by the pattern for the first time. For each earlier generated pattern we record the number of faults uniquely detected. We use this number when considering faults that are detected exactly one time by earlier tests.

When we consider adding already detected faults to a clique we first consider faults detected only once in the following order. If the pattern p that detects a fault f uniquely detects fewer faults, then f is placed in the clique ahead of other faults. This heuristic increases the probability of dropping patterns that uniquely detected fewer faults since faults not uniquely detected by such patterns are already detected by other patterns.

While considering faults detected two or more times we use the following heuristics. Faults are considered in increasing order of the number of times they are detected. Let faults f and g be detected the same number of times and patterns p and q detect f and g for the first time respectively. We place f ahead of g if pattern p detected fewer faults for the first time than pattern q.

4.4 Experimental results

In this section, experimental results for ISCAS benchmark circuits and industrial designs are presented. All results for ISCAS circuits were obtained using a 3.6-GHz processor. Results for industrial circuits were obtained using a 2.8-GHz processor. The proposed procedures were implemented as add on to a commercial ATPG based on the D-algorithm.

CUT	Pattern Count					
	SDA	EDA	SC	LB	MT	CT
c432	37	37	32	27	27	29
c499	52	52	52	52	52	52
c880	23	21	20	13	16	21
c1355	85	85	84	84	84	84
c1908	115	111	107	106	106	106
c2670	56	49	47	44	44	45
c3540	104	103	99	80	84	91
c5315	51	52	50	37	37	44
c6288	26	22	***21**	6	12	14
c7552	92	88	83	65	73	80
Total	641	620	595	514	535	566

Table 4.1: Pattern count results for ISCAS85 benchmark circuits

CUT	Time (sec)				BCE		
	SDA	EDA	MT	CT	SDA	EDA	SC
c432	0.06	0.23	6.2	7	80.05	83.94	81.96
c499	0.07	0.64	17.4	5	92.92	92.98	92.98
c880	0.07	0.24	10.4	12	84.52	84.96	84.09
c1355	0.28	1.86	29.4	16	92.33	92.38	92.28
c1908	0.34	1.78	78.9	55	91.16	91.47	90.99
c2670	0.32	1.21	73.3	130	90.42	92.27	92.01
c3540	0.78	3.26	178.1	262	89.11	89.81	89.39
c5315	0.63	2.69	265.4	362	89.18	91.11	90.61
c6288	3.03	14.42	65.6	398	96.73	95.89	95.63
c7552	1.38	7.80	794.7	1311	94.28	95.88	95.57
Total	6.96	34.13	1519.4	2558	90.07	91.07	90.55

Table 4.2: Run time and BCE for ISCAS85 benchmark circuits

4.4.1 Results on ISCAS circuits

Tables 4.4.1 4.4.1, 4.4.1, 4.4.1, 4.4.1 and 4.4.1 show results for the ISCAS benchmark circuits set. In Tables 4.4.1, 4.4.1 and 4.4.1, after the circuit name the pattern counts for different test generation methods are shown. In Tables 4.4.1, after the circuit name, under column *Time (sec)* the run times for the different methods are shown. Next, under column *BCE* the bridge coverage estimates, computed as in [76], are shown for some test generation methods. In Tables 4.4.1 4.4.1, 4.4.1, 4.4.1, 4.4.1, the abbreviation *SDA* refers to the single detections method, *EDA* refers to the extra detections method, *SC* is for static test compaction of [66] used on tests obtained using method EDA, *LB* is the highest known lower bounds on the test set sizes from [52], *MT* is the method in [52] and *CT* is the method in [51]. Run times for *SC* are not shown since they were negligible. For the largest ISCAS circuit (s38584), it took 0.14 seconds to perform SC on the test set obtained using EDA. The results for the ISCAS89 benchmark circuits set is broken into two tables with totals in the last (Part b) of them to fit them into the page format.

From these tables we can see that the proposed algorithms approach and even sometimes match the lower bounds known for ISCAS circuits. Only in the three circuits marked (*) the method failed to approach the lower bound and produced a high pattern count. In one case (**), both the proposed methods produced the smallest test set known so far. The run times for different methods cannot be directly compared since [52] was run on a 200-MHz Pentium Pro PC, [51] was run on a SUN SPARC 2 and the proposed methods were run on a 3.6-GHz processor. Given the disparity of the capabilities of the computers used we can only focus on the run time trends relative to circuit sizes to discern to discern the scalability of different methods. To illustrate this, in Figure 4.5 we plot the run times of methods SDA, EDA, MT and CT normalized by dividing the run time by the number of

