Fall 2014

Generation of compact test sets and a design for the generation of tests with low switching activity

Amit Kumar
University of Iowa

Recommended Citation

Kumar, Amit. "Generation of compact test sets and a design for the generation of tests with low switching activity." PhD (Doctor of Philosophy) thesis, University of Iowa, 2014.
https://doi.org/10.17077/etd.93exb2ob

GENERATION OF COMPACT TEST SETS AND A DESIGN FOR THE GENERATION OF TESTS WITH LOW SWITCHING ACTIVITY

by

Amit Kumar

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 2014

Thesis Supervisors: Professor Sudhakar M. Reddy
Dr. Janusz Rajski

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

Amit Kumar

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 2014 graduation.

Thesis Committee: ______________________________
Sudhakar M. Reddy, Thesis Supervisor

____
Janusz Rajski, Thesis Supervisor

____
Jon G. Kuhl

____
David R. Andersen

____
Mona K. Garvin

____
Hantao Zhang

To My Parents

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisors Professor Sudhakar M. Reddy and Dr. Janusz Rajski for all their support and guidance throughout the years of my graduate work.

I am also thankful to my thesis committee members for their encouragement and support.

Finally, I would like to thank my friends and family for their love and support.

ABSTRACT

Test generation procedures for large VLSI designs are required to achieve close to 100% fault coverage using a small number of tests. They also must accommodate on-chip test compression circuits which are widely used in modern designs. To obtain test sets with small sizes one could use extra hardware such as test points or use software techniques. An important aspects impacting test generation is the number of specified positions, which facilitate the encoding of test cubes when using test compression logic. Fortuitous detection or generation of tests such that they facilitate detection of yet not targeted faults, is also an important goal for test generation procedures.

At first, we consider the generation of compact test sets for designs using on-chip test compression logic. We introduce two new measures to guide automatic test generation procedures (ATPGs) to balance between these two contradictory requirements of fortuitous detection and number of specifications. One of the new measures is meant to facilitate detection of yet undetected faults, and the value of the measures is periodically updated. The second measure reduces the number of specified positions, which is crucial when using high compression. Additionally, we introduce a way to randomly choose between the two measures.

We also propose an ATPG methodology tailored for BIST ready designs with X-bounding logic and test points. X-bounding and test points used to have a significant impact on test data compression by reducing the number of specified positions.

We propose a new ATPG guidance mechanism that balances between reduced specifications in BIST ready designs, and also facilitates detection of undetected faults. We also found that compact test generation for BIST ready designs is influenced by the order in which faults are targeted, and we proposed a new fault ordering technique based on fault location in a FFR. Transition faults are difficult to test and often result in longer test lengths, we propose a new fault ordering technique based on test enumeration, this ordering technique and a new guidance approach was also proposed for transition faults. Test set sizes were reduced significantly for both stuck-at and transition fault models.

In addition to reducing data volume, test time, and test pin counts, the test compression schemes have been used successfully to limit test power dissipation. Indisputably, toggling of scan cells in scan chains that are universally used to facilitate testing of industrial designs can consume much more power than a circuit is rated for. Balancing test set sizes against the power consumption in a given design is therefore a challenge. We propose a new Design for Test (DFT) scheme that deploys an on-chip power-aware test data decompressor, the corresponding test cube encoding method, and a compression-constrained ATPG that allows loading scan chains with patterns having low transition counts, while encoding a significant number of specified bits produced by ATPG in a compression-friendly manner. Moreover, the new scheme avoids periods of elevated toggling in scan chains and reduces scan unload switching activity due to unique test stimuli produced by the new technique, leading to a significantly reduced power envelope for the entire circuit under test.

PUBLIC ABSTRACT

Test set size is known to increase at a faster rate than device size. Test cost for a given circuit is directly proportional to the number of independent test data inputs called test vectors, hence reducing the number of test vectors reduces test cost. Automatic Test Pattern Generation (ATPG), a tool, used to generate test patterns is used to generate tests. Decisions made while generating tests play an important role and should be made in a manner to facilitate detection of undetected faults. Since, modern design use compression logic a mechanism in which only a given number of deterministically specified positions can be encoded, reduction in the number of specified positions is a necessity.

We first propose new heuristics for making ATPG decisions based on undetected faults. The heuristics presented create appropriate balance between two contradictory requirements of allowing a test to facilitate detection of undetected faults and the number of specified positions in tests. Further, we propose an ATPG methodology to reduce test set sizes when additional hardware is used to improve test pattern counts. Next we propose a new fault ordering scheme based on ease of detection of a fault which also aids in reducing test pattern counts.

When a chip is tested, usually power dissipation is more than functional specifications leading to over-stressing of tested chips, potentially causing damage to chips during test. We also propose, what we call Isometric Compression Scheme to simultaneously reduce test power and test set size.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

In deep submicron era device complexity is increasing at a phenomenal rate. On the same sized die, roughly double the number of transistors are printed every eighteen months. This increasing density of fabrication also leads to new problems. It is becoming increasingly difficult to verify that chips which are manufactured match their functional and operating specifications. Since a percentage of devices manufactured usually do not comply with operating requirements, it is important to screen them out by testing the manufactured devices in an efficient and cost−effective manner. Unfortunately test cost is known to increase at a faster rate than device size. Additionally, due to non-functionality of tests, applied over-testing can affect yield and impact future operations of device functionality [1, 2, 3]. When a chip is tested, usually power dissipation is more than functional specifications leading to over−stressing of tested chips, causing reliability failures and defects that may usually not show up in normal functional operations. It is therefore important to test chips without compromising chip quality and reliability.

In this thesis, we report methods that address some of these concerns. We address two problems with tests in this work: test data volume (amount of data that needs to be loaded into a tester) and test application time . Test data volume for a given circuit is directly proportional to the number of independent test data input sequences known as test vectors, hence reducing the number of test vectors reduces test data volume. Usually test data volume is proportional to test application time:

time (cycles) required to apply a test to a circuit under test in the context of present work; however, exceptions are known [4]. We present two novel techniques that we introduced [5, 6] to reduce test set sizes in this report. Next, we will also propose work on test power reduction in the latter part of this thesis [7].

1.1 Testing Basics

A manufacturing test is typically performed once the chip has been manufactured to screen for defective circuits [1, 2, 8]. A typical industrial test environment consists of a circuit under test (CUT) and equipment called Automatic Test Equipment (ATE). Usually for economic reasons, test time on an ATE is costly. An ATE has finite memory and stores some vectors to apply them to the chip. ATE subsequently observes the generated response of these vectors and compares them to expected ”golden” responses. If the chip response deviates from golden response, in most cases the chip is marked as faulty and is discarded. In some cases, if the chip complies with most of the existing parameters but does not meet stringent operating criteria like speed or temperature, the chip can still be used after marking (binning) as a lower−quality chip that may be used by low end customers. In case of memories, some connections may be reconfigured to render the chip re-usable; however, it requires having some additional redundancy on the chip itself [2].

The test vectors stored on an ATE and golden responses are generated using a software program named ATPG (Automatic Test Pattern Generator) [9, 10]. As the name suggests, an ATPG automatically generates a test set that can possibly separate faulty and non-faulty chips. ATPG usually can operate on two circuit types combinational and sequential circuits.

A combinational circuit has no memory element in it. It is comparatively easier for an ATPG to generate test patterns for combinational circuits, since current input values exclusively determine output values. Test generation for sequential circuits requires, sequential elements in the circuit to be set to desired values. In this case, the ATPG needs to create test sequences over many clock cycles to set desired sequential elements to a desired value. Sequential circuit test generation is typically more complex in comparison to combinational circuit test generation.

ATPG commonly uses scan−inserted designs for test generation. Scan insertion converts a sequential design into a combinational one in test mode [11]. Flip-flops in any design are usually not controllable or observable from the output. A proper initialization sequence may be required to initialize a flip-flop. We connect flip-flops in design in the form of shift registers named scan-chains to provide full controllability − a way to initialize flip-flops with a given value and observability − to observe what value was present in a given flip-flop. Next, we discuss more details about scan chains.

1.1.1 Scan Chains

Scan design provides the necessary controllability of a flip-flop by connecting all flip-flops in a design in the form of one or more shift registers during testing (in test mode) called scan-chains, as shown in Figure 1.1 . When the test mode signal

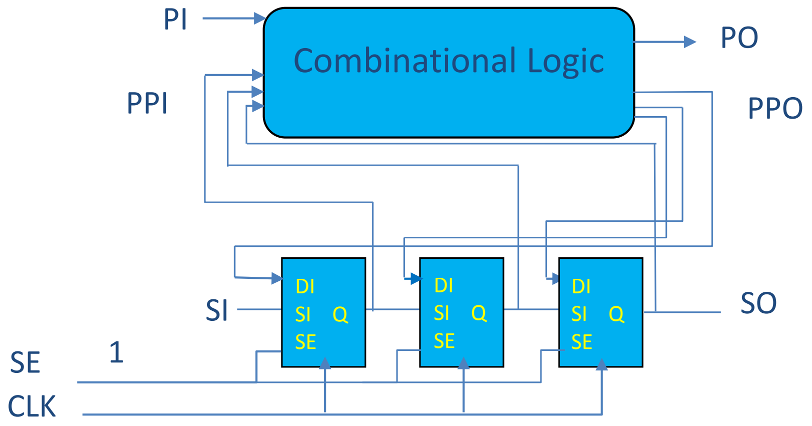


Figure 1.1: Scan Chains

is disabled, the circuit is logically equivalent to its functional mode. The inputs and outputs of the shift registers obtained are made into primary output and input, respectively. Because there are two input sources for a scan cell (flip-flops in a scan chain), a selection mechanism must be provided to allow a scan cell to operate in two different modes: normal/capture mode, and shift mode. In normal/capture mode, data input is selected to update the output. In shift mode, scan input is selected to update the output. If n is the number of flip-flops in a scan-chain we can control or observe any flip-flop in maximum n shifts. Various scan chain methods are used. The commonly used methodology is replacing design flip-flops with Muxed scan flops. Since scan-chains provide full controllability and observability, the output of a scan-cell is similar to a primary input and is called Pseudo-Primary Input (PPI) and,

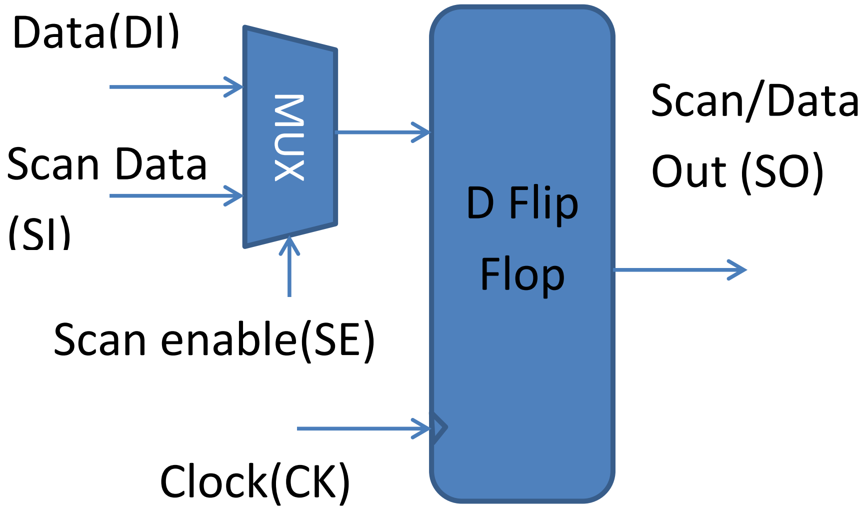


Figure 1.2: Muxed – Scan Flip-flop

similarly, input of a scan-cell is essentially a primary output called Pseudo-Primary Output (PPO), as shown in Figure 1.1.

Muxed Scan Cell: This is the most widely used scan-cell design [1, 2]. Muxed scan cell shown in Figure 1.2 consists of a D flip-flop and a multiplexer. The multiplexer select input is controlled by scan enable (SE) input to select between the data input (DI) and the scan input (SI). Muxed scan cell is the most common form of scan cell used in practice.

ATPG operates on a logical abstraction of real defects called faults. Test generation complexity is heavily dependent on the target fault type. In the next section, we discuss some common fault models.

1.2 Fault Models

Fault models are logical abstractions of real defects. Using a logical fault model makes it easier to simulate the impact of a real defect, to a large extent. However, in practice, fault models have their handicaps and not all defects can be modeled accurately with the same fault model. Using logic−level fault models can simplify simulating the fault effect caused by the real defect. Fault models, such as stuck-at fault model, bridge fault mode, open fault mode, transition fault model, path delay fault model, and cell internal fault mode are most commonly used.

Stuck-at fault model: A stuck-at-v [1] fault on a line l forces a logic value v on the line l. Line l can be primary inputs, primary outputs, gate inputs and outputs, fanout stems, and fanout branches. A stuck-at fault can be of two types: stuck at a constant value v, where v is 0 or 1, referred to as stuck-at-0 (SA0), or stuck-at-1 (SA1), respectively.

Transition fault model: In some cases, gate 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 [12]. A transition delay fault occurs when the time required for switching outputs from 0 (1) to 1 (0) in the gate, due to a change in the gates inputs, takes substantially (infinitely) longer than its normal time. Slow change from 0(1)-to-1(0) is called slow-to-rise (fall) fault.

Path Delay Fault Model: The path delay fault model focuses on the testing of a set of predefined paths in order to detect the cumulative delays along these paths [13]. A path is a set of gates which starts at a PI/PPI and ends at PO/PPO. The path delay fault model takes the cumulative delays along a path into account, while the transition fault model accounts for gross gate delay on a gate input or output. Path delay fault is more comprehensive but not used extensively, due to complex ATPG process and unmanageable test set sizes.

Cell Aware Fault Model: Some fault models were proposed to describe the defects inside a (complex) gate. Instead of modeling the defects at inputs and outputs of gates, the internal defect models −− such as stuck-at fault, stuck-open fault, resistive-open fault and short/bridge −− represent the internal defects existing in and between transistors and interconnects inside gates [14].

Bridge fault model: An unintended short between two circuit elements is commonly referred to as a bridging fault [15, 16]. These elements can be transistor terminals or connections between transistors and gates. The case of an element being shorted to power (VDD) or ground (VSS) is equivalent to the stuck-at fault model; however, when two signal wires are shorted together, bridging fault models are required.

The number of faults detected by a test set (group of tests) can be reported in two formats:

- Fault Coverage: Percentage of faults detected out of the total number of faults present in the modeled fault set.
- Test Coverage: Percentage of faults detected from the list of detectable faults.

The above fault models are most commonly used during test generation with ATPG. In most cases, stuck-at fault ATPG is modified to generate test sets for other fault models. Typically ATPG uses combinational or scan−inserted circuits to generate necessary test sets.

1.3 Scan Types

Some other common scan variations that are used are:

Clock Scan Cell: Input selection in clocked scan is conducted using two independent clocks; data clock DCK and shift clock SCK [1, 2]. In shift mode, SCK is high and is used to shift data into scan-cell. If data needs to be shifted, then DCK is used to shift data into the scan cell.

Level Sensitive Scan Cell: In order to guarantee race-free operation, three clocks −− A, B, and C −− are applied in a non overlapping manner. Advantage: The scan can be applied to latch based design [1, 11].

Replacing all flip-flops in a design with scan cells and stitching them in scan chain is called full scan architecture. Various types of other scan architecture were proposed in literature:

Partial Scan: Unlike full-scan design, where all storage elements in a circuit are replaced with scan cells, partial-scan [17] design only requires that a subset of storage elements be replaced with scan cells and connected into scan chains.

Random Access Scan: Makes each scan cell randomly and uniquely addressable, similar to storage cells in a random-access memory (RAM) [18]. Its advantages lie in reducing test power and test data required for testing. However, routing (interconnections wire length) and additional hardware required are high.

Enhanced Scan: Enhanced scan [19] increases the capacity of a typical scan cell by allowing it to store two bits of data that can be applied consecutively to the combinational (functional) logic driven by the scan cells.

1.4 Built-In Self-Test

Traditional test techniques that use ATPG to target single faults for digital circuit testing have become quite expensive. One approach to alleviate these testing problems is to incorporate Built-in Self-Test (BIST) [1, 20, 2] features into a digital circuit at the design stage. With logic BIST, circuits that generate test patterns and analyze the output responses of the functional circuitry are embedded in the chip.