CUT	Pattern Count					
	SDA	EDA	SC	LB	MT	CT
s208	32	32	32	27	27	27
s298	24	24	24	23	23	24
s344	15	15	15	13	13	15
s349	15	15	15	13	13	14
s382	27	27	25	25	25	25
s386	65	64	64	63	63	63
s400	25	25	25	24	24	24
s420	70	70	***68**	43	43	43
s444	25	25	24	24	24	24
s510	56	56	55	54	54	54
s526	52	52	52	49	49	50
s526n	52	52	51	49	49	50
s641	24	24	23	21	21	22
s713	24	24	22	21	21	22
s820	97	97	96	93	93	94
s832	97	97	97	94	94	94
s838	146	146	***140**	75	75	75
s953	81	79	78	76	76	76

Table 4.3: Pattern count results for ISCAS89 benchmark circuits. Part a

CUT	Pattern Count					
	SDA	EDA	SC	LB	MT	CT
s1196	131	131	120	113	113	118
s1238	138	138	129	121	121	124
s1423	26	27	26	20	20	26
s1488	106	106	102	101	101	101
s1494	104	104	102	100	100	100
s5378	108	104	102	97	97	103
s9234	142	136	125	100	105	108
s13207	236	236	236	233	233	235
s15850	99	96	95	91	95	95
s35932	11	11	****10**	9	12	13
s38417	83	79	78	62	68	85
s38584	119	118	117	93	110	115
Total	2230	2210	2148	1927	1962	2019

Table 4.4: Pattern count results for ISCAS89 benchmark circuits. Part b

CUT	Time (sec)				BCE		
	SDA	EDA	MT	CT	SDA	EDA	SC
s208	0.03	0.05	0.4	0.8	81.40	83.94	83.94
s298	0.03	0.05	0.7	1.5	80.89	81.72	81.72
s344	0.01	0.06	0.7	1.5	79.78	80.50	80.50
s349	0.02	0.02	0.7	1.7	80.04	80.67	80.67
s382	0.03	0.12	0.8	1.7	83.94	85.77	84.49
s386	0.06	0.11	3.1	3.8	74.71	76.78	76.78
s400	0.04	0.09	0.8	1.8	82.48	84.36	84.36
s420	0.07	0.16	2.9	3.2	82.99	84.72	84.21
s444	0.04	0.13	0.9	2.3	82.62	84.46	83.72
s510	0.07	0.21	3.6	6.0	84.79	85.51	85.26
s526	0.04	0.19	3.0	4.9	83.64	85.96	85.96
s526n	0.08	0.17	3.3	4.9	83.78	85.86	85.66
s641	0.05	0.17	2.1	3.1	83.09	85.90	85.45
s713	0.07	0.21	2.8	4.6	84.70	89.01	87.86
s820	0.20	0.32	34.1	19	73.19	74.31	74.06
s832	0.14	0.33	80.1	20	73.12	74.09	74.09
s838	0.21	0.51	15.3	13	84.38	85.05	84.51
s953	0.13	0.44	30.3	25	85.52	86.13	85.93

Table 4.5: Run time and BCE for ISCAS89 benchmark circuits. Part a

CUT	Time (sec)				BCE		
	SDA	EDA	MT	CT	SDA	EDA	SC
s1196	0.25	0.68	43.6	48	84.41	86.53	85.61
s1238	0.24	0.70	127.4	102	84.35	86.12	85.39
s1423	0.16	0.69	205.3	32	87.05	89.58	89.24
s1488	0.23	0.68	75.1	40	81.46	82.52	81.64
s1494	0.25	0.72	80.4	43	80.94	82.16	81.79
s5378	0.73	2.96	131.5	216	92.19	94.86	94.75
s9234	1.49	6.18	3157.1	1085	87.41	90.40	89.74
s13207	2.90	10.04	1178.4	1096	88.84	93.44	93.44
s15850	3.13	12.86	9252.2	1375	91.50	94.17	94.11
s35932	2.32	5.26	11334.5	8388	75.14	75.73	74.56
s38417	7.21	26.39	28955.8	13210	92.89	94.77	94.69
s38584	8.05	29.76	38538.9	14446	92.31	95.17	95.12
Total	28.28	100.26	93265.8	40199.8	83.45	85.34	84.97

Table 4.6: Run time and BCE for ISCAS89 benchmark circuits. Part b

faults in the circuit. The normalized run times are given on a logarithmic scale. It can be seen that the run time per fault of SDA and EDA procedures are essentially constant where as the run times for MT and CT methods increase dramatically as the circuit size grows.

Bridge coverage estimate was proposed in [76] as a measure of detection of unmodeled defects. From the last three columns of Tables 4.4.1, 4.4.1 and 4.4.1, we note that the BCE for tests generated using EDA is higher than that of the tests generated using SDA even though the test set sizes of EDA are smaller. The BCE of test sets obtained after reducing the test sets of EDA by using static compaction are also higher. Thus we conclude that EDA produces smaller but higher quality test sets. Of course EDA requires longer run times compared to SDA.

4.4.2 Results on industrial designs

The results on four industrial circuits are given in Table 4.7. After the circuit name we give the circuit gate count in thousands of gates followed by pattern counts by various test generation methods. Here *Rand* and *SCOAP* stand for test generation procedures using random decision order and SCOAP based decision order, respectively. The results given under *Rand* and *SCOAP* are obtained after performing the static compaction technique [66] on the test sets. Method SDASC represents the use of SD followed by static compaction and the other methods are the same as given earlier in Table 4.4.1. Next three columns give run times relative to the run time of the *Rand* procedure. In the last two columns we give the bridge coverage estimate for SC and the best BCE of *Rand* and *SCOAP* based test generation methods.

From Table 4.7 we conclude that the proposed test generation methods produce the smaller test sets for all circuits. It can be noted that for some circuits the random decision order based method gives much smaller test sets than the

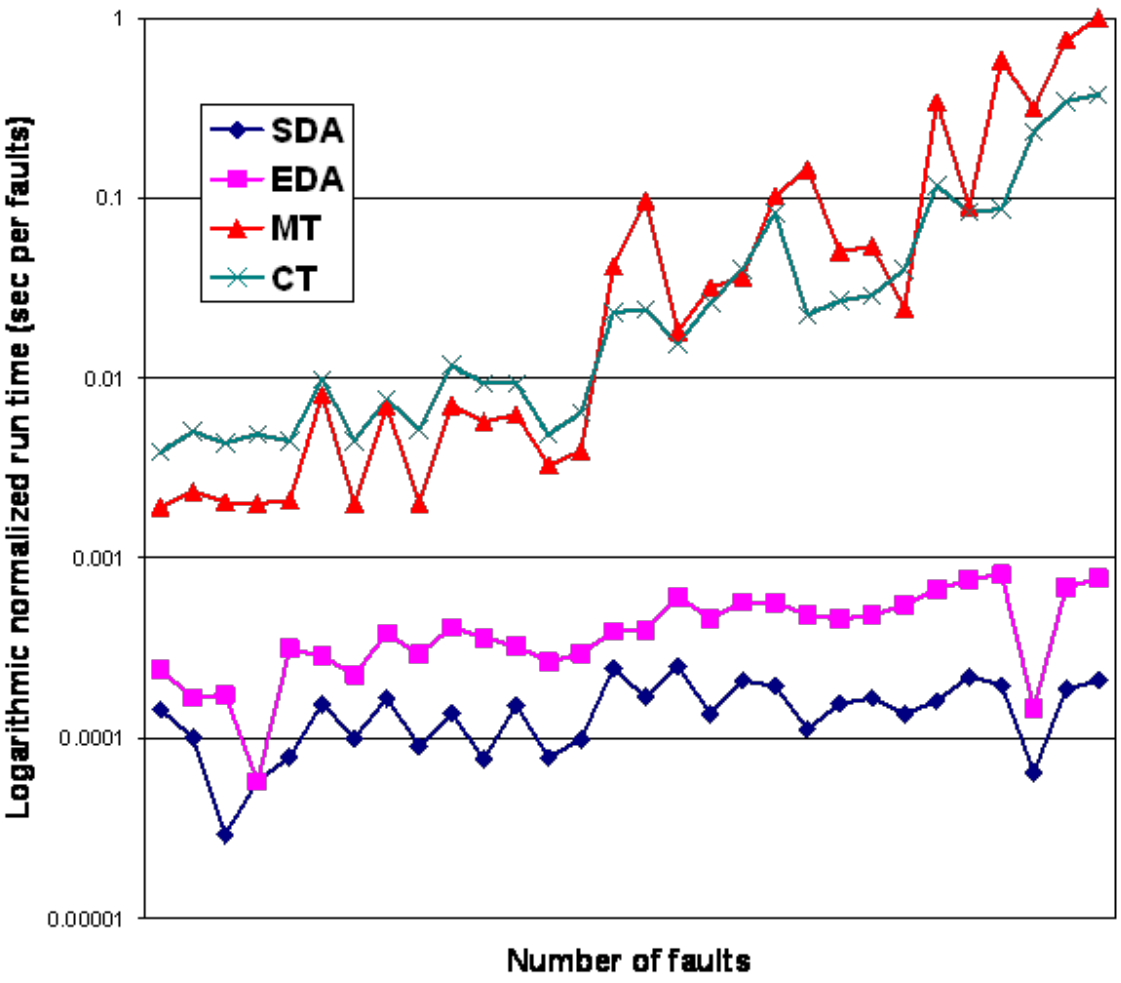

Figure 4.5: Run times for the various methods

CUT	KG	Pattern Count					
		Rand	Scoap	SDA	SDASC	EDA	SC
c-210	210	899	1192	659	653	634	615
c-260	260	8489	3632	3641	3639	3631	3301
c-419	419	1228	1256	1247	1243	1145	1077
c-845	845	905	5565	515	515	515	503