BIST has several advantages. BIST eliminates the huge test data volume necessary to store the output responses for comparison. Moreover, a circuit embedded with BIST circuitry can be easily tested after being integrated into a system or in-field. Another advantage of BIST is that it aids multiple detection of a fault, which is known to detect more non-modeled faults.

A typical Built-in-Self Test (BIST) [1, 20] scheme shown in Figure 1.3 consists of a test pattern generator (TPG) (usually a linear feedback shift register) which automatically generates test patterns. Usually random patterns are used, however variations such as weighted random patterns or patterns with low−switching activity may also be used. The test patterns generated by a TPG are applied to the inputs of the circuit under test (CUT). The output obtained is observed in a compressed form called signature by an output response analyzer (ORA). The output signature

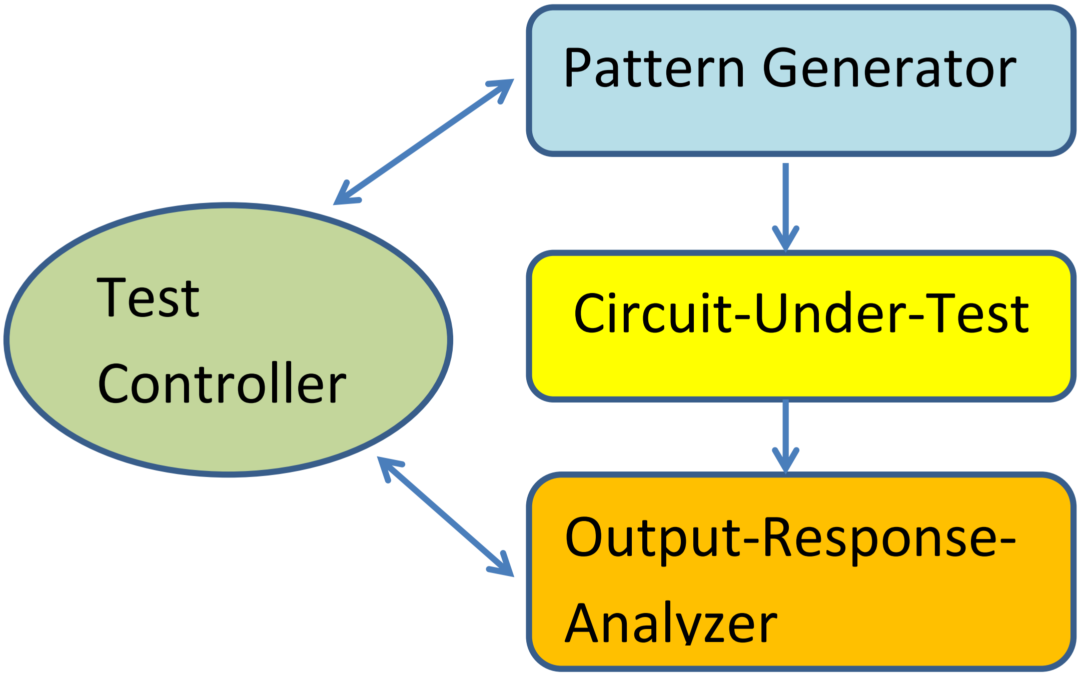


Figure 1.3: BIST scheme

observed by ORA is rendered useless even if a single unknown value referred to as X reaches the ORA. An application−specific BIST controller applies scan-enable, clocks, and other control signals in a required sequence. The logic BIST controller compares the obtained response with the expected response captured by ORA, and decides on pass/fail.

However, logic BIST patterns are usually aided by other patterns like stored patterns or externally applied patterns to achieve complete coverage. Problems faced by logic BIST are:

- Typically fault coverage is very low.

- Extra hardware in the form of test points (hardware to make faults in the CUT testable by pseudo-random patterns) is required to increase coverage [21, 22].
- X-bounding [23] (masking all x-sources by known value during testing) needs to be performed to suppress all Xs in the design from reaching the ORA.
- Usually large numbers of patterns are required.

A design has to be rendered BIST-ready (with test points and x-source blocking) to apply BIST for testing. Next, we discuss directly generating patterns using an ATPG.

1.5 Test Generation

Test generation process strives to generate a test set which can detect all the targeted faults. Faults are usually of the same fault model, but a mix of faults from various fault models can also be used.

The patterns in the test set are ultimately intended to detect defects occurring in the circuit due to flaws in the manufacturing process. Test generation consists of two main steps: fault activation and (fault effect) propagation [1, 9], described below.

- *Fault activation:* Fault activation sets the signal on a given line to a given value, depending on the fault model. For stuck-at faults, a value opposite to that produced by a fault is used in order to excite the fault. In case of transition fault model activating, a fault needs to be assigned two different values to fault site in consecutive clock cycles.
- *Fault propagation:* Fault effect propagation means to set proper values to the

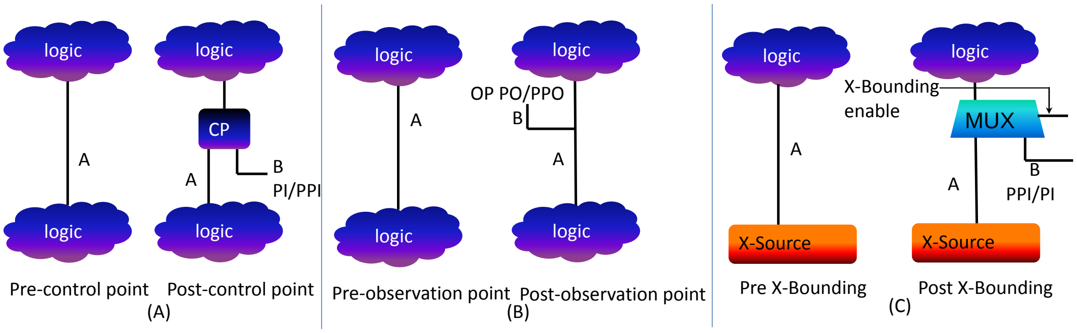


Figure 1.4: Control/Observation point and x-bounding logic

off-path inputs along a selected path so that the path is sensitized and the fault effect can be observed at an observation point.

Example: Given an AND gate with inputs A, B and output C. To detect a stuck at 1 fault at the input A, the fault needs to be activated by placing an opposite value of the fault type on A, i.e. a 0. To propagate the fault effect to output C, B should be 1 (if B is assigned a value of 1 the output value is uniquely determined by the value at line A). In case of a fault−free circuit, C is assigned a 0; if a stuck-at 1 fault exists on A, C takes the value 1.

1.6 Test Points and X-bounding Logic

In BIST, the fault coverage is limited by the presence of random pattern−resistant faults. Test point [21, 22, 24, 25, 26] adds control points and observation points for providing additional controllability and observability to improve the detection probability of random pattern resistant faults so they can be detected during BIST. However, test points can also be used to reduce test set sizes for an ATPG [24]. Test points are of two types: control points and observation points. Figure 1.4 (A) and Figure 1.4(B) show a basic implementation of control points and observation points, respectively. In Figure 1.4(A), a control point is inserted to increase the controllability of line A. A two input AND (OR) gate is used in place of CP for zero(one) control point. A zero control point needs only one input B as 0 to place a 0 on line A.

Similarly, an observation point in its simplest form is a simple connection from line A to PO/PPO, as shown in Figure 1.4(B). Any values on line A can be directly observed on PO/PPO.

A control point $v \in (0, 1)$ is typically inserted at a line l to set its value to a value which is difficult to obtain through circuit input values; the difficulty is measured by what are called testability measures, of which SCOAP [1] is the most commonly used. When SCOAP [1] controllability of v is high indicating that setting l to v is difficult, a typically large number of inputs are needed to be set to desired values to justify the value v on l. The addition of a control point reduces SCOAP controllability of v to the least value, and ATPG decisions for a value v on line l are typically guided toward control points if ATPG makes decisions based on SCOAP values. Similarly the addition of an observation point reduces SCOAP observability of a line, and an observation point is used for facilitating observation of fault effect values referred to as D or $\overline{D}$. The insertion of test points impacts ATPG decisions using controllability and observability. This may have an adverse effect on the detection of $yet - undetected$ faults during the generation of small test sets unless ATPG decisions are properly guided.

An X-source is any circuit element whose output can exhibit unknown (X) behavior during a test. Because any unknown (X) value that propagates directly or indirectly to the output response analyzer will corrupt the signature and cause the BIST design to malfunction, no unknown (X) values can be tolerated. Any unknown (X) source in the BIST-ready core, which is capable of propagating its unknown (X) value to the ORA directly or indirectly, must be blocked and fixed using a DFT repair approach often called X-bounding. Figure 1.4(C) shows a typical X-bounding mechanism using a MUX. A MUX control point is placed on the line feed by X-source to place a constant 1 or 0 on it.

An X-source in a circuit is typically assigned very high controllability; bounding X-source reduces SCOAP controllability for both 0 and 1 for the X-source. Making justification decisions to be guided toward X-bounding [22] logic reduces the number of specifications required for line justification. Hence, X-bounding has the similar impact as a 1 or 0 control point on a line l.

1.7 Test Compression

With ever increasing device complexity and size, new kinds of defects are becoming dominant and test sets for existing fault sets are growing. Tests for additional fault models like transition and bridging fault models are becoming necessary. All the above result in the increase of data that is needed to be stored on a tester. However, tester memory is costly and it may be impractical to increase tester memory.

To deal with this issue, test compression schemes were proposed. Various

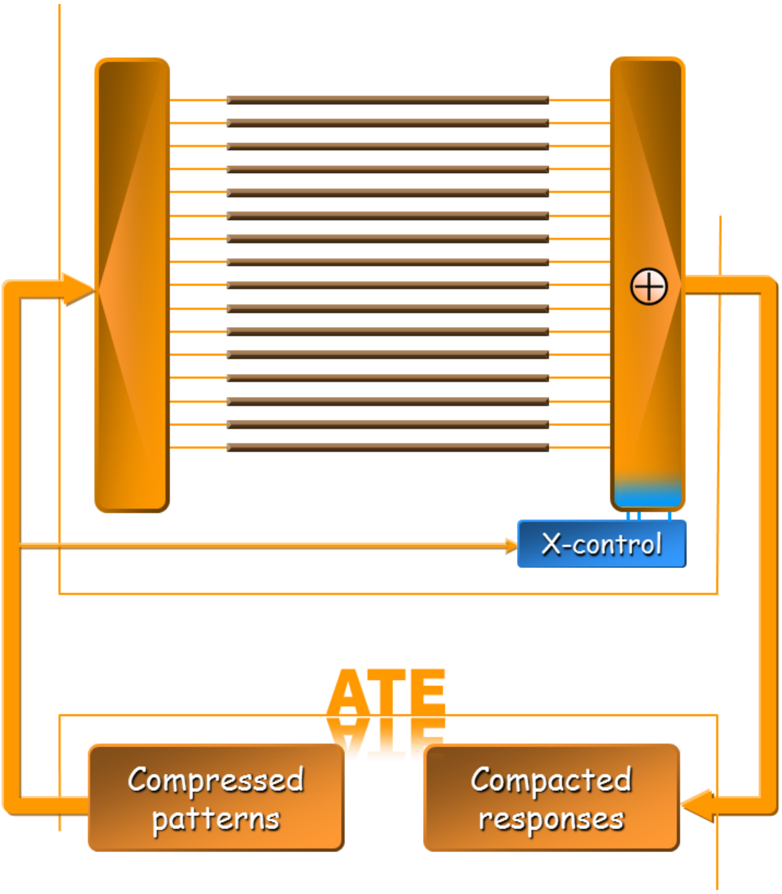


Figure 1.5: EDT Architecture

compression schemes reduce the amount of data needed to be stored on the tester. Koneman [27] was the first to propose the idea of LFSR reseeding, which forms the backbone for some of the commonly−used compression schemes.

Another advantage of test compression schemes is that a low−cost tester may be used for testing instead of upgrading the tester. Test compression schemes usually consist of an on-chip decompressor that decompresses the encoded test stimuli values, and a compactor that compacts the output response obtained.

Test data is highly compressible as typically only 1 to 5% of the bits in a test have specified values. Specified values of test cubes are known to be highly correlated due to structural relations in the circuit.

In this work, we use designs with Embedded Deterministic Test (EDT) schemes [28] shown in Figure 1.5. EDT architecture consists of an on-chip decompressor located between the external scan input ports and the internal scan chains, as well as an onchip selective compactor inserted between the internal scan chains and the external scan output ports. The test data decompressor is implemented by a ring generator (optimized LFSR) and a phase shifter. The phase shifter is necessary to drive a large number of scan chains and to reduce linear dependencies between sequences entering the scan chains. In addition, the phase shifter design guarantees balanced use of all memory elements in the ring generator. Scan cells are divided evenly, if possible, into several shorter scan 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 maximum compression possible.

1.8 Test Compaction

By test compaction or generation of compact tests, we mean the generation of small or minimal−sized test sets to achieve the desired fault coverage. Two methods exist to generate minimal test sizes. The first approach is to use extra hardware such as test points. Another method uses software techniques to produce minimal test set sizes. In this work, we use software techniques. Several works have considered methods to generate compact (small) test sets . Generally, test−compaction procedures use static compaction or dynamic compaction or a combination of both [29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]. Static compaction procedures merge compatible test cubes generated to detect individual faults [32, 33, 34]. They also use fault simulation of tests in test sets in different order(s) than the order in which the tests were generated to drop some tests in the test set. Tests are dropped without losing fault coverage. Dynamic compaction procedures use the unspecified values in a test cube to detect additional faults [29, 30, 31, 32, 35]. Generating tests with a minimal number of specified values help reduce the number of tests generated using static and dynamic compaction procedures. Tests with a minimum number of specified values also facilitate encoding of test cubes for designs with test compression logic, and hence aid generation of compact test sets for such designs.

A simple example of static compaction can be the case in which compatible vectors are identified in a test set made up of incompletely specified vectors. For example, vectors 11X1 and X101 are compatible. These two vectors can be combined and replaced by a vector 1101.

Dynamic compaction, on the other hand, tries to intelligently fill in the unspecified positions (Xs) in the test vectors such that more undetected faults can be detected. For example, consider an AND gate with inputs A, B and output C. Stuck-at 1 fault at C can be detected by vector (A=0, B=X); however if we specify B=1, a test for stuck-at 1 at C is also a test for stuck-at 1 at A.

1.9 Overview of the Thesis

In Chapter 3, we give details about test set compaction making judicious ATPG decisions. We consider the generation of compact test sets for designs using on-chip test compression logic. Even though the proposed procedure can be used for designs without compression logic, the experiments we performed are all on designs with test compression logic. We introduce two new controllability and observability measures. Additionally, we introduce a way to randomly choose between the two proposed measures. One of the new controllability/ observability measures is meant to facilitate the detection of yet undetected faults similar to the Controllability/ C-observability measures in SCOAP and these values are periodically updated. However, the proposed measures for a line l differ from those of SCOAP as they take into consideration the yet undetected faults in the fan-in cone and fan-out cones of l. The main contributions of the completed work are:

- The new method to generate compact test sets is software−based and applicable to designs with and without on-chip test compression logic.
- Two new controllability and observability measures are given to guide line justification and fault propagation steps of ATPGs.
- A method to randomly select between the above two measures, depending on the logic level of the gates, is proposed.

- The techniques presented can be used together with some of the earlier methods such as delayed justification [39], and fault clustering [38].
- Experiments on several industrial designs show that the proposed techniques, on average, reduce the number of test patterns by about 24%, relative to that generated by a state−of−the−art commercial ATPG tool.

Later, we present a method to generate compact test sets for BIST ready designs: we observed that to take full advantage of the X-bounding and test points present in BIST ready designs generating compact (small) test sets, existing ATPGs may have to be modified. ATPGs should not only generate tests that detect targeted faults; while generating tests, one should also facilitate detection of yet undetected faults to generate close to minimal test sets . Specifically, always utilizing test points to derive tests may minimize specifications in tests, but usually will not facilitate detection of yet undetected faults.

We propose a new ATPG guidance mechanism that balances between reduced specifications in tests by using test points and bounded X-sources, and facilitating detection of yet undetected faults by exploring higher specification options with comparatively lower probability. We also found that compact test generation for BIST ready designs is influenced by the order in which faults are targeted, and we utilize this observation. A new fault ordering and guidance approach is proposed for transition faults. Ordering scheme balances between fortuitous detection capability and difficulty in detection, two reasons impacting test length to generate a fault order which produces compact test sets. A new guidance approach is also proposed for transition faults.