Table 4.7: Pattern count for industrial designs

CUT	Time (norm. to Rand)			BCE	
	Scoap	SDA	EDA	SC	Best R-S
c-210	1.86	1.72	3.89	95.45	91.47
c-260	0.59	1.37	2.91	87.93	81.31
c-419	2.29	0.81	2.51	97.95	92.95
c-845	5.37	1.15	2.53	93.08	99.30

Table 4.8: Run time and BCE for industrial designs

SCOAP based decision order but for other circuits it produces larger test sets. However the proposed methods consistently give smaller test sets for all circuits. The BCE of the test sets generated using EDA followed by static compaction are typically higher than the BCE of the *Rand* and *SCOAP* methods. Only for circuit c-845 the BCE of SC is lower than the best BCE of *Rand* and *SCOAP* due to the fact that their test set sizes are several times larger than the one for *SC*.

4.5 Conclusions

A scalable dynamic compaction technique that relies on preprocessing to determine guidance for the ATPG decisions was proposed. The proposed method

generates minimal or close to minimal test sets, except in three cases, for ISCAS benchmark circuits. For industrial designs it outperforms the best results of Random and SCOAP based decision guidance by always producing similar or better test set sizes.

CHAPTER 5
CONCLUSIONS

As transistor count in digital circuits grow so does gate count in them. Run time for software tools that create test sets for digital circuits grows with gate count. The complexity of manufacturing test increases with transistor count. Another trend is the increase in computers capabilities which alleviates the problem. Given this situation and the fact that these problems are NP-hard, as stated in the introductory discussion, every solution must be based on heuristics and therefore its scope should be temporal and not definitive.

This dissertation focused on two problems: pattern count increase and low power test.

For low power test, a new heuristic called *preferred fill* was proposed. The method is based on a fast pre-processing step that computes the most likely values to be held by a digital circuit and replaces random fill with these precomputed values. In this way, *preferred fill* reduces capture power by statistically matching the inputs of digital circuits to their responses to the inputs. *Preferred fill* can be used during test generation in replacement of random fill, or it can be used in a post-processing step for a pre-existing test set.

The properties of this solution are given next. It achieves substantial reduction in capture power. It is scalable to industrial designs because the pre-processing step used to compute the preferred values is linear in the gate count of the circuit. *Preferred fill* requires no additional hardware overhead as it is a software solution to low power test. It can also be combined with other existing techniques. When combined with adjacent fill it reduces both peak capture and average shift power dissipation.

The method trade off, when used during test generation, is the observed increase in test set size versus the reduction in power dissipation with no increase in computational effort. When used as a post-processing step on an already created test set the trade off is in computational effort against the reduction in power dissipation. The increase in computational effort in this case is due to the necessary relaxation step and not to the computation of preferred values.

For the pattern count increase problem two solutions were proposed: observation point insertion and a guidance algorithm for ATPG.

The observation point insertion approach aims at reducing test set size by making compatible sets of previously incompatible faults. To achieve this objective the cubes created during the ATPG run are analyzed and this information is stored in a database. Then the database is analyzed and the best observation point is selected based on a heuristic cost function. Then the database is updated and the process iterated until the desired number of observation points is achieved.

Given the nature of the process described above, the observation points computed in this way are dependant on the ATPG heuristics. How the ATPG justifies the J-frontier, propagates the D-frontier and selects the next target fault will affect the database and therefore the observation locations computed.

Given the statistical nature of the method, it is necessary that the circuit is large enough and the ATPG produces several cubes when it targets the circuit faults. Otherwise, the database will not have enough information to produce effective observation points.

The method is usable for stuck-at faults achieving great reductions in pattern count and therefore in test application time. This also holds true under compression environments, as experiments using an EDT-based tool confirmed. Inserting observation points for transition delay faults is risky. The same idea, on which the technique is based, can be applied to gross delay faults because they will be

detected anyway. However, transiton delay test sets are currently used in industry to detect small delay faults. In this case, the insertion of an observation point shortens the propagation path of the fault effect, and therefore the accumulated delay may be less than the clock period rendering the fault undetected. For this reason, no research was done to extend the method to transition faults.