In Chapter 4, we present a novel test data compression scheme to achieve higher compression with low toggling activity named *Isometric Compression Scheme* [7]. The new scheme deploys an on-chip power-aware test data decompressor, the corresponding test cube encoding method, and a compression-constrained ATPG that allows loading scan chains with patterns having low−transition counts while encoding a significant number of specified bits produced by ATPG in a compression-friendly manner. As a result, the new solution offers very high compression ratios and preserves all benefits of continuous flow compression. Moreover, the new scheme avoids periods of elevated toggling in scan chains and reduces scan unload switching activity due to unique test stimuli produced by the new technique, leading to a significantly reduced power envelope for the entire circuit under test.

The rest of this thesis is divided as follows: In Chapter 2, we discuss the previous work related to this thesis. In Chapter 3, we discuss the previous work in the field of ATPG for high compression. Details on the fault−based compaction mechanism we have proposed is discussed first. Later in Chapter 3, we present a new compact test generation methodology for BIST ready designs developed by us in Chapter 3. Isometric compression scheme is discussed in Chapter 4. We conclude the thesis in Chapter 5.

CHAPTER 2
PREVIOUS WORK

In this chapter, we the discuss previous work relevant to our work. First, we will discuss ATPG for test generation.

2.1 ATPG Algorithms

In this section, we describe some of the basic terminologies that will be used in this work.

Sensitized Path: A path starting at line l is said to be a sensitized path if any value change on line l changes value of one or more PO/PPO.

Fault Detection: In a full scan circuit, a fault stuck-at-v is said to be detected if line l can be activated to $\bar{v}$ and a path exists between line l and PO/PPO such that any change on line l also changes one of the specified PO/PPO.

Now we will define some more definitions commonly used by ATPG test generation programs. These are implication, justification, and fault effect propagation procedures.

Implication: It is the procedure to find all the uniquely determined signals under the current circuit state. Justification is responsible for selecting a way among several possible ways to achieve desired values on signal lines. Fault effect propagation takes care of selecting a path to propagate the fault effect and determining which signal (along the path) should be set to what value.

Justification: The process by which the output of any gate is satisfied by values

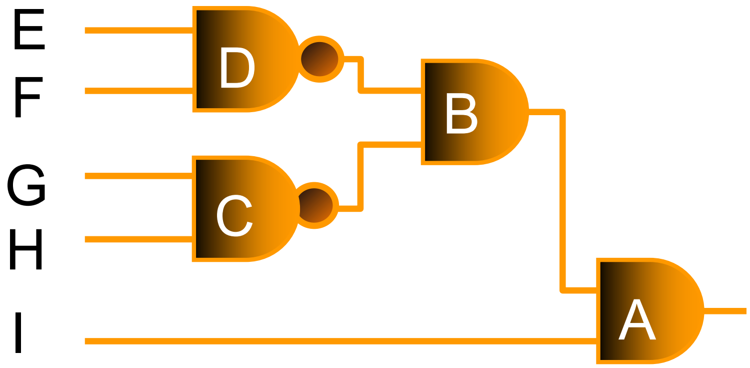


Figure 2.1: Example Circuit

at its input if many choices exist.

Propagation: Fault propagation deals with creating a sensitized path from the fault location to at least one of the PO/PPO. A sensitized path is created if one of the input is sensitized and other non-sensitized inputs are assigned non-controlling values.

Example: The fault D s-a-0 needs to be tested. To activate the fault, we require a value 1 on line D. Value 1 on A can be justified in two ways: either E or F are assigned a value of 0 to create a sensitized path between ouput and fault location. Inputs not on the sensitised path, i.e. C and I, are assigned non-controlling values. Value 1 on C can be justified by chossing G or H to justify.

Implication is the process in which all the values can be uniquely assigned by a given assignment. If B is assigned a value of 0, we know the output of gate A will be 0. Zero at B is said to imply a 0 on A. The above is an example of forward implication. However, a 0 on B does not implies in a backwards direction, as now three choices exist (not uniquely determined or implied) : 0 on D, C, or both.

2.1.1 D-Algorithm

D$-$algorithm was the first complete algorithm which searches the whole solution space, given that no constraints on run-time exist. D$-$algorithm uses a five alphabet algebra. The alphabets used are $\{0, 1, D, \bar{D}, X\}$ where D stands for signal value 1 in good circuit, and 0 in faulty circuit. Similarly, $\bar{D}$ stands for value 0 in good circuit, and 1 in faulty circuit.

The generation of a test for a stuck-at fault, say line r stuck-at-a, $a = 0$ or 1, starts by implying value $\bar{a}$ on line r, where $\bar{a}$ is the complement of a. This creates a D or $\bar{D}$ on line r using the D notation introduced in [9]. Next, the value on line r is justified by assigning appropriate values to (some) input(s) of the gate, driving line r, and the D or $\bar{D}$ on line r is propagated towards the observed outputs through selected intervening gates.

The line justification and propagation of Ds are facilitated by using two data structures called J-frontier and D-frontier, respectively . The entries in the J-frontier are output lines of gates whose outputs have specified values not yet implied by the values of the inputs of the gates. The entries in the D-frontier are gates for which some input(s) have D or $\bar{D}$ values and the outputs are unspecified. Justifying the required value on a line r in the J-frontier is done by assigning a proper value to one of the inputs of the gate driving r. Propagating D or $\bar{D}$ to an observed output is done by selecting a gate from the D-frontier and propagating the D or $\bar{D}$ from its input(s) to its output. The selection of a proper input to be assigned to justify a line value in the J-frontier, and selecting a proper gate from the D-frontier to propagate D or $\bar{D}$, is important to obtain tests with a minimum number of specified values, to obtain minimal size test sets and to reduce test generation times. In this work, we will use an ATPG based on D-algorithm for test generation. However, the ATPG is highly enhanced by the use of some other heuristics.

2.1.2 PODEM

It was proposed by Goel [10], and aims at using direct search instead of specifying values at internal nodes. PODEM (path oriented decision making) makes all the decisions on the PI/PPI. The major highlights of PODEM are:

- Decisions are made only at primary input, as a result of this no−backward implication is needed in the PODEM.
- Propagation decisions are made by choosing the gate on D-frontier which is closer to the PO/PPO of the gate.
- X-path checking a new method to detect the existence of no solution is introduced. It checks that an unspecified path, x-path, exists between current D-frontier and PO's. If no such path exists, than fault can not be propagated using current D-frontier and a backtrack has to be performed.

- Backtrace is used to justify lines; PI's in the fan-in cone are iteratively assigned values unless the value needed to be justified is satisfied.
- PODEM is a complete algorithm, like D-algorithm, and can generate a test vector if one exists, given no constraint on run time.
- PODEM is approximately 2.5 times faster than D-algorithm, as reported by the author in [10].

The main disadvantage of PODEM is that it has difficulty identifying redundant faults as only assignments to PI are made.

2.1.3 FAN

FAN [40] was a successor of PODEM , and borrows some of the ideas of PODEM and D-algorithm. The main highlights of the FAN are:

- Introduces a new concept called unique sensitization. When only one prorogation path exists, we should propagate the D-frontier through it and, at the same time, justify the off-path inputs of gates on the propagation path to non controlling values. It is similar to propagation−first methods described later.
- Does both forward and backward implications like D−algorithm, leading to fast detection of conflicts or redundant faults.
- Unlike PODEM, FAN does not backtrace any J-frontier gates to primary inputs. Instead, it stops at certain circuit lines called headlines. Headlines are the output signals of fanout-free regions. Due to the fanoutfree nature of each cone, all signals outside the cone that do not conflict with the headline assignment

would never require a conflicting value assignment on the primary inputs of the corresponding fanin cone. In other words, any value assignment on the headline can always be justified by its fanin cone. Because each headline has a corresponding fanin cone with several primary inputs, this allows the number of decision points to be reduced.

- Multiple Backtrace: Since we now use unique sensitization, more than one objective may be required to be satisfied simultaneously. Multiple backtrace procedure handles this.
- Usually FAN results in up to 2-5X reduction in test generation time in comparison to PODEM.

2.1.4 SOCRATES

The main points about Socrates [41] test generation algorithm are:

1. In Socrates [41], both forward and backward implications are used. Also, contrapositive implications are also learned. If we know that (a=0) **implies** (b=1) then we know (b=0) **implies**(a=1) is also true.
2. While static implications are computed one time for the entire circuit, dynamic implications are updated dynamically during the ATPG process.
3. At a given step in the ATPG process, some signals in the circuit would have taken on values, including D or D.
4. This set of values may imply other signals which are currently unassigned to necessary value assignments.

5. In general, dynamic implications work locally around assigned signals to see if any implication can be derived.

2.1.5 Recursive Learning

Learning specially dynamic learning is time consuming. Efficient ways were needed especially for hard−to−detect faults. Recursive learning to generate test or declare them redundant for hard−to−detect faults [42]. Recursive learning tries all possible solutions to a problem and finds the intersection set of these solutions. The intersection set must be true for a solution to exist. A complete set of pair-wise implications could be computed. In order to keep the computational costs low, a small recursion depth can be enforced in the recursive learning procedure.

2.1.6 SPOP

Single path oriented fault effect propagation (SPOP) [43] is an idea different from traditional ATPGs which consider propagation and justification simultaneously. In SPOP, we first built a sensitized path; after a sensitized path is built, all justified lines are subsequently justified. Creating a sensitized path before justification accelerates the identification of redundant fault.

2.2 Test Set Compaction

Compaction techniques can be divided into two sub−categories: dynamic and static compaction. Dynamic compaction technique modifies the ATPG procedure so that every test pattern generated is incrementally specified to detect more undetected faults [29, 30, 31, 32, 35]. Static compaction, however, deals with a complete test set; any generated test set is reordered to modify or drop test patterns that are redundant [32, 33, 34]. Test generation process is not impacted by static compaction. Most of the state−of−the art techniques that result in the smallest test set sizes use a combined approach of static and dynamic compaction. In combined approaches, generally a dynamic compaction procedure is applied first, followed by a static compaction procedure.

Next, we describe some of the previous work in the field of compaction. We discuss test set compaction for scan designs and stuck-at-fault models.

2.2.1 Compaction Approach by Goel and Rosales

Authors in [29] were the first to identify the benefits of reducing the test set size by using compaction procedures. Two methods - static and dynamic compaction - were used in this work. In dynamic compaction, a test vector is generated for a fault f. The remaining undetected faults are targeted incrementally by specifying the non−specified positions of the current test vector, where the current test vector is the test vector originally generated for fault f and other faults which were successfully detected by incrementally specifying test vector for f. New faults for given test vectors are targeted until all PIs are specified, or maximum threshold or possibility of success is low. In this work, static and dynamic compaction were not combined and were treated as separate methods.

During static compaction tests for all faults in the test set are generated and the compatible test vectors are subsequently merged, therefore run time for static compaction was very high.

In the end of the paper, the authors concluded that dynamic compaction procedures are more effective in generating compact test sets in comparison to static compaction procedures. Static compaction procedures take more memory and time as test vectors for all faults in the test sets were targeted.

2.2.2 COMPACTEST

Compactest was proposed in [30] and gives one of the best results. Compactest combines the following new concepts to attain a low test set compact test.

- Fault ordering has an impact on test pattern count. Two faults are said to be independent if they do not have a common test vector that detects both of them; example, any stuck-at 1 and stuck-at 0 faults on any input of an AND gate. Independent set fault lists for each FFR are generated and formed into groups. Groups of faults are arranged according to the number of faults in the group. Independent set fault list in an FFR are generated using an efficient algorithm; the largest independent fault list (for FFR's) is targeted first.
- Potential compatible fault list (faults which can be detected by the same vector) is generated and used as an input fault list for dynamic compaction procedure.
- A method named maximal compaction was introduced, which starts with a generated test vector. If any specified PI is non-essential then after inverting the value of that PI - if the targeted fault will still be detected by the modified

test vector. Specified position at the PI is not essential and can be changed to X. Commonly, this method is mentioned as relaxation in test literature. Even though it is not guaranteed to be a test, the probability of a relaxed test not being able to detect a previously detected fault is very small.

- Rotating Backtrace: Usually, testability measures are used to guide backtrace decisions. Testability measures are computed once and used for the whole test generation flow. Usually, testability measures guide ATPG toward the easiest possible option. This leads to ATPG being guided toward the same set of nodes, or searching the same search space multiple times. Compaction suffers as ATPG is less likely to be guided toward non−targeted faults. A good way is to rotate between the possible options to facilitate the detection of undetected faults.

2.2.3 ROTCO

Reverse Order Test COmpaction (ROTCO) [44] is a static compaction procedure. Its input is a complete test set generated by any other ATPG tool. ROTCO works with faults that were detected only once during the test generation flow. Usually, a lot of inputs remain unspecified even after a complete test set generation. It might be possible to specify these unspecified locations so that a test is generated for some other faults other than those currently detected by the current test vector. If a fault f is detected only once during test set, a test vector T_i specific to that fault is required in the test set. If by incrementally specifying some other test vector T_j detects fault f, and no fault exists that is only detected by T_i in the complete test set, T_i is redundant and can be removed from the test set.

The patterns were modified in the reverse order to that of test set generation. This method can be applied even after any dynamic compaction approaches.

2.2.4 COMPACT

Compact is a very simple and efficient test generation procedure and has been widely used [31]. The fundamental difference between this and other procedures is, instead of trying to generate a test for a fault by a same vector, it generates vectors individually for faults and merges them Compatible faults usually have similar vectors that can be merged together. The overall approach is described below:

1. Generate a test for a set of faults from yet undetected faults. Store it in a buffer.
2. Go to the first test vector T_i in the buffer for fault f; if it is compatible to any other vectors in the buffer, merge the other vectors to T_1.
3. Fault simulate T_1 after merging and drop the faults that are detected by T_1. Remove the vectors corresponding to these faults if one exists in the buffer.
4. If there are undetected faults left, go to step 1.

The main advantage of COMPACT is that the procedure is fast and yet produces reasonably good results.

2.2.5 CODEM

CODEM (Compaction Oriented Decision Making) proposed in [36], was developed for industrial circuits. The main ideas of the approach are as follows:

- Modifies decision making process for test generation to facilitate the detection of a present set of undetected faults. Both propagation and justification decisions are modified by using a new dynamic testability measure instead of static SCOAP based measures.

- Testability measures are changed so that ATPG is guided in a manner that test vectors for undetected faults are generated by incrementally specifying the unspecified PIs of the test generated for the current targeted fault.
- Testability measures are updated after the generation of every test pattern for a fault, followed by dynamic compaction.
- However, the approach results in only 6% reduction in test length.
- The approach can easily be applied to large industrial circuits, as comparable overheads in terms of memory and run time are low.
- Also, redundant faults were identified in the start of the ATPG process and are used to prevent ATPG from re-targeting redundant faults again, leading to wasted ATPG effort.

2.2.6 Double Detection and Two-by One

Double detection and Two by One techniques were proposed in [32]. The improvement was done over COMPACTEST.

2.2.6.1 Double Detection

Highlights of the double detection procedure are:

1. *Fault Ordering:* Fault ordering is based on an independent set of faults, with faults in FFR with the largest number of independent faults first. The only

change is that the target list is updated dynamically, based on the current set of undetected faults.

2. *Changes to Rotatory Backtrace:* Rotating backtrace should be used for different vectors, while generating a test for the same vector, it should not be used.
3. *Double Detection:* A test set is generated so that every fault is detected a minimum of two times. Fault simulation is used, and tests that detect essential faults (faults not detected by just one test vector in test set) are simulated first. All the test vectors that do not detect any new faults are dropped. Fault simulation is repeated until every vector contains one essential fault.

2.2.6.2 Two-by-One Approach

Two-by-One is a static compaction technique and can be used after generation of any test set. The method works as follows:

- First we extract the essential faults of each test vector.
- In the second step, all independent fault sets are identified. If a fault pair is not independent, two separate vectors are required to generate tests for them.
- If two faults in a pair are not independent, a new test vector is generated that detects all essential faults detected by both vectors. If such a vector is generated, replace both vectors with this new test vector.
- If any such pair of faults still exist, repeat the procedure for them.

Experimental results showed that the combination of double detection, two by one, improved fault ordering, and improved backtrace procedure gives shorter test length than if each of these procedures were used individually.