For this method, the trade off is between pattern count reduction and its benefit on test application time and circuit area. As observation points are added, new scan cells are needed for them. This effect is minimal because the method achieves substantial reductions in pattern count with only one observation point per thousand gates. Extra computational effort is also needed to analyze the cubes and the database, modify the netlist and re-run the ATPG. But since these operations are in the same run time order of the initial ATPG run, the method can be performed on industrial designs.

The guidance algorithm for ATPG is based on a pre-process step that computes necessary assignments for faults and forms a set of potentially compatible faults. Then, based on the set of necessary assignments it guides the J-frontier and D-frontier decisions for each fault to avoid violating other faults necessary assignments, when possible. Otherwise it attempts to create a test for the fault with a minimum number of conflicts with the set necessary assignments. It also sorts the fault list to target it in a better manner based on heuristics. These heuristics depend on an estimation of how many necessary assignments the fault has and therefore an estimation of how incompatible the fault is with other faults.

The method was researched for stuck-at faults, but it can be extended to other fault models using the same principle. Since transition and propagation path delay faults have more necessary assignments than stuck-at faults, the task is harder for the older model (stuck-at). Stuck-at faults have D-frontier decisions to guide, while propagation path delay faults do not. It was noticed in the experiments

performed (but not previously mentioned in this thesis) that the more necessary assignments that we computed, wheter it was using learning or dominators, the better the method performed.

The necessary assignments clique formation method can be applied to any circuit size achieving close to minimal pattern count in ISCAS circuits. Its trade off is the extra computational effort needed for the pre-processing step against the pattern count reduction that it achieves. Since the pre-processing step is based on simple ATPG to compute trivial necessary assignments it is less computationally expensive than the cube creation step making the method scalable.

5.1 Future Research

The three different techniques proposed in this dissertation share the objective of contributing to dilucidate current manufacturing test problems. But their continuation must follow different paths that will be described next.

For the low power test technique, *preferred fill*, there is a need to compatibilize it with compression environments. Presently, almost every digital circuit manufacturer uses compression to reduce data volume and test application time. *Preferred fill* principles are not incompatible with EDT or broadcast scan but more research is needed to integrate those techniques. If *preferred fill* is integrated into the flow of a compression technique, a viable solution for low power test for industrial designs will be created.

The test point insertion technique is incomplete in the sense that only a method for observation point computation is presented. A technique for the localization of control points that is both scalable and compatible with the observation points needs to be developed. Moreover a general method that can decide when to insert an observation point and when to insert a control point is necessary. The control-observe point methods must have synergy *i.e.* the next test point to insert must be in the optimal location given the previous test points

inserted. In this way, they will break the inter-fault dependencies better, producing smaller test sets.

The ATPG guidance algorithm needs sensitivity analysis to be performed on it. The different parts of the method work well together but their individual impact both in pattern count and run time is unknown. Also, the method is scalable due to its practically linear growth in run time with circuit size but its current implementation is a phase that can be described as research code *i.e.* it was developed to prove a point, not to be a tool. If the method is to be used in a commercial tool, a proper analysis on its effects on run time, when compared to other guidance methods, is needed. For this, development on the technique is needed, which is beyond the scope of the present work.

For the ATPG guidance algorithm, more research is necessary to adapt it to all the currently used fault models, such as transition, path propagation delay, four-way bridging faults, etc. This will prevent the existence of different ATPG methods for different fault models, which is sub-optimal for commercial tools.

REFERENCES

[1] P. Whol, J. A. Waicukauski and S. Ramnath, "Fully X-tolerant combinational scan compression", Proc. of ITC 2007, pp. 1–10, October 2007.

[2] D. Gizopoulos, Advances in Electronic Testing: Challenges and Methodologies, Springer, New York, NY, 2006.

[3] J. Segura and C. Hawkins, "Tutorial 2: Understanding Failure Mechanisms and Test Methods in Nanometer Technologies", Proc. of ATS, October 2007.

[4] J. P. Shen, W. Maly and F. J. Ferguson, "Inductive Fault Analysis of MOS Integrated Circuits", IEEE Design and Test of Computers, Vol. 2, Issue 6, pp. 13–36, 1985.

[5] N. Nicolici, "Power minimisation techniques for testing low power VLSI circuits", Ph.D. Dissertation, University of Southampton, October 2000.

[6] M. Abramovici, M. Breuer, and A. Friedman, Digital Systems Testing and Testable Design, IEEE Press, Piscataway, NJ, 1994.

[7] E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability", Proc. of DAC, pp. 462–468, 1977.

[8] R. D. Eldred, "Test Routines Based on Symbolic Logical Statements", Journal of the ACM, Vol. 6, pp. 33–36, 1959.

[9] J. A. Waicukauski, E. Lindbloom, B. K. Rosen and V. S. Iyengar, "Transition Fault Simulation", IEEE Design and Test of Computers, Vol. 4, Issue 2, pp. 32–38, 1987.

[10] J. Savir and S. Patil, "Scan-based transition test", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 12, Issue 8, pp. 1232–1241, 1993.

[11] J. P. Roth, W. G. Bouricius and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits", IEEE Transactions on Electronic Computers, Vol. EC–16, pp. 547–580, October 1967.

[12] V. D. Agrawal, C. R. Kime and K. K. Saluja, "A Tutorial on Buil-In Self Test, Part 1: Principles", IEEE Design and Test of Computers, Vol. 10, Issue 1, pp. 73–82, March 1993.

[13] V. D. Agrawal, C. R. Kime and K. K. Saluja, "A Tutorial on Buil-In Self Test, Part 2: Applications", IEEE Design and Test of Computers, Vol. 10, Issue 2, pp. 69–77, April 1993.

[14] J. Rajski, J. Tyszer, M. Kassab and N. Mukherjee, "Embedded Deterministic Test", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 23, Issue 5, pp. 776–792, May 2004.

[15] J. Rearick, "Too Much Delay Fault Coverage Is a Bad Thing", Proc. of ITC, pp. 624–633, March 2001.

[16] J. Saxena, K. M. Butler, V. B. Jayaram, S. Kundu, N. V. Arvind, P. Sreeprakash and M. Hachinger, "A Case Study of IR-drop in Structured At-Speed Testing", Proc. of ITC, pp. 1098–1104, September 2003.

[17] P. Girard, "Low Power Testing of VLSI Circuits: Problems and Solutions", Proc. of ISQED, pp. 173–179, March 2000.

[18] S. Gerstendorfer and H. J. Wunderlich, "Minimized Power Consumption for Scan-Based BIST", Proc. of ITC, pp. 77–84, September 1999.

[19] L. Whetsel, "Adapting Scan Architectures for Low Power Operation", Proc. of ITC, pp. 863–872, October 1999.

[20] S. Bhunia, H. Mahmoodi, D. Ghosh, S. Mukhopadhyaay and K. Roy, "Low-Power Scan Design Using First-Level Supply Gating", IEEE Transactions on VLSI Systems, Vol. 13, Issue 3, pp. 384–395, March 2005.

[21] R. Sankaralingam and N. A. Touba, "Inserting Test Points to Control Peak Power During Scan Testing", Proc. of Intl. Symp. on DFT, pp. 138–146, November 2002.

[22] S. Sharifi, J. Jaffari, M. Hosseinabady, A. Afsali-Kusha and Z. Navabi, "Simultaneous Reduction of Dynamic and Static Power in Scan Structures", Proc. of DATE, pp. 846–851, March 2005.

[23] P. M. Rosinger, B. M. Al-Hashimi and N. Nicolici, "Scan Architecture with Mutually Exclusive Scan Segment Activation for Shift and Capture Power Reduction", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 23, Issue 7, pp. 1142–1153, July 2004.

[24] K.-J. Lee, S.-J. Hsu and C.-M. Ho, "Test Power Reduction with Multiple Capture Orders", Proc. of ATS, pp. 26–31, November 2004.

[25] Z. Zhang, S. M. Reddy, I. Pomeranz, J. Rajski and B. M. Al-Hashimi, "Enhancing Delay Fault Coverage through Low Power Segmented Scan", Proc. of ETS, pp. 21–26, May 2006.

[26] V. Dabholkar, S. Chakravarty, I. Pomeranz and S. M. Reddy, "Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 17, Issue 12, pp. 1325–1333, December 1998.

[27] M. Cho and D. Z. Pan, "PEAKASO: Peak-Temperature Aware Scan-Vector Optimization", Proc. of VTS, pp. 52–57, April 2006.

[28] Y. Bonhomme, P. Girard, L. Guiller, C. Landrault, S. Pravossoudovitch and A. Virazel, "Design of routing-constrained low power scan chains", Proc. of DATE, pp. 62–67, February 2004.

[29] S. Wang and S. Gupta, "LT-RTPG: A New Test-Per-Scan BIST TPG for Low Heat Dissipation", Proc. of ITC, pp. 85–94, September 1999.

[30] R. Sankaralingam, B. Pouya and N. A. Touba, "Reducing Power Dissipation During Test Using Scan Shain Disable", Proc. of VTS, pp. 319–324, May 2001.

[31] N. Z. Basturkmen, S. M. Reddy and I. Pomeranz, "A Low Power Pseudo-Random BIST Technique", Proc. of ICCD, pp. 468–473, September 2002.