2.2.7 Set Cover Method

This paper maps the test compaction problem to a set cover problem [45]. In this case, the problem was mapped to airline scheduling problems. The problem can be thought of as the following:

- Suppose test vectors are sets and faults are elements in a set.
- Set cover problem is defined as a minimum number of sets (test vectors) that are required to cover all elements (faults).
- The problem is known to be an NP Complete problem, so no good solution exists. The only hope is to use an approximate solution.
- The problem was mapped to airline scheduling problems, and an LP relaxation based solution was proposed.
- Since the number of faults is huge, only a given number of faults are considered once and a solution is generated for them.

The solution resulted in a good test length, and can even reduces the test length of some previously known procedures.

2.2.8 Mintest

The Mintest works on two techniques [35]: essential fault removal (EFR) and redundant vector elemination (RVE). In EFR, a test vector is taken and all essential faults are assigned to other vectors, by incrementally specifying them. Even if all the essential faults can not be assigned to other vectors the status of vectors are preserved and the possibility of removal is increased, as some vectors that are redundant may be left with fewer faults after each iteration. Fewer essential faults may lead to increased probability of the essential faults being assigned to other vectors.

2.2.9 Forward Looking Fault Simulation

Reverse-order fault simulation (ROFS) was found to be efficient in quickly reducing the size of a test set. However, with only ROFS, it is possible for some redundant vectors that don't contain any essential fault to be retained. Prior to using the technique [46], the original test set is simulated and each fault is associated with an index which links to the vector that first detects the fault. Next, the test set is re-simulated in reverse order. A fault is dropped from the fault list as soon as it is detected by a vector. A vector i is dropped in this process if either it detects no new fault, or no fault it detects has index i.

If no fault detected by i has index i, that means there is at least another vector that detects the fault that will be simulated in the future. Hence, there is no need to keep vector i since all faults it detects are going to be detected by later vectors. Experimental results showed that using forward-looking fault simulation yields a more compact test set than using reverse-order fault simulation.

2.2.10 Pattern Generation for Embedded BIST

In this work [47] the authors used a hybrid BIST scheme where random patterns are applied first, and then patterns are generated for a BIST scheme having constraints on encoding. Only a given number of specified positions can be encoded in every cycle. The ATPG has some interesting highlights :

- ATPG was guided based on undetected faults.
- If no undetected fault detection is facilitated by ATPG, a decision leading to the minimum number of specification is used.
- If D-frontier has a controlling value for gate on propagation path, any faults on the other inputs of the gate are not observable and no faults in the fan-in of these input can be detected, leading to a minimum specification decision.
- Both static and dynamic compaction schemes were used to generate compact tests for embedded BIST.

2.2.11 Tests With Large Number of Unspecified Bits

In this work a dynamic and static compaction scheme was proposed for circuits with a large number of unspecified bits.

- *Static Compaction:* A given test vector is considered, and specified bits are relaxed so that all essential faults detected by the compaction schemes are detected even after relaxation and fewer specified bits.
- *Dynamic Compaction:* A fault is considered compatible if a single test detects a fault, possibly with less number of specified bits. A compatibility graph is constructed so that the vertices's in the graph consists of fault, and the edge weight represents the number of specified bits if two faults are compatible. At first a few faults are targeted and compatibility relations are explored by merging compatible faults together if a test vector detecting both of them with a low

number of specifications exists. In later operations, more faults are considered.

2.2.12 Delayed Justification

Delayed justification approach was proposed in [39]; it has long been known that headlines [40] can be justified later during the test generation process. The same concept is applied to compression with some modifications:

- Headlines are identified not only statically but dynamically, depending on the circuit node values; this leads to an increase in the number of headlines.
- A fanout is allowed in the fan-in cone of the headline, resulting in the possibility that conflicting values can be assigned. However, as number headlines with conflicting values assigned to them are low in comparison to the total number of headline candidates available after this relaxation.
- Headlines are justified in the end of the test generation process, giving more opportunity to improve encoding by using available free variables.

2.2.13 Compact Test Generation Based on Compatible Faults

In this work [37, 38], compatible fault clusters are formed based on necessary assignments. If two faults have non-conflicting necessary assignments, they are said to be compatible.

First, a group CA is formed of all necessary assignments required to test a given cluster of compatible faults NA. While targeting a fault, effort is made to honor assignments in corresponding CA. ATPG justification and propagation decisions are made so that assignments in CA are not conflicted.

Main disadvantage of this approach results from two phased ATPG, once for learning followed by ATPG for actual test generation.

2.2.14 QBF Based Test Pattern Generation

Recently, a quantified boolean formula (QBF) based on compact test generation approaches was proposed to generate provably optimal compact tests[48]. Similar to SAT, required test properties were mapped to a logic formula. For each input, candidate a multiplexer was encoded to dynamically un-specify the input. Another approach to generate accurate test patterns in the presence of the unknown was given in [49]. This approach removes the pessimism of conventional fault simulation algorithm and can accurately classify faults as untestable or testable.

2.3 Compression to Achieve Low Toggling and Reduced Data Volume

Power dissipation in the circuits can be categorized into two types: static power dissipation and dynamic power dissipation. In static power dissipation, the power is dissipated by a gate when it is inactive. Static power dissipation is currently negligible [3]. Dynamic power dissipation, on the other hand, is due to current requirements, as a result of the charging and discharging of inherent load capacitances in CMOS. The power dissipation is directly proportional to the load capacitances which are charged or discharged at every toggle. Hence, reducing toggle is an important requirement for low power testing.

In this chapter, we give the overview of the proposed future research works. Compression is becoming a mainstay and is typically added to reduce test data volume during a scan test. It is known that a scan-chain test might have a switching rate many times higher than a functional test. Any typical scan-pattern may result in power requirements that are above normal functional mode requirements [50]. High toggling during tests may also result in over-stressing, reliability failures, and may even result in yield loss [3]. In this work, we propose to develop new methods to reduce shift power during test in compression environment.

The problem of reducing power dissipation for scan-test can be divided into two main categories: reducing shift power and reduction of capture power, power dissipated during the scanning out of the test response. Power during scan testing has been addressed by partitioning [51, 52], test scheduling [53], reordering of scan-chains [54, 55], and filling of unspecified positions [56, 57].

2.3.1 Test Vector Filling Techniques

Some earlier methods try to reduce shift power by assigning a constant value to unspecified positions. To address capture power, methods like preferred fill try to reduce hamming distance between captured values and applied test vectors. Specifically, preferred fill uses a technique based on signal probabilities. Inputs are assigned equi-probable probabilities for 0 and 1.

Preferred fill calculates the signal probabilities of each PPO. If the signal probability of a scan-cell capturing 1 is high, a 1 is assigned to corresponding PPI if it is unspecified, or else a 0 is assigned [56]. Another method named JP fill uses a signal probability and justification based method to reduce power supply noise.

2.3.2 Scan Cell Reordering

A scan cell reordering technique is proposed in [54, 55] to reduce switching activity, and hence power dissipation in scan design by changing the order of scan cells in each scan chains. The best order of cells is the one that gives the best compromise between reducing the transitions in the scan cells during both scanning in test patterns and scanning out captured responses.

2.3.3 Clock Gating to Reduce Capture Power

Clock gating logic is common in modern designs to reduce dynamic power. In test mode, scan-enable signals override the clock gate enable signal, which leads to successful scan-shift operations [58, 3]. However, in capture mode clock gating logic can be used to disable capture by some scan-cells as during the capture cycle, the scan−enable is inactive and the clock gate is controlled from its functional enable. The the functional enable is judiciously controlled by an ATPG capture power and can be reduced without impacting the ability to capture fault effects. Figure 2.2 shows a typical clock gating circuit.

2.3.4 Previous Work In Low Power Compression

Some of the above techniques like design partitioning and test scheduling can be used for low power compression. Some popular low−power test methods to reduce power in compression environment are addressed in [59, 60, 61, 62] :

Reseeding to reduce toggle rates, authors in [63] propose a method that reduces the number of transitions and specified bits at the same time.

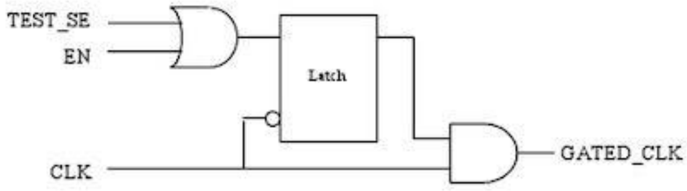


Figure 2.2: A typical clock gating circuit

We use EDT compression environment in this work [28]. A low power EDT compression scheme is shown in Figure 2.3. Typically, the number of specified scan-cells is around 1% of the total scan cell. Rest scan cells are filled at random using the decompressor, leading to unusually high switching activity. Due to such a low percentage of specified scan cells, only a few scan-chains have specified scan-cells in it. Scan-chains without any specified scan-cells can be assigned a constant value either 0 or 1. Assigning a constant value will reduce the number of transitions for that scan-cell. However, it is important to note that since all specified scan-cells contain specified values necessary to detect a fault, all faults targeted by the pattern are detected with this low transition pattern.

The overall scheme shown in Figure 2.4 works as following: input channels provide the compressed input data to the decompressor. Input channels are also routed to control the register which drives an XOR tree, which in turn drives the AND (OR) network at the input of scan-chains. If a 1 exists at the input of AND gate, scan-chains are directly driven by a decompressor; else in case of 0, a constant

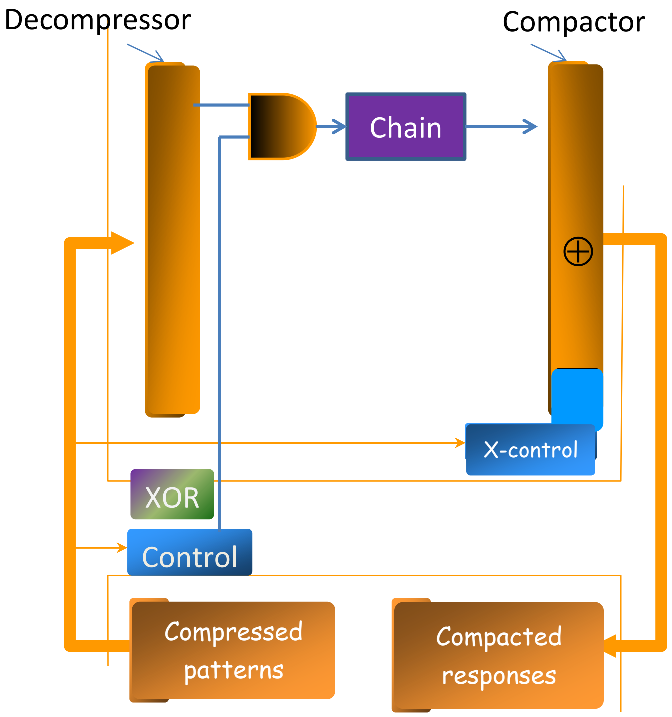


Figure 2.3: Low Power EDT

value of 0 is shifted in the corresponding scan chain.

We now define two terms which will be used frequently in this work:

Scan Slice: All the scan cells that are shifted in the same scan cycle are referred to be of the same scan-slice.

Free Variable: Every bit stored at the tester is called free variable as it can be assigned either 0 or 1.

Due to the low number of specifications for a test cube, it is possible that for a few scan slices – no new specification is required, or requirements for test generation are satisfied even if the same data is supplied to scan chains for a given k number of

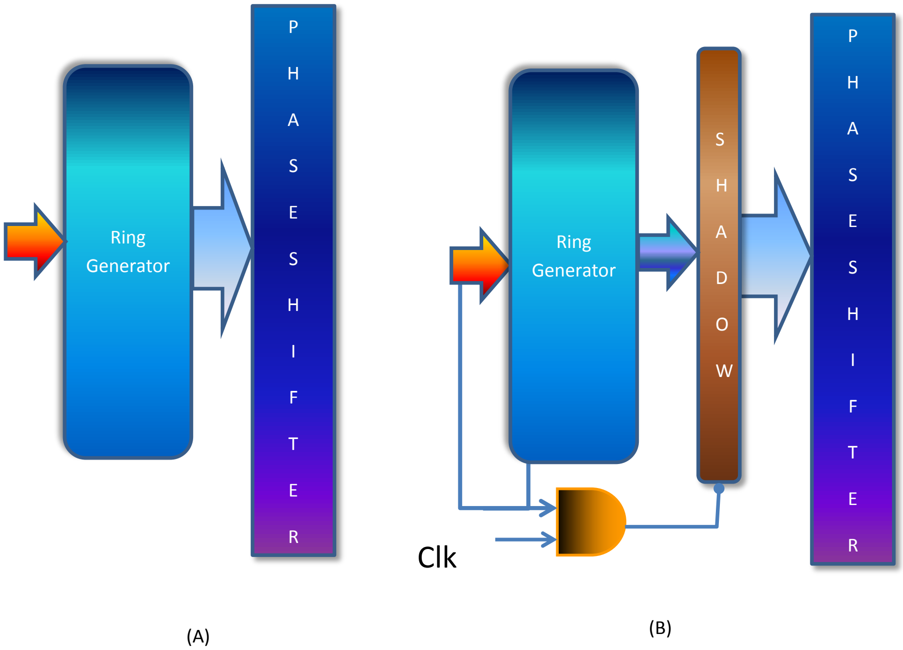


Figure 2.4: Decompressor (A) Original (B) With shadow register

shift cycles. The above mentioned approach further reduces transition count and thus reduces power during the test. New and old decompressor hardware is shown in Figure 2.4 [61]. However, this scheme requires some hardware changes; a new mechanism was added so that ring-generator changes state and accept new free variables which can be used for the encoding of any specifications after hold cycles in which no free variables were consumed. The above was achieved by adding a shadow register between the ring generator and the phase shifter; the same data is provided by the ring generator for k consecutive cycles if the shadow register hold control input is set to high, or else if hold control input is set to 0, a new state from the ring generator is fed to phase the shifter. The shadow register is transparent for a hold register set to 0.

In every shift cycle, an input channel is used to control the shadow register directly from the input channels. If a given control bit is set to 1, then the shadow register updates its state before the ring generator reaches its next state. Instead of the control channel, one can use the decompressor input channels to deliver the control information merged with the seed variables. The main advantage of the above scheme is that it can reduce toggle in scan-chains that are fed using decompressor directly, thus reducing toggle activity significantly.

The above low power compression solution described in [61], however, suffers from the following drawbacks:

- One input channel is used for all cycles to control the hold register, leading to a reduced number of free variables, those are available for encoding specified scan cells.
- High additional hardware cost of one gate per scan-chain, appropriate XOR-based controlling mechanisms and control registers are needed.
- High number of initialization cycles, as in the start we load control registers that control the AND gates which, in turn, decide if a scan-chain is loaded by constant values or decompressor.

We will address all the above disadvantages of the approach proposed, as well as improve compression by using a new hardware architecture to provide hold signal to the shadow register using an template register; the new architecture is supported by appropriate changes to ATPG flow.

CHAPTER 3
GENERATION OF COMPACT TEST SETS

In this chapter, we present our work published in [5, 6]. Typical methods to select proper inputs for line justification and D propagation use controllability and observability measures, or randomly select with equal probability, one of the unspecified inputs of gate driving lines on the J-frontier. During D-propagation randomly select with equal probability one of the gates from the D-frontier [38].

In Table 3.1, we describe equations to calculate SCOAP measures commonly used to make ATPG decisions for various gate types [1]. SCOAP controllability measures are calculated in forward levelized trace from PI/PPI toward PO/PPO. Similarly, SCOAP observeability measures are calculated in a backward levelized trace from PO/PPO towards PI/PPI. C_0 or controllability of 0 for any line is an estimate of the minimum number of inputs that need to be specified to control line l. Observability of a line l is a measure of the number of inputs that need to be specified to propagate a fault effect on line l to a PO/PPO.

Table 3.1: Calculation *SCOAP* controllability and observability values.

Gate Type	Controllability 0	Controllability 1	Observability
OR	${C_0}^C = {C_0}^A + {C_0}^B$	${C_1}^C = min({C_1}^A, {C_1}^B)$	$O^{A(B)} = O^C + {C_0}^{B(A)}$
NOR	${C_0}^C = min({C_1}^A, {C_1}^B)$	${C_1}^C = {C_0}^A + {C_0}^B$	$O^{A(B)} = O^C + {C_0}^{B(A)}$
AND	${C_0}^C = min({C_0}^A, {C_0}^B)$	${C_1}^C = {C_0}^A + {C_0}^B$	$O^{A(B)} = O^C + {C_1}^{B(A)}$
NAND	${C_0}^C = {C_0}^A + {C_0}^B$	${C_1}^C = min({C_0}^A, {C_0}^B)$	$O^{A(B)} = O^C + {C_1}^{B(A)}$
INV	${C_0}^C = {C_1}^A$	${C_1}^C = {C_0}^A$	$O^A = O^C$
BUF	${C_0}^C = {C_0}^A$	${C_1}^C = {C_1}^A$	$O^A = O^C$
SL	1	1	0

Table 3.2: Circuit Properties

CKT	Size	FF Num	Num of Chains	Longest Chain	Input Channels	Output Channels	Decompressor Size
D1	1.1M	75K	382	190	2	5	32
D2	3.3M	73K	244	300	3	3	32
D3	1.3M	52K	209	250	3	3	32
D4	107K	1640	15	115	1	1	16
D5	500K	28K	400	71	4	8	28
D6	400K	41K	400	104	2	5	22
D7	2.2M	167K	1200	142	16	16	65
D8	1.2M	75K	400	202	2	5	32
D9	1.3M	68K	658	104	3	6	64

Decisions based on SCOAP measures are known to result in test cubes with a minimum number of specified values. It has been observed that random choice based decisions (random decisions for short) most often yield smaller test sets for industrial designs in comparison to test generated using SCOAP based decisions.

In Table 3.2, we give the gate count (size), number of flip-flops (FF), number of chains and other design properties for nine industrial circuits. Next, in Table 3.3 we give the fault coverage (FC) with the ATPG tool default value for the backtrack limit and the size of test sets referred to as test length (TL), using random decisions

Table 3.3: Test length Random and SCOAP.

	SCOAP based ATPG		Random Decision ATPG	
CKT	FC	TL	FC	TL
D1	96.71	15040	96.71	18881
D2	93.31	5696	93.31	7012
D3	94.56	20032	94.56	23716
D4	95.31	4992	95.31	5406
D5	99.26	7938	99.26	7598
D6	98.63	2048	98.63	2406
D7	91	2496	91	2955
D8	97.04	8064	97.04	9636
D9	95.94	9359	95.94	11446

and SCOAP measures. SCOAP based decisions, on average, results in a 18% increase in test set sizes in comparison to a test set size obtained by a random decision ATPG.

3.1 Fault Based ATPG Decisions

3.1.1 Overview of the proposed method for line justification and D propagation

The proposed method to generate compact test sets uses three new techniques. The first one marks each gate in the design with probability p as described in Section 3.1.2. Gates are marked once in a preprocessing step. The line justification and D propagation decisions at a marked gate are made using a static measure described in the next section. The line justification and D propagation decisions at unmarked gates are made using a dynamic measure described in Section 3.1.3. The measure described in the Section 3.1.4 is called static, as it is computed once for each line in the circuit. Dynamic measure is updated periodically during test generation. Next, we describe how the gates in the design are marked.

3.1.2 Procedure to mark gates

Gates in the design are marked for line justification decisions and D propagation decisions separately. Gates are marked for line justification with a probability based on their level counted from inputs (primary and pseudo-primary), which we call forward level. Gates are marked for D propagation based on their level counted from circuit outputs (primary outputs and pseudo-primary outputs), which we call backwards level. Note that all gates at a level are marked with the same probability. In a preprocessing step, each gate at level l is marked with probability p according to

Table 3.4: Probability p calculation

forward(backwards) level l	p
$l < 20$	0.1
$20 \leq l < 100$	$l/100$
$l \geq 100$	1

Table 3.4. The intuition behind the marking procedure is described next.

Both the static and dynamic measures described later are similar to controllability and observability measures typically used in ATPG as guidance to solve line justification and D propagation. Similar to SCOAP based measures [64], the static measures we use are effective in leading to a minimal number of specified values in test cubes and the dynamic measures facilitate detection of yet undetected but not targeted by the current test being developed. The logic level based random selection between static and dynamic measures prefers static measures if the decision point is at a higher level. The intuition behind this strategy is the follows: if a line value is to be set to satisfy a line justification requirement at a higher forward level, the fan-in cone of the line tends to be large and can also involve many circuit inputs leading to many specified values in test cubes unless the proposed static measure is used to guide the selection of input to solve the line justification problem using minimal specification. Similarly, for D propagation through a gate at a higher backwards level, the D has to be propagated through many levels of gates whose off-path inputs have to be justified to non-controlling values and, in this case, the static measures guide D propagation through gates closer to the observed outputs.

3.1.3 The Proposed Static Measure

We use SCOAP [1, 64] to determine the static measures to be assigned to each circuit line. For each line l, we first compute 0 and 1 controllability, $C0(l)$ and $C1(l)$, and observability $O(l)$ using the procedure in [1] which is slightly different than the original SCOAP procedure in [64]. We then compute the new static measures $SJ0(l), SJ1(l)$ and $SP(l)$ as: $SJ0(l) = 1/C0(l)^3$, $SJ1(l) = 1/C1(l)^3$ and $SP(l) = 1/(1 + O(l))^3$. That is, the new static measures are the inverse of the cube of the classical SCOAP measures. In computing $SP(l)$ we add a 1 to $O(l)$ to avoid dividing by 0 when $O(l)$ is zero. The names for the static measures are chosen based on the following rationale: controllability measures are used in ATPG's to guide line justification decisions, and hence the static measures computed from controllability values are called $SJ0$ and $SJ1$, where S stands for static and J stands for justification. Similarly, the static measure computed from observability is called SP, where P stands for propagation of Ds done using these measures.

In Figure 3.1 we give an example to illustrate the computation of the static measures using the ISCAS85 benchmark circuit C17. In Figure 3.1, we give the values of $C0(l)$ and $C1(l)$ as pairs $(C0(l), C1(l))$ for each line and the corresponding $SJ0(l)$ and $SJ1(l)$ values as pairs $[SJ0(l), SJ1(l)]$. In Figure 3.2 we give $O(l)$ values as $(O(l))$ and $SP(l)$ values as $[SP(l)]$ for each line.

The reason for the inverse cubic measures described above is the following: instead of using controllability and observability measures directly to guide line justification and D propagation decisions, as discussed later, we use a weighted random

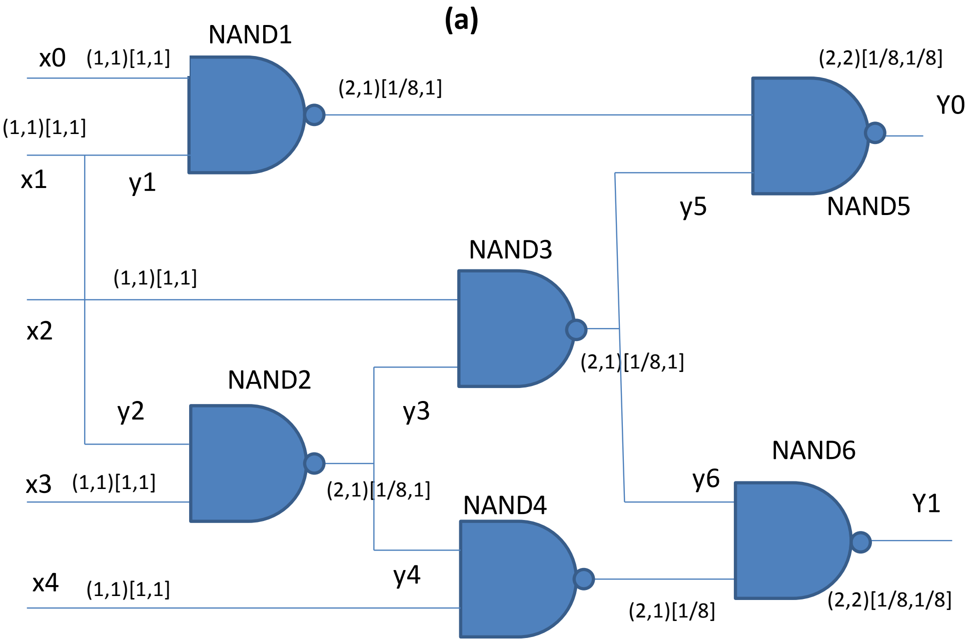


Figure 3.1: $SJ0$ and $SJ1$ measures.

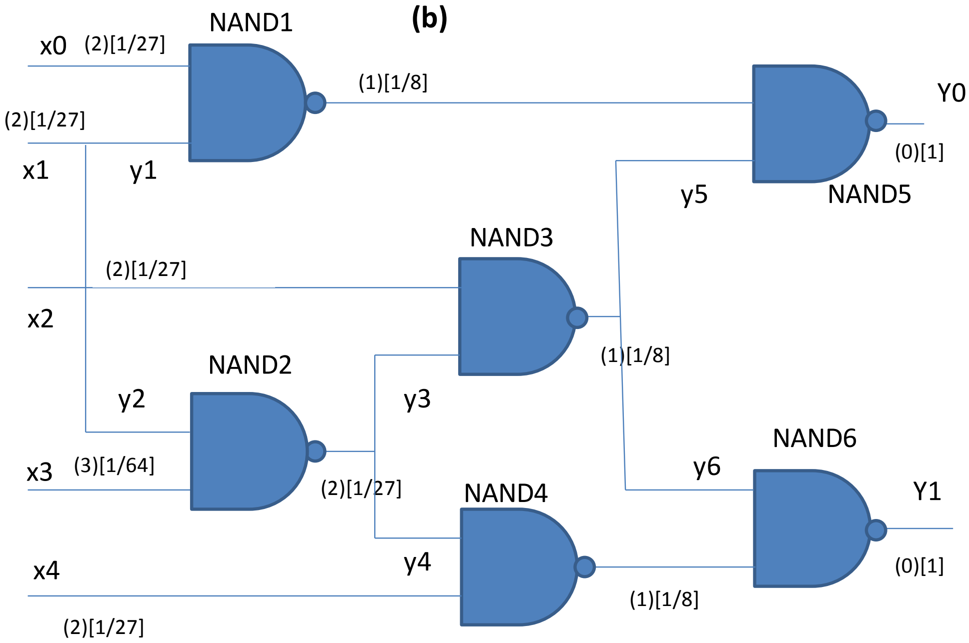


Figure 3.2: SP measures

decision based on controllability and observability measures. This allows us to balance between deterministic decisions as in the case of using the SCOAP measures directly, which typically yield minimally specified test cubes and the pure random decisions that we showed earlier lead to smaller test sets. As the experimental results presented later show, this balance is effective in reducing test set sizes. Additionally, we found that instead of using SCOAP measures directly in making the weighted random decisions for line justification and D propagation, using a function of the SCOAP measure leads to better control for balancing between deterministic and random decisions. The inverse cubic measures were found to provide this balance.

3.1.4 The Proposed Dynamic Measures

We compute the proposed dynamic measures for line justification and D propagation for every circuit line l afresh after generation of k tests, and these measures are meant to facilitate the detection of yet undetected faults. In our implementation, we used k=64.

3.1.4.1 Dynamic Measure for Guiding Line Justification (DMJ)

DMJ has two values for each line, DMJ^1 and DMJ^0, which are computed iteratively from circuit inputs to circuit outputs. DMJ^1 (DMJ^0) for a line l is an estimation of the number of faults in the fan-out free region (FFR) driving line l, which are sensitized through line l with a 1 (0) on line l in the fault-free circuit. DMJ values are calculated using the algorithm given in Figure 3.3. Only the undetected faults in the FFR driving l influence DMJ values of line l. Both DMJ values for stems

```
Require: Set $DMJ^1$ $DMJ^0$ to 0 for all lines
  for Forward trace from PI/PPI towards PPO in a levelized manner do
    if If output of gate i has a $stuck-a-1$ fault then
      $DMJ^{i^0} = DMJ^{i^0} + 1$
    end if
    if If output of gate i has a $stuck-a-0$ fault then
      $DMJ^{i^1} = DMJ^{i^1} + 1$
    end if
    for every input $j$ to gate $i$ feed by stem do
      if input $j$ to gate $i$ feed by stem then
        if  $j$ has a stuck-at-1 fault and $i$ is non-inverting gate or $j$ has a stuck-at-0 fault and $i$ is inverting gate then
          $DMJ^{i^0} = DMJ^{i^0} + 1$
        end if
        if  $j$ has a stuck-at-0 fault and $i$ is non-inverting gate or $j$ has a stuck-at-1 fault and $i$ is inverting gate then
          $DMJ^{i^1} = DMJ^{i^1} + 1$
        end if
      end if
      if  gate i is non-inverting then
        $DMJ^{i^1} = DMJ^{i^1} + DMJ^{j^1}$
        $DMJ^{i^0} = DMJ^{i^0} + DMJ^{j^0}$
      end if
      if  gate i is inverting then
        $DMJ^{i^1} = DMJ^{i^1} + DMJ^{j^0}$
        $DMJ^{i^0} = DMJ^{i^0} + DMJ^{j^1}$
      end if
    end for
    If stem $DMJ^{i1} = 0$, $DMJ^{i0} = 0$
  end for
```

Figure 3.3: Calculation of DMJ^1 and DMJ^0.

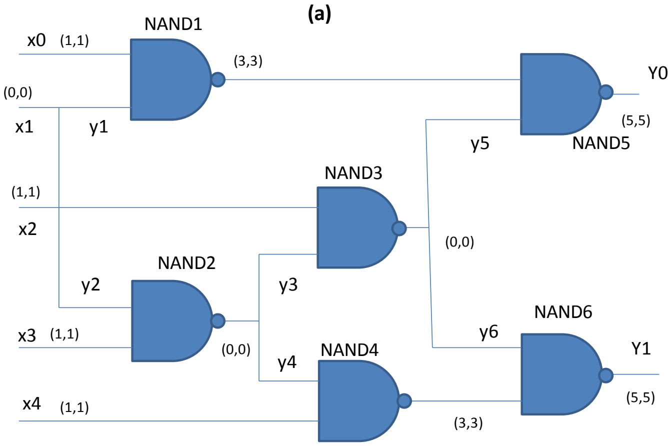


Figure 3.4: Calculation of DMJ values

are set to 0. In Figure 3.4, we give DMJ values for each line l in ISCAS85 benchmark circuit c17 as (DMJ^0, DMJ^1), assuming that both stuck-at-1 and stuck-at-0 faults on all lines of the circuit are yet undetected.

3.1.4.2 Dynamic Measure for D Propagation

Similar to the computation of observability measure of SCOAP, which is computed using controllability values, the proposed dynamic measures for propagation are calculated after calculating the dynamic measures for justification given above. However, unlike SCOAP observability, DMP have two separate values: one for 0 and one for 1. DMP^0 (DMP^1) for a line l is an estimation of the number of yet undetected faults whose propagation is facilitated by 0(1) on line l in the fault free circuit.

We need the following definitions to explain things further:

```
Require: Set $DMP^0$ $DMP^1$ to 0 for all lines
  for Backward trace from PO/PPO towards PPI in a levelized manner do
    if gate i is a PPO/PO then
      If gate i a stuck−at−0 fault $DMP^{i1} = DMP^{i1} + 1$
      If gate i a stuck−at−1 fault $DMP^{i0} = DMP^{i0} + 1$
    else
      for input's j of gate i do
        if j has a stuck−at fault then
          Add 1 to $DMP^{j1}$ for stuck−at−0 and 1 to $DMP^{j0}$ for stuck−at−1
        end if
        for $v \in \{0, 1\}$ do
          if value $v$ at j uniquely determines value for i then
            if $v$ is 1 $DMP^{j1} = DMP^{j1} + DMJ^{j1}$
            if $v$ is 0 $DMP^{j\bar{1}} = DMP^{j\bar{1}} + DMJ^{j\bar{1}}$
          else
            Collect needed inputs other than j for any fault effect to propagate to
            i, store respective lines in array L
            for every line l in array L do
              if $v = 1$ $DMP^{j1} = DMP^{j1} + DMJ_{l1} + DMJ_{l\bar{1}}$
              else $DMP^{j0} = DMP^{j0} + DMJ_{l1} + DMJ_{l0}$
            end for
          end if
        end for

        if i is inverting gate then
          $DMP^{j0} = DMP^{i1} + DMJ^{j0}$,
          $DMP^{j1} = DMP^{i0} + DMJ^{j1}$
        else
          $DMP^{j1} = DMP^{i1} + DMJ^{j1}$ ,
          $DMP^{j0} = DMP^{i0} + DMJ^{j0}$

        end if
      end for
    end if
    If stem add all values of for branches
  end for
```

Figure 3.5: Calculation of DMP^1 and DMP^0.