[32] X. Liu and M. S. Hsiao, "Constrained ATPG for Broadside Transition Testing", Proc. of Intl. Symp. on DFT, pp. 175–182, November 2003.

[33] I. Pomeranz, "On the Generation of Scan-Based Test Sets with Reachable States for Testing under Functional Operation Conditions", Proc. of DAC, pp. 928–933, June 2004.

[34] Y.-C. Lin, F. Lu, K. Yang and K.-T. Cheng, "Constraint Extraction for Pseudo-Functional Scan-Based Delay Testing", Proc. of ASP-DAC, pp. 166–171, January 2005.

[35] I. Pomeranz and S. M. Reddy, "Generation of Functional Broadside Tests for Transition Faults", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 25, Issue 10, pp. 2207–2218, October 2006.

[36] Z. Zhang, S. M. Reddy and I. Pomeranz, "On Generating Pseudo-Functional Delay Fault Tests for Scan Design", Proc. of Intl. Symp. on DFT, pp. 398–405, October 2005.

[37] X. Wen, S. Kajihara, K. Miyase, T. Suzuki, K. K. Saluja, L. T. Wang, K. S. Andel-Hafez and K. Kinoshita, "A New ATPG Method for Efficient Capture Power Reduction During Scan Testing", Proc. of VTS, pp. 58–63, April 2006.

[38] O. Sinanoglu and A. Orailoglu, "Scan Power Minimization through Stimulus and Response Transformations", Proc. of DAC, pp. 404–409, February 2004.

[39] K. M. Butler, J. Saxena, A. Jain, T. Fryars, J. Lewis and G. Hetherington, "Minimizing power consumption in scan testing: pattern generation and DFT techniques", Proc. of ITC, pp. 355–364, October 2004.

[40] A. Chandra and K. Chakrabarty, "Combining Low-Power Scan Testing and Test Data Compression for System-on-a-Chip", Proc. of DAC, pp. 166–169, June 2001.

[41] S. Kahijara, K. Ishida and K. Miyase, "Test Vector Modification for Power Reduction During Scan Testing", Proc. of VTS, pp. 160–165, April 2002.

[42] X. Wen, Y. Yamashita, S. Kajihara, L. T. Wang, K. K. Saluja and K. Kinoshita, "On Low-Capture-Power Test Generation for Scan Testing", Proc. of VTS, pp. 265–270, May 2005.

[43] W. Li, S. M. Reddy and I. Pomeranz, "On Reducing Peak Current and Power During Test", Proc. of ISVLSI, pp. 156–161, May 2005.

[44] X. Wen, Y. Yamashita, S. Morishima, S. Kajihara, L. T. Wang, K. K. Saluja and K. Kinoshita, "Low-Capture-Power Test Generation for Scan-Based At-Speed Testing", Proc. of ITC, pp. 1019–1028, November 2005.

[45] S. Remersaro, X. Lin, Z. Zhang, S. M. Reddy, I. Pomeranz and J. Rajski, "Preferred Fill: A Scalable Method to Reduce Capture Power for Scan Based Designs", Proc. of ITC, pp. 1–10, October 2005.

[46] J. Savir and S. Patil, "Broad-Side Delay Test", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 13, Issue 8, pp. 1057–1064, August 1994.

[47] J. Savir, "Skewed-Load Transition Test: Part I, Calculus", Proc. of ITC, pp. 705–713, September 1992.

[48] X. Wen, Private Communication, November 2005.

[49] A. El-Maleh and A. Al-Suwaiyan, "An Efficient Test Relaxation Technique for Combinational Circuits & Full-Scan Sequential Circuits", Proc. of VTS, pp. 53–59, April 2002.

[50] S. Kajihara and K. Miyase, "On Identifying Don't Care Inputs of Test Patterns for Combinational Circuits", Proc. ICCAD 2001, pages 364–369, November 2001.

[51] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. M. Reddy, "Cost-Effective Generation of Minimal Test Sets for Stuck-at Faults in Combinational Logic Circuits", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 14, Issue 12, pp. 1496–1504, December 1995.

[52] I. Hamzaoglu and J. H. Patel, "Test Set Compaction Algorithms for Combinational Circuits", IEEE Transactions on CAD of Integrated Circuits and Systems, Vol. 19, Issue 8, pp. 957–963, August 2000.

[53] A. J. Briers and K. A. E. Totton, "Random Pattern Testability by Fast Fault Simulation", Proc. of ITC, pp. 274–281, September 1986.

[54] V. S. Iyengar and D. Brand, "Synthesis of Pseudo-Random Pattern Testable Designs", Proc. of ITC, pp. 501–508, August 1989.