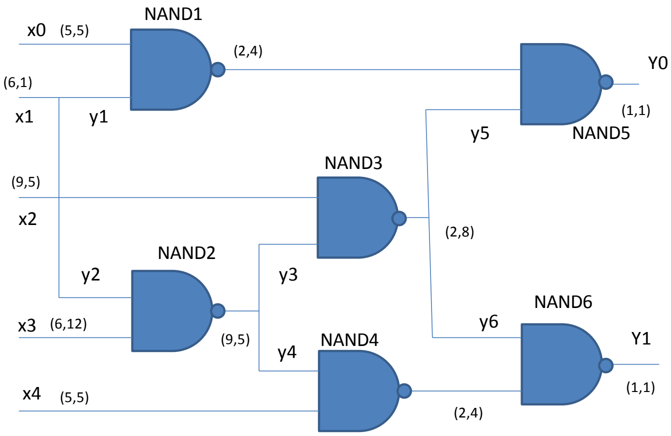


Figure 3.6: Calculation of DMP values

On Path Inputs: Inputs to a gate on the D-frontier with a D or $\overline{D}$ values.

Off Path Inputs: Inputs of a gate on the D-frontier whose current values are not D or $\overline{D}$.

For all gates along a fault propagation path if on-path inputs of a gate have a non-controlling fault-free machine value, all the undetected faults in the FFRs driving the off-path and the on-path inputs are facilitated by the fault-free value on the on-path input. However, if the fault-free value on the on-path input is a controlling value, all the undetected faults in the FFRs driving the off-path inputs are blocked by the on-path fault-free value. Therefore, depending on the on-path fault-free value, DMP values are different for 0 and 1.

The algorithm given in Figure 3.5 is used to calculate DMP values. In Figure 3.6, we give values for DMP^0 and DMP^1 in pairs as (DMP^0, DMP^1) for ISCAS85 benchmark circuit c17, assuming that both the stuck-at-1 and stuck-at-0 faults on all lines are yet undetected.

3.1.5 Line Justification Decisions

Line justification decisions justify a required value at a line l in the J-frontier by assigning a controlling value to a selected input of the gate driving line l, which currently has an unspecified value. If the gate is marked, the decision is based on static justification measures $SJ0$ or $SJ1$ for all the unspecified inputs $ip_1, ..., ip_k$ of the gate. We make a weighted random decision based on the probabilities computed from $SJ0$ or $SJ1$ values of inputs $ip_1, ..., ip_k$. If the controlling input of the gate is $v \in (0, 1)$, then the probability of picking input ip_j to justify the line value is:

$p(ip_j) = SJv(ip_j)/\sum_i^{i=1...k} SJv(ip_i)$

As an example, consider a 3 input AND gate which is marked, and we need to justify its output to be 0 by implying a 0 on one of its three inputs. Let the inputs of the gate be a, b, and c and the static justification values, $SJ0(a), SJ0(b)$, and $SJ0(c)$ be 2, 4, and 8, respectively. Then the probability of selection of a, b, or c is p(a)=2/14, p(b)=4/14, and p(c)=8/14.

Instead of using the SJ measures directly to make decisions, we use a weighted random decision which have the following advantages: It causes decisions to typically yield minimum specifications of inputs with a very high probability. However, a degree of randomness is added by assigning lower probability of selection to other inputs with lower SJ measures. Secondly, in the case of two inputs having very similar SJ values close to minimum of values at a gate, SJ based decision will always choose the same one of the two possibilities. However, weighted random approach will assign similar probabilities for selections of either two.

Given a gate g on D-frontier, inputs $ip_1, ..., ip_r$ are on-path inputs and have a D or $\overline{D}$ on them, if any on-path input $ip_1, ..., ip_r$ has a controlling value in the fault free circuit, undetected faults in the FFRs in the fan-in cone of the off-path inputs cannot be detected by the test under generation. Gates in such fan-in cones are called blocked. If a gate is blocked, all gates in the same FFR in the fan-in cone are also blocked. Any decisions made on blocked gates are made using static testability measures $SJ1$ or $SJ0$ to minimize the number of specified values in the test.

3.1.5.1 Line Justification Decisions Using Dynamic Measures

Dynamic guidance measures DMJ are used for justification decisions at a gate in the J-frontier if the gate is un-marked.

Line justification decision based on dynamic measures DMJ^1/DMJ^0, for all the unspecified inputs $ip_1, ..., ip_k$ of a gate on the J-frontier is used to select an input with probability computation similar to the case of using static measures discussed above. If the controlling value of the gate on the J-frontier is $v \in (0, 1)$:

$p(ip_j) = DMJv(ip_j)/\sum_i^{i=1...k} DMJv(ip_i)$

3.1.6 D Propagation Decisions

D propagation decisions are required when we have multiple gates on the D-frontier and one of them is selected using a given criteria to propagate a D or $\bar{D}$ to the output of the selected D frontier gate by implying non-controlling values on the unspecified inputs of the selected gate. Decisions are divided into two parts: when at least one of the gates on the D-frontier with multiple gates is marked, and when no gate on the D-frontier with multiple gates is marked.

We first start with analyzing if any gate on the D-frontier is marked and, in case it is, we pick a gate among the marked gates using the proposed static D-propagation measures $SP(l)$ for the gate picked. If the marked gates on D-frontier are $g_1, ..., g_k$, we make a weighted random decision based on the probabilities computed from SP values of $g_1, ..., g_k$. The probability of picking gate g_j to propagate a D is:

$p(g_j) = SP(g_j)/\sum_i^{i=1...k} SP(g_i)$

In the case no gate on the D-frontier is marked, then we use the dynamic D-propagation measures, DMP, to compute the probability of selecting a gate. If $g_1, ..., g_k$ are the D-frontier gates for which a D or $\overline{D}$ propagates to output if all un-specified inputs of the gate are assigned a non-controlling value, assign an array $A_1, ..., A_k$ so that if input g_i is D, $A_i = DMP1(i)$ or else $A_i = DMP0(i)$. The probability of picking gate g_j for propagation is:

$p(g_j) = A_j/\sum_i^{i=1...k} A_i$

3.2 Generating Compact Test Sets for BIST Ready Designs

In this section, we describe the next part of our work on generating compact tests for BIST ready designs accepted in [6]. Test data compression schemes usually do not insert test points or X-bounding logic [28, 65, 66], and deterministic tests are generated for all detectable faults [28]. Xs in test response are partially suppressed by using X-masking logic to mask some or all Xs at scan-chain outputs [28]. Since the bits to mask Xs in this manner need to be encoded together with the specified bits of test cubes for BIST ready designs, X-bounding and test points used can have a significant impact on test data compression. Post X-bounding, no input data is required for X-masking as no Xs exist in test responses. X-bounding, therefore, impacts both input and output test data volumes. Test points reduce the number of specifications in tests [24, 22], as well as facilitate higher test response compaction. By specifications for (in) tests, we mean the number of specified bits in tests. Reduction in the number of specifications facilitates test encoding in case of designs with test data compression. Another important factor effecting compact test pattern generation is fault ordering. Faults whose detection facilitates detection of other faults should be targeted first [67, 68, 69].

To take full advantage of X-bounding and test points in generating compact (small) test sets, existing ATPG may have to be modified. ATPG should not only generate tests that detect targeted faults; while generating tests, one should also facilitate detection of yet undetected faults to generate close to minimal test sets [5, 30, 47]. Specifically, always utilizing test points to generate tests may minimize specifications in tests but usually will not facilitate detection of undetected faults. We propose a new ATPG guidance mechanism that balances between reduced specifications in tests by using test points and X-bounding and facilitating detection of undetected faults by exploring larger specification options with comparatively lower probability. We also found that compact test generation for BIST ready designs is

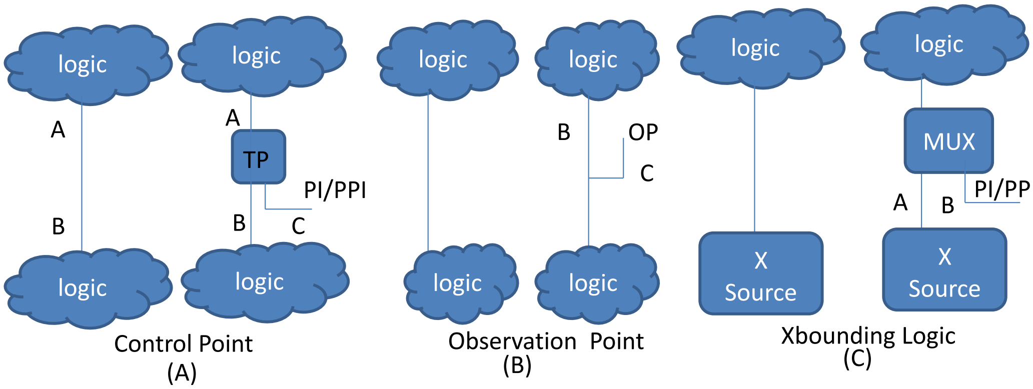


Figure 3.7: (A) Control Points (B) Observation Points(C) X-bounding

influenced by the order in which faults are targeted [67]. Fault-ordering and new methods to guide ATPG to achieve compact test sets for BIST ready designs are the focus of the work presented in this thesis.

3.2.1 ATPG Decisions Using Existing Techniques

Shortcomings of previous ATPG guidance approaches in dealing with BIST ready designs is discussed next. While using random decisions for line justification and/or D-propagation, the probability of the selection of test points is significantly reduced as now an ATPG will select test points and other costlier options with equal probability. In Figure 3.7 (A), we illustrate two versions of the same circuit: pre and post insertion of an AND type control point (TP) on a line A-B. C is the pseudo-primary input used to control the control point. ATPG using random decisions: to justify a 0 at an AND control point will have equal probability of selection of input's B or C for justification. Random decision results in under-utilization of test points and over-specification, even if detection of any other undetected faults is not facilitated.

While using an ATPG guided by faults [5, 36] guiding ATPG decisions towards C can facilitate detection of only one additional fault. ATPG, when guided toward B, typically facilitates the detection of a larger number of faults with a comparatively larger number of specified positions. Since ATPG decisions are based on the number of undetected faults, almost all justification decisions will be guided toward B, resulting in under-utilization of test points and difficulty in encoding.

A SCOAP based ATPG will always guide an ATPG towards a test point as it is easier to control and observe, hence reducing the possibility of facilitating test generation for undetected faults in the fan-in cone of B.

In Figure 3.7(B), we illustrate a case of pre and post observation point insertion on a line. A new observation point OP is added to line B with a new PO/PPO. In case of an observation point, as shown in Figure 3.7(B), a random decision ATPG will equi-probably choose between observation point C and fan-out B for propagation, leading to higher specifications. A SCOAP based method will always guide ATPG toward an observation point OP, which will prevent facilitation of test generation for a large number of faults along propagation path using B. If the current set of undetected faults is used as a guidance measure [36, 47], ATPG will almost always be guided toward B rather than the observation point OP, leading to higher specifications and difficulty in encoding.

In Figure 3.7(C) we illustrate a pre and post X-bounding scenario of an X-source. X-source A is X-bounded using a MUX and insertion of another PI/PPI B.

While using X-bounding, a random decision ATPG will pick randomly among X-source A and a fixed value at the input of MUX inserted for X-bounding B, leading to unsuccessful justification decision in 50% of cases.

An ATPG guided by undetected faults is oblivious of the presence of an X-source and will be guided based on faults, which by high priority are the same due to the presence of a s-a-1 or s-a-0 at A and s-a-1 or s-a-0 at B. Therefore, both random and fault based ATPG fail to make good decisions in the presence of X-bounding logic.

SCOAP guided ATPG will work well with X-bounding logic as X-sources are assigned ∞ values for SCOAP controllability of 0 and 1, thus ATPG will not be guided toward X-source A; other possibilities i.e. B will be chosen for justification decision.

The above discussion illustrates the shortfalls of the current ATPG guidance measures in handling designs with test points and X-bounding logic.

3.2.2 Overview of ATPG for BIST Ready Designs

We consider the issues discussed above to augment a state of the art commercial ATPG to generate more compact tests for BIST ready designs. A novel fan-out-free region (FFR) based fault ordering scheme which facilitates detection of undetected faults is proposed. We propose a new ATPG decision methodology to obtain compact test sets for BIST ready designs with test points and X-bounding. ATPG decisions are guided by the proposed S and M measures and using a weighted

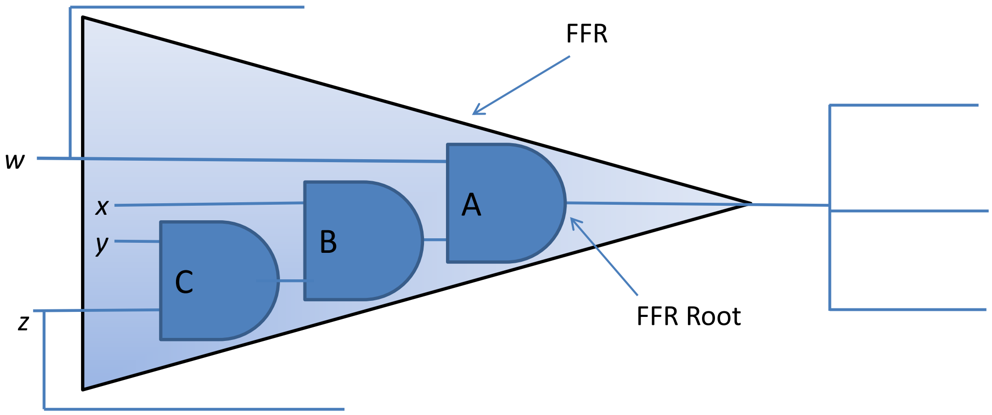


Figure 3.8: Fan-out free region.

random decision approach to balance between effective usage of X-bounding and test points without compromising detection of yet undetected faults during dynamic compaction.

3.2.3 FFR Based Fault Ordering

In this section, we will describe the proposed fault ordering method. Faults whose tests lead to detection of other undetected faults should be targeted first [68, 69]. We propose a simple FFR based fault ordering heuristic. We start with some basic definitions.

FFR Root: Output gate of an FFR.

Gate Height: Number of gates needed to be traversed from a gate to the FFR root is the height of the gate. Inverters and buffers are ignored in this count.

Gate height can be calculated by running a preprocessing step to partition the circuit into FFRs and then assign every gate its corresponding height.

FFR Leaf: A gate which has either an FFR root, a scan cell, or a primary input at one of its fanin.

Example: In Figure 3.8, we show an FFR with root A. The height of gates A, B, and C are 0, 1, and 2 respectively. The heights of inputs *w, x, y,* and *z* are 0, 2, 3, and 0, respectively.

Consider two undetected faults in the current circuit B s-a-1 and C s-a-1. If a test for B s-a-1 is generated as (1,0,X,X), clearly no possibility exists to generate a test for fault C s-a-1 by incrementally specifying the unspecified inputs of the circuit. However, if fault C s-a-1 is targeted first, every test for C s-a-1 is also a test for B s-a-1. Instead of two test vectors required to generate a test for both fault a single vector is feasible.

The above example illustrates that the ordering of faults is important. Any fault at a greater height (i.e. further away from the output (root) of the FFR) will require propagating a D or $\overline{D}$ from the faulty gate to the root of the FFR through a larger number of gates. If such a D or $\overline{D}$ is propagated, all the compatible faults along the path from the gate will also be detected. Also, it may possibly help in detecting other faults, as propagation of a D or $\overline{D}$ translates to the setting of other inputs of the gates on the path, to non-controlling values. Thus, considering faults in a given FFR, tests generated for faults at higher level (greater height) gates usually facilitate detection of more undetected faults in comparison to faults at a lower height (closer to FFR root).

The importance of fault-ordering can also be explained using the basic concepts of test compaction. For any compaction scheme the number of specifications per detected fault can be considered as a figure of merit. Low specification per fault typically results in lower pattern count by facilitating the detection of other undetected faults [36, 47]. Height based ordering has a greater impact for BIST ready designs, as typically FFRs are smaller in BIST ready designs due to the insertion of test points.