[55] K.-T. Cheng and C.-J. Lin, "Timing-Driven Test Point Insertion for Full-Scan and Partial-Scan BIST", Proc. of ITC, pp. 506–514, October 1986.

[56] Y. Savaria, M. Yousef, B. Kaminska and M. Koudil, "Automatic Test Point Insertion for Pseudo-Random Testing", Proc. of ISCAS, pp. 1960–1963, May 1991.

[57] B. H. Seiss, P. M. Trouborst and M. H. Schulz, "Test Point Insertion for Scan-Based BIST", Proc. of ETC, pp. 253–262, 1991.

[58] K.-J. Lee, J.-J. Chen and C.-H. Huang, "Using a Single Input to Support Multiple Scan Chains", Proc. of ICCAD, pp. 74–78, 1998.

[59] M. J. Geuzebroek, J. T. van der Linden and A. J. van de Goor, "Test Point Insertion for Compact Test Sets", Proc. of ITC, pp. 292–301, October 2000.

[60] M. J. Geuzebroek, J. T. van der Linden and A. J. van de Goor, "Test Point Insertion that Facilitates ATPG in Reducing Test Time and Data Volume", Proc. of ITC, pp. 138–147, October 2002.

[61] I. Pomeranz and S. M. Reddy, "Test-Point Insertion to Enhance Test Compaction for Scan Designs", Proc. of DSN, pp. 375–381, June 2000.

[62] M. Yoshimura, T. Hosokawa and M. Otha, "A Test Point Insertion Method to Reduce the Number of Test Patterns", Proc. of ATS, pp. 298–304, November 2002.

[63] M. Yoshimura, T. Hosokawa and M. Otha, "Design for Testability Strategies Using Full/Partial Scan Designs and Test Point Insertions to Reduce Test Application Times", Proc. of ASP-DAC, pp. 485–491, January 2001.

[64] N. Tamarapalli and J. Rajski, "Constructive Multi-Phase Test Point Insertion for Scan-Based BIST", Proc. of ITC, pp. 649–658, October 1996.

[65] J. A. Waicukauski, P. A. Shupe, D. J. Giramma and A. Matin, "ATPG for Ultra-Large Structured Designs", Proc. of ITC, pp. 44–51, September 1990.

[66] X. Lin, J. Rajski, I. Pomeranz, and S. M. Reddy. "On Static Test Compaction and Test Pattern Ordering for Scan Designs". *Proc. IEEE International Test Conference*, pages 1088–1097, October 2001.

[67] L. N. Reddy, I. Pomeranz, and S. M. Reddy. "ROTCO: A Reverse Order Test COmpaction Technique". *Proc. 1992 Euro-ASIC Conf.*, pages 189–194, June 1992.

[68] P. Goel and B. Rosales. "Test generation and dynamic compaction of tests". *Dig. 1979 Test Conf.*, pages 189–192, October 1979.

[69] I. Pomeranz, L. N. Reddy, and S. M. Reddy. "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits". *IEEE Trans. Computer-Aided Design*, 12(7):1040–1049, July 1993.

[70] L. H. Goldstein and E. L. Thigpen. "SCOAP: Sandia Controllability/Observability Analysis Program". *Design Automation, Conference on*, pages 190–196, June 1980.

[71] Z. Wang and D. Walker. "Dynamic Compaction for High Quality Delay Test". *IEEE VLSI Test Symp.*, pages 243–248, April 2008.

[72] T. Kirkland and M. R. Mercer. "A Topological Search Algorithm for ATPG". *Design Automation, 24th Conference on*, pages 502–508, June 1987.

[73] W. Kunz and D. K. Pradhan. "Recursive Learning: a New Implication Technique for Efficient Solutions to CAD Problems-Test, Verification, and Optimization". *IEEE Trans. Computer-Aided Design*, 13(9):1143–1158, September 1994.

[74] M. H. Schulz, E. Trischler, and T. M. Sarfert. "SOCRATES: A Highly Efficient Automatic Test Pattern Generation System". *IEEE Trans. Computer-Aided Design*, 7(1):126–137, January 1988.

[75] J. Geuzebroek, E. J. Marinissen, A. Majhi, A. Glowatz, and F. Hapke. "Embedded Multi-Detect ATPG and Its Effect on the Detection of Unmodeled Defects". *Proc. IEEE International Test Conference*, pages 1–10, October 2007.

[76] B. Benware, C. Schuermyer, N. Tamarapalli, K.-H. Tsai, S. Ranganathan, R. Madge, J. Rajski, and P. Krishnamurthy. "Impact of Multiple-Detect Test Patterns on Product Quality". *Proc. IEEE International Test Conference*, 1:1031–1040, September 2003.