3.2.3.1 Height Based Ordering of Faults

In height-based ordering of faults, we order faults in descending order of height. However, if a fault list is generated in a level by level trace of a circuit, chances are that adjacent entries in the list are from the same FFR. Targeting faults in the same FFR usually use similar paths for propagation of a D or $\overline{D}$ from the FFR root, leading to these tests facilitating the same set of undetected faults. We avoid listing faults in the same FFR consecutively by combining random shuffle and height based ordering, as described in the procedure given in Figure 3.9. In lines 2-4, we organize list F_i in sorted order based on the heights of faults in their respective FFRs. Lines 8-14 deals with division of the list into two parts: LISTA which contains odd positions of F_i, and LISTB contains even positions, respectively. In line 15, we randomly shuffle LISTB containing half the number of faults. Lines 17-28 deal with the formation of a new list F_f with i_{th} even positions occupied by LISTB[i/2]. When i is odd assign fault at LISTA[i/2] to $F_f[i]$. In the above procedure, since half of the final fault list is still sorted according to their FFR height, faults in deep FFRs are given priority, but

```
Require: Initial fault list $F_i$ of size $n$.
  Output: An ordered fault list $F_f$
  Fault ordering is run at the start of test generation process
  Randomly shuffle fault list $F_i$
  for All faults in list $F_i$ do
    Arrange faults in descending order of FFR height of their respective faulty gates
    //Height is number of multiple input gates travered from fault location to the
    root of the ffr containing the fault
  end for
  i=0;j=0;k=0
  Create two empty lists LISTA=empty, LISTB=empty
  //Store odd positions of sorted list in LISTA and even in LISTB
  while $i < n$ do
    if i is odd then
      LISTA[j]=$F_i$[i], j=j+1, i=i+1
    else
      LISTB[k]=$F_i$[i], k=k+1, i=i+1
    end if
  end while

  Randomly shuffle LISTB
  //Create new list with odd position occupied by LISTA and even by randomly
  shuffled LISTB
  i=0;j=0;k=0
  Create empty list $F_f$, for storing final ordered fault list
  while $i < n$ do
    if i is odd then
      //Odd position of final list contains faults in increasing FFR heights
      $F_f$[i]=LISTA[j], j=j+1, i=i+1
    else

      // Even position in the list contains randomly shiffled faults
      $F_f$[i]=LISTB[k], k=k+1, i=i+1
    end if
  end while

  $F_f$ is ordered fault list for ATPG
```

Figure 3.9: FFR height-based fault ordering procedure.

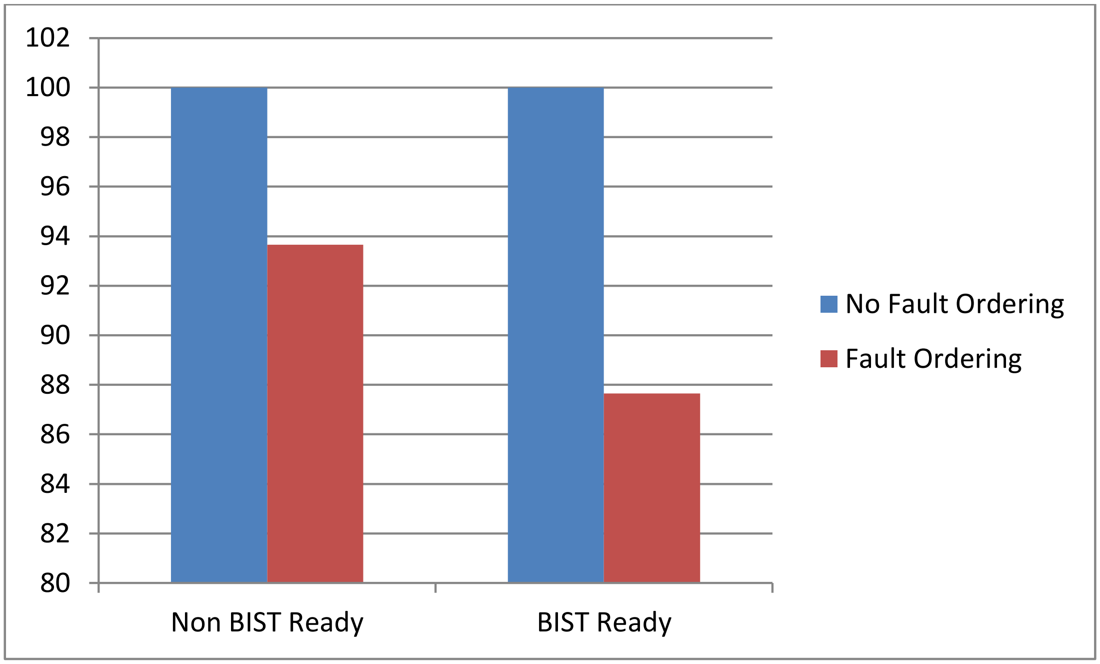


Figure 3.10: Number of specifications and fault ordering.

random selection of half of the list reduces the probability of the selection of faults from the same FFR.

In Figure 3.10, we compare the number of specifications per detected faults for a non-BIST ready design with and without fault ordering, normalized to 100 for an ATPG without fault ordering. The same comparison for BIST ready designs is also given.

As shown in Figure 3.10 a non-BIST ready version of the circuit results in only a 6% reduction in the number of specifications due to fault ordering, however, in the case of a BIST ready design, the number of specifications per detected faults is reduced by 11%.

3.2.4 ATPG Guidance Measures

We use two SCOAP based measures, called S and M guidance measures, to guide ATPG decisions. We will first describe how to calculate them, followed by a brief discussion on their use for guiding ATPG decisions. SCOAP appropriately captures the following:

1. Presence of a test point, as observability/controllability values change to minimum values.
2. Post X-bounding controllability of any line l driven by an X-source is reduced significantly.
3. No periodic update is required during the test generation process, compared to some other ATPG methods, to generate a compact test as in [5, 36].

The exploration of new circuit areas is necessary for creation of compact tests [30, 36]. Weighted random approaches used in this work explore all options with some probability. Decision options resulting in a high number of specifications are used with lower probability.

3.2.4.1 S Guidance Measures

The S guidance measures are derived from SCOAP measures, previously used to guide ATPG in [29] and are calculated for every line [1]. For a given line l where S stands for SCOAP, $SJ1(SJ0)$ are SCOAP based justification measures for 1(0), and SP stands for SCOAP based measures to guide D-propagation. The original SCOAP [1] measures are $C1(l), C0(l)$, and $O(l)$ where $C1(l)$ is the 1 controllability, $C0(l)$ is the 0 controllability, and $O(l)$ is the observability of a line l, respectively. The S measures for line l are, $SJ1(l) = 1/C1(l)$ and $SJ0(l) = 1/C0(l)$. These are used for guiding line justification decisions. $SP(l) = 1/(O(l)+1)$, where $SP(l)$ is the measure to guide ATPG propagation decisions. A 1 is added to $O(l)$ as an observability value of 0 and can lead to invalid division operations. Next, we discuss our second SCOAP based measure used to guide ATPG.

3.2.4.2 M (Minimum) Guidance Measures

Minimum (M) measures are also derived from SCOAP [1]. Minimum measures for a given line l are:

$MJ0 = 1/(C0^3)$, $MJ1 = 1/(C1^3)$, $MP = 1/((1+O)^3)$,

where $MJ0(MJ1)$ are minimum justification measures for 1(0) and MP is the minimum propagation measure, respectively.

These measures were used in [5] and were called SJ and SP measures. While using the above measures, instead of making decisions directly based on them, we make weighted probabilistic decisions based on the measures described above. As discussed earlier, test points and X-bounding usually lead to reduction in SCOAP testability values and corresponding S and M measures increase accordingly. Creating a balance between minimum number of specification and guiding ATPG toward other non-minimum specification options, to facilitate detection of undetected faults, is a must for creating compact tests for compression. We create this balance by using a mix of S and M measures.

3.2.5 Justification Decisions

Justification decisions are required for a gate g in the J-frontier [9] whose outputs have specified values not yet implied by its inputs. A controlling value is assigned to one of the unspecified inputs of the gate, driving the gate on the J-frontier. While making a justification decision, we calculate probabilities of selection for all unspecified inputs using both M and S justification measures. Later, we calculate a mixed probability based on both S and M measures to make a final decision. Let the unspecified inputs of a gate g on the J-frontier be $ip_0, ..., ip_m$. If $v \in (0, 1)$ is the controlling value for gate g. The probability of the selection of ip_j to justify v on gate g using S guidance measures is given by:

$p(ip_j)^S = SJv(ip_j)/\sum_i^{i=1...m} SJv(ip_i)$

Similarly, the probability of the selection of ip_j using M guidance measures is given by:

$p(ip_j)^M = MJv(ip_j)/\sum_i^{i=1...m} MJv(ip_i)$.

The final probability actually used for the selection of input ip_j is the average of the two probabilities $p(ip_j^S)$ and $p(ip_j^M)$:

$p(ip_j)^{mixed}{=}(p(ip_j)^M{+}p(ip_j)^S)/2$

Next, we give an example of justifying a 0 to inputs of a 3 input AND gate with inputs A, B, and C. SCOAP controllability (0) values for inputs A, B, and C being (2, 4, 8). The probability of a selection written as (p(A), p(B), p(C)) using SCOAP based decision is (1, 0, 0). Random decisions lead to the probability of the selection as (1/3, 1/3, 1/3). Probabilities calculated using S-Measures, M-Measures and guidance mechanisms used in this work are (0.57, 0.29, 0.14), (0.88, 0.11, 0.01), and (0.72, 0.20, 0.08), respectively. Guidance by S Measures lead to a weighted random decision where the least favorable input C with SCOAP value 8 has a 14% chance of being selected. Decisions by M-measures are more focused toward the selection of the easiest to justify input. The probability of the selection of inputs with a higher SCOAP value is considerably reduced in comparison to other inputs. However, the average of probabilities obtained by S and M measures result in creating a balance between the number of specifications and the facilitation of generations of tests for undetected faults.

3.2.6 Propagation Decisions

Propagation decisions are required when we have multiple gates on the D-frontier and one of them is selected to propagate a D or $\overline{D}$ to the circuit outputs by implying a non-controlling value to all other unspecified inputs of the selected D-frontier gate.

As in the case of justification decision, we calculate the probability of the selection of a given D-frontier gate using S and M measure and calculate a final mixed probability for the selection of gates on the D-frontier to balance between facilitation of detection of yet undetected faults and the number of specifications required for propagation.

If $g_0, ..., g_m$ are gates on the D-frontier the probability of the selection of gate g_j is calculated for both S and M ATPG guidance measures.

The probability of the selection of gate g_j using M measures is given by:

$p(g_j)^M = MP(g_j)/\sum_i^{i=1...m} MP(g_i)$.

Similarly, the probability of the selection of gate g_j using S measures is given by:

$p(g_j)^S = SP(g_j)/\sum_i^{i=1...m} SP(g_i)$.

The final probability actually used for the selection of gate g_j on D-frontier is the average of the two probabilities $p(g_j^S)$ and $p(g_j^M)$:

$p(g_j)^{mixed}{=}(p(g_j)^M{+}p(g_j)^S)/2$

3.3 Transition Fault ATPG

Test generation of transition faults requires initialization of the fault site in the first time frame, and activating a transition at the fault site and propagating the fault effect to PO/PPO's for observation in the second time frame. The generation of tests for a transition fault slow-to-rise(fall) start with justifying a value 1(0) at the fault site. A value 1(0) is added to the J-frontier for slow-to-fall(rise) faults for the first frame for initialization of the fault. Next, the value on the line is justified by assigning appropriate values to the input(s) of the gate driving the faulty line.

Next, in the second time frame depending on the transition fault type a transition needs to be activated and propagated. In other words for the second time frame-for slow-to-fall(rise) fault a 0(1) needs to be assigned to the fault site and the fault effect(transition) needs to be propagated. During the second time frame the problem translates to the detection of a stuck-at-1(0) fault in the second time frame. In other words in the second time frame, a for slow-to-fall (rise) fault, a 0(1) value needs to be justified at the fault site (added to the J-frontier) and a $\bar{D}(D)$ needs to be added to D-frontier to propagate the fault effect. During the test generation, any values in the second time frame are satisfied from PPIs/PIs in the first time frame. Primary inputs can not be specified in the second time frame and primary outputs can not be observed in the first time frame, when launch-on-capture testing schemes are used.

The entries in the D-frontier are gates for which some input(s) have D or $\bar{D}$ values and the outputs are unspecified. Justifying the required value on a line in the J-frontier is done by assigning the proper value to one of the inputs of the gate driving in the frame under consideration. All propagation decisions take place in the second (propagation) time frame. Propagating D or $\bar{D}$ to an observed output in the propagation frame is done by selecting a gate from the D-frontier and propagating the transition effect, D or $\bar{D}$, to its output. The selection of a proper input to be assigned to justify a line value in the J-frontier during initialization and propagation frame and selecting a proper gate from the D-frontier to propagate D or $\bar{D}$ in the propagation frame, is important to obtain tests with a minimum number of specified values, to obtain minimal size test sets, and to reduce test generation times.

The ordering of faults also has an impact on the overall test length. One would think that targeting faults that help in the generation of tests for other undetected faults first is a good idea. The ordering should help with the following:

1. *Fortutious Detection:* Faults should be targeted in an order to increase fortuitous detection. Tests can be generated for large numbers of other, yet to be

detected faults while generating a test for originally targeted fault.

2. *Hard to Detect Faults:* A few faults are hard to detect; typically a test generated for a hard-to-detect fault does not result in fortuitous detections. A hard-to-detect fault, if targeted in the end, results in low fortuitous detections and typically increases pattern count.

A balance needs to be created to target faults in such a way that large numbers of faults are detected fortuitously, while keeping pattern counts in reasonable limits.

In the next sub-section, we propose a new fault ordering mechanism that balances between targeting faults with high fortuitous detection and hard to detect faults.

3.3.1 Fault Ordering Based on Test Enumeration

We developed a new fault ordering mechanism; before that we propose two new measures: Test Enumeration Measures for Propagation (TEP) and Test Enumeration Measure for Justification (TEJ) to help classify faults in to one, having a high fortuitous detection or hard-to-detect fault.

Consider an example of a three input AND gate with inputs A, B, C, output D, and stuck at fault model. It is important to note that all stuck-at-1 or stuck-at-0 faults on A, B, and C, respectively, can be initialized at the same time.

However, for launching the final transition value at the fault site and propagating the fault effect, it is different. All slow-to-rise faults can be tested simultaneously, as each of them require the same inputs (1, 1, 1) on (A,B,C), respectively. However, for slow-to-fall faults on (A, B, C), we require three vectors (0,1, 1), (1,0,1), and (0,1,1), respectively.

Note that initialization and final launching of the transition during the propagation phase are comparatively different as, during initialization phase, the fault location only needs to initialized.

All the values in the propagation frame are controlled by initialization frames it is much easier to justify a value in the initialization frame; hence, we will consider only propagation frame.

We use two test enumeration measures, that estimates number of faults whose detection is facilitated by a decisions, these measures are named Test Enumeration Measures (TE measures).

3.3.1.1 TE Measures for Justification

Any justification decision in the propagation frame should facilitate placement of the final transition on other fault sites; we estimate this by TEJ measures- each line l is assigned two TEJ measures for both 0 and 1, respectively.

ATPG is guided more often toward options which facilitate the placement of final transition value 1 for slow-to-rise and 0 for slow-to-fall transition faults.

For example, for a 3 input AND gate, TEJ0 will have a value of 3 and TEJ1 will have a value of 1 as slow-to-rise faults on inputs need 1 pattern to test and to create a transition (0s can be initialized simultaneously), but three patterns to test slow-to-fall faults as each slow-to-fall faults on input need a separate pattern. The

```
Require: Set $TEJ^1$ $TEJ^0$ to 0 for all lines
  for Forward trace from PI/PPI towards PPO in a levelized manner do
    if If gate is scan cell then
      $TEJ^{i^0} = TEJ^{i^0} = 1$
    end if
    if If gate is primary input then
      $TEJ^{i^0} = TEJ^{i^0} = 0$
    end if
    if If gate is AND(NAND) then
      Output $TEJ^{0(1)}$ is sum of input $TEJ^0$'s
      $TEJ^{output0(1forNAND)} = \sum TEJ^0$all inputs
      Output $TEJ^{1(0)}$ is equal to maximum among input $TEJ^0$'s
      $TEJ^{output1(0forNAND)} =$ max $TEJ^1$ among inputs
    end if

    if If gate is OR(NOR) then
      Output $TEJ^{1(0)}$ is sum of input $TEJ^1$'s
      $TEJ^{output1(0forNAND)} = \sum TEJ^1$all inputs
      Output $TEJ^{0(1)}$ is equal to maximum among input $TEJ^1$'s
      $TEJ^{output0(1forNAND)} =$ max $TEJ^0$ among inputs
    end if

    if If gate is INV(BUF) then
      BUF Output $TEJ^{0(1)}$ is equal input $TEJ^{0(1)}$
      INV Output $TEJ^{1(0)}$ is equal input $TEJ^{0(1)}$
    end if
    Similarly TEJ values can be calculated for other gates- TEJ value at output that
    can be satisfied by controlling values at inputs, corresponding TIJ's at controlling
    inputs are added

    In case TEJ value at output satisfied by non-controlling values at inputs we take
    maximum among inputs taking care of polarity

    if Stem then
      $TEJ^0 = 1$, $TEJ^1 = 1$ respectively for stem gate.
    end if
  end for
```

Figure 3.11: Calculation of TEJ^1 and TEJ^0.

```
Require: Set $TEP^0$ $TEP^1$ to 0 for all lines
  for Trace in a leavelized manner starting from PO/PPO towards PI/PPI do
    if gate i is a PPO/PO then
      $TEP^0$=0, $TEP^1 = 0$
    else
      for input's j of gate i do
        for $v \in \{0,1\}$ do
          if value $v$ at j uniquely determines value for i then
            if $v$ is 1 $TEP^{j1} = TEP^{i1} + TEJ^{j1}$
            if $v$ is 0 $TEP^{j0} = TEP^{j1} + TEJ^{j1}$
          else
            Store all inputs other than $j$ required to propagate value at input $j$ to
            output $i$ in an array $L$
            for every line l in array L do
              if $v = 1$ $TEP^{j1} = TEP^{j1} + TEJ_{l1} + TEJ_{l1}$
              else $TEP^{j0} = TEP^{j0} + TEJ_{l1} + TEJ_{l0}$
            end for
          end if
        end for
        If i is inverting gate $TEP^{j0} = TEP^{i1} + TEJ^{j0}TEP^{j1} = TEP^{i0} + TEJ^{j1}$
        Else $TEP^{j1} = TEP^{i1} + TEJ^{j1}$ ,$TEP^{j0} = TEP^{i0} + TEJ^{j0}$
      end for
    end if
    If stem add all values of for branches
  end for
```

Figure 3.12: Calculation of TEP^1 and TEP^0.

procedure to calculate TEJ measures is illustrated in Figure 3.11.

3.3.1.2 TE Measures for Propagation

Test enumeration measures for propagation (TEP) measures are of two types: TEP measures to estimate the number of times a justification is required at a line during the propagation frame, and TEP measures for propagation estimates of the facilitation of propagation of transition to observed PO/PPO.

On the lines of SCOAP observability measures, we calculate TE measures for propagation (TEP). We use TEJ values to calculate TEP measures; however, TEP values are dependent on the polarity of fault-free values after transition, currently being propagated. Hence, two TEP values are assigned to each line corresponding to 1 and 0, respectively. Figure 3.12 shows the procedure to calculate TEP measures.

3.3.1.3 Generation of Final Fault List

Total Test Enumeration Measures (TET) are used to get the total of fortuitous detection capabilities of a fault. TET combines estimates of fortuitous detection of a fault during propagation and justification and hard-to-detect faults that would require a separate pattern. The procedure to generate the final fault list targeted by ATPG is given in Figure 3.13.

3.3.2 ATPG Decisions

ATPG decisions are made in a similar fashion, as the previously- discussed ATPG technique [5]. If the gate is marked, static measure is used for making ATPG decisions, or else appropriate test enumeration measures are used instead of dynamic measures, keeping the time-frame into consideration.

We use a propagation-first ATPG approach to generate tests. The advantages of using a propagation-first ATPG approach for transition faults are:

1. While generating compact tests for circuits with compression the number of specified positions is important. Most of the specified positions are resultant of

```
**Require:** Set $TEJ^1$ $TEJ^0$ to 0 for all lines
  We define a new value, Test Enumeration Measures Total(TET), which is measure of fortutious detection possible when a test
  fora given fault is generated
  For every line $i$ we have two total test enumeration measures $TET_i1(0)$ which is sum of test enumeration measure for propogation $TEP_i1(0)$ multiplied by test enumeration measure for justification $TEJ_i1(0)$, respectively and store it in a list $List_A$
  Sort $LIST_A$ in descending order according to Total Test Enumeration Measures
  //Detection of hard to detect fault and fortutious detection is balanced in next step.
  Make a final list of faults $List_f$, by selecting a fault from beginning of $List_A$ and moving it from $LIST_A$ to $LIST_f$ and then a fault from the end of $List_A$ and moving it from $List_A$ to $LIST_f$.
  $LIST_f$ is the final fault order targeted for test generation by ATPG
```

Figure 3.13: Generation of Final Fault Order

the creation of a propagation path from fault location to the PO/PPO.

2. Only the initialization time-frame is directly controllable from the inputs. Propagation time frame is controlled by tracing all the values from the propagation frame to the initialization frame PO/PPO. Hence, the impact of a propagation decision is more pronounced.

3. In case of stuck-at faults propagation, first approach is known to work better than justification-first approach.

3.3.2.1 Propagation Decisions

Propagation decisions are made in propagation frame during test generation for transition faults. Initialization of a fault is comparatively easier than creating transitions at the fault location and propagating it to the PO/PPO. Fault detection for transition faults can be sub-divided into two parts:

1. Initialization of fault location (line l) to 0 (1) for slow to rise(fall) fault in initialization frame.
2. Detection of a stuck-at-0(1) fault on line l (fault location) in propagation time frame.

Since any line l is controllable only from initialization frame, the second step of detection of stuck-at faults in the propagation frame is comparatively difficult to achieve and results in a higher number of specified scan-cells (during the initialization frame). Therefore, to generate compact tests it is important to make prorogation decisions in a manner that facilitates the detection of additional stuck-at-0(1) (for slow-to-rise(fall)) faults in the propagation frame to enhance fortuitous detection. Simultaneously, ATPG decisions should try to create a balance between fortuitous detection and the number of specified positions.

Dynamic and static guidance measures proposed in [5] using a probabilistic marking scheme and creates a balance between fortuitous detection and the number of specified positions. We use dynamic and static ATPG guidance measures described in [5] to make propagation decisions as:

1. Dynamic measures for propagation facilitates the second step in transition fault tests, the detection of a stuck-at fault in propagation frame. ATPG decisions are made in a manner that the second step is facilitated for a large number of faults.

Table 3.5: Comparison of the Proposed and Random Decision

	Random Decision ATPG		Method1 ATPG		Method1 + Flt. Ordering	
CKT	FC	TL	TL	%Red	TL	%Red
D1	96.71	15040	11548	23.2	11016	26.8
D2	93.31	5696	4513	20.8	4385	23.0
D3	94.56	20032	10280	48.7	9672	51.7
D4	95.31	4992	4277	14.3	4232	15.2
D5	99.26	7938	6671	16.0	6389	19.5
D6	98.63	2048	1699	17.0	1706	16.7
D7	91	2496	1920	23.1	1893	24.2
D8	97.04	8064	6327	21.5	5918	26.6
D9	95.94	9359	7738	17.3	7626	18.5
Avg.		75665	54974	27.3	52837	30.2

Table 3.6: Effect of Compression Ratio

Circuit	Compression Ratio	Proposed ATPG		Random Decision ATPG		%Red
Size		FC	TL	FC	TL	TL
D1 1.2M	50	96.08	3645	96.08	3883	6
	100	96.08	6035	96.08	7151	16
D2 3.2M	50	93.35	3146	93.35	3685	15
	100	93.35	5181	93.35	6788	24

2. Step1 of activating the transition fault is much easier, as all values need to be justified to initialization-frame scan cells.

3.3.2.2 Justification Decisions

Justification decisions are made in initialization and propagation frames. In propagation frames, we make justifications decisions based on dynamic guidance measures; this creates a balance between fortuitous detection and the number of specified scan-cells. We analyze the observability of a fault being propagated forward. If the fan-in of the gate on the D-frontier has a fault-free value which is also the controlling value for the gate on the D-frontier, no fault whose D-frontier reaches the off-path inputs can be detected. D-frontier for all such faults is killed by the presence of the controlling value. Hence static measures, which result in the reduction of the number of specified positions, are used for ATPG guidance.

3.4 Experimental Results

We first present the experimental results for fault-based ATPG guidance approach presented in [5], referred to as *Method1.* In the next sub-section, we will present the results for ATPG for BIST-ready designs presented in [6]. In Section 3.4.1 and 3.4.2 we consider stuck-at faults and in Section 3.4.3 we consider transition faults.

3.4.1 Fault-Based ATPG

We incorporated the proposed methods of Method1 to guide line justification and D-propagation decisions into a state-of-the-art commercial ATPG. The ATPG is D-algorithm [9] based. It uses dynamic compaction [29], where unspecified values in test cubes are used to detect additional faults called secondary target faults. In the current use of the base line, ATPG line justification and D-propagation decisions are made randomly, as it is known to yield smaller test sets than using SCOAP-based controllability and observability measures for these decisions.

In Table 3.5, we compare the results of test generation using random decisions and the Method-1 ATPG for line justification and D-propagation. After the circuit name, we give the fault coverage obtained and test lengths using random decisions, followed by the values when the Method-1 ATPG is given. In the last column, we give the percentage reduction in test length obtained by Method-1. It can be seen that,

Table 3.7: Results when p is 1/2 for all gates

	Random ATPG		Gates marked with p = 1/2		
Circuit	FC	TL	FC	TL	% Red TL
D1	96.71	15040	96.71	13121	12.8
D2	93.31	5696	93.31	4954	13.0
D3	94.56	20032	94.56	14881	25.7
D4	95.31	4992	95.31	4632	7.2
D5	99.26	7938	99.26	7183	9.5
D6	98.63	2048	98.63	1869	8.7
D7	91	2496	91	2218	11.2
D8	97.04	8064	97.04	7047	12.6
D9	95.94	9359	95.94	8656	7.5
Avg.		75665		64561	14.7

Table 3.8: Results using only static guidance measures

	Random ATPG		Proposed Static Measures		
Circuit	FC	TL	FC	TL	% Red TL
D1	96.71	15040	96.71	14325	4.8
D2	93.31	5696	93.31	5524	3.0
D3	94.56	20032	94.56	16884	15.7
D4	95.31	4992	95.31	5231	-4.8
D5	99.26	7938	99.26	7659	3.5
D6	98.63	2048	98.63	1972	3.7
D7	91	2496	91	2367	5.2
D8	97.04	8064	97.04	7369	8.6
D9	95.94	9359	95.94	8937	4.5
Avg.		75665		70268	7.1

on average, the Method-1 ATPG reduces test lengths by about 21%. In contrast, earlier methods using modified line justification and D propagation decisions [36, 39] achieved, on average a test set size reduction of 6% and 5%, respectively.

In the next experiment, we generated tests for circuits D1 and D2 when the designed compression ratios were set to 100X and 200X, and the results are reported in Table 3.6, similar to Table 3.5. It can be noted that the percentage reduction in test lengths using the Method-1 ATPG improves as the compression ratio is increased.

Finally, we performed three additional experiments to elicit the contribution

Table 3.9: Results using only dynamic measures

Circuit	Random ATPG		Proposed Dynamic Measures		
	FC	TL	FC	TL	% Red TL
D1	96.71	15040	96.71	13573	9.8
D2	93.31	5696	93.31	4727	17.0
D3	94.56	20032	94.56	17485	12.7
D4	95.31	4992	95.31	4432	11.2
D5	99.26	7938	99.26	6865	13.5
D6	98.63	2048	98.63	1828	10.7
D7	91	2496	91	2118	15.2
D8	97.04	8064	97.04	7127	11.6
D9	95.94	9359	95.94	8094	13.5
Avg.		75665		66249	12.4

of each of the three modifications to ATPG decisions we introduced, viz., marking of gates, static measures, and dynamic measures. The results of these experiments are reported in Tables 3.7, 3.7, and 3.8 using the same format as in Table 3.5. In Table 3.7, we give the results of test generations when all the gates in a design are marked with probability 1/2 and the Method-1 ATPG static and dynamic measures are used. In this case we note that, on average, test set sizes are reduced by 12%, relative to the base line ATPG results. Even though randomly marking gates lead to smaller test sets than with the baseline ATPG, the test set size reduction is lower than when the gates are marked with the probabilities used in the Method-1 ATPG. In the next experiment, we did not use the marking of gates and used only static measure, given in this work, to guide line justification and D propagation. The results of this experiment are given in Table 3.8, from which we note that using only static measures and no marking yields, on average, a reduction of 6.1% in test set sizes, relative to the baseline ATPG. In the final experiment, we again did not use the marking of gates and used only the Method-1 dynamic measures to guide line justification and

Table 3.10: Comparison of fault ordering methods

	Random Decision ATPG		Method2 Ordering		New Flt Ordering	
CKT	FC	TL	TL	%Red	TL	%Red
D1	96.71	15040	14547	3.3	14239	5.6
D2	93.31	5696	5595	1.8	5490	3.8
D3	94.56	20032	19768	1.3	18706	7.1
D4	95.31	4992	4850	2.9	4706	6.1
D5	99.26	7938	7564	4.7	7427	6.9
D6	98.63	2048	1933	5.6	1973	3.8
D7	91	2496	2418	3.1	2380	4.9
D8	97.04	8064	7868	2.4	7728	4.3
D9	95.94	9359	9028	3.5	8834	5.9
Avg.		75665	73570	2.8	71484	5.8

D propagation decisions. Results are reported in Table 3.9, from which we observe that using dynamic measures without marking gates yields, on average, a test set size reduction of 12%.

In Table 3.10 we compare FFR height based fault ordering methods to newly proposed fault ordering based on the ease of detection in Section 3.4.3. The new fault-ordering method results in a 5.8% reduction in test length, in comparison to 2.8% by the FFR height-based ordering procedure described in [6]. From the four experiments discussed above, we conclude that using all four together maximally reduces the test set sizes.

3.4.2 Experimental Results for BIST Ready ATPG

We used a commercial tool to make the designs BIST ready. The tool inserted logic for X-bounding and inserted test points (control and observation), connected to new scan cells that do not control functional logic. The maximum number of test points allowed in a design was 0.5% of the number of gates in the design. We also used a commercial tool to add test data compression logic using a sequential

Table 3.11: Results for non-BIST ready designs

	Random Decision ATPG		Method1 + New Ordering		Method2	
CKT	FC	TL	TL	%Red	TL	%Red
D1	96.71	15040	11016	26.8	12212	18.8
D2	93.31	5696	4385	23.0	4437	22.1
D3	94.56	20032	9672	51.7	11667	41.8
D4	95.31	4992	4232	15.2	4033	19.2
D5	99.26	7938	6389	19.5	6933	12.7
D6	98.63	2048	1706	16.7	1802	12.0
D7	91	2496	1893	24.2	1914	23.3
D8	97.04	8064	5918	26.6	6273	22.2
D9	95.94	9359	7626	18.5	7624	18.5
Avg.		75665	52837	30.2	56893	24.8

decompressor and test response compactor to the BIST ready designs, as well as the original designs.

In Table 3.11, we compare the results of test generation using random decisions, the test length obtained by Method-1 ATPG, with new fault ordering and Method-2 ATPG [6]. We also give the percentage reduction in test length in comparison to the test length obtained by a random decision ATPG. It can be seen that, on average, the Method-1 ATPG combined by the new fault ordering methodology reduces test lengths by about 30%. In contrast, line justification and D propagation decisions using Method-2 [6] achieve, on average a test set size reduction of 24.8%.

Next, in Figure 3.14 we compare test lengths obtained for BIST ready designs using various line justification, fault propagation, and fault ordering methodologies. Test lengths are normalized in comparison to random decision ATPG, which is assigned a value of 100. On average Method-1 combined with the new fault ordering procedure resulted in a 30.3% reduction in test length. Method-2, results in a lower 28% reduction in test length.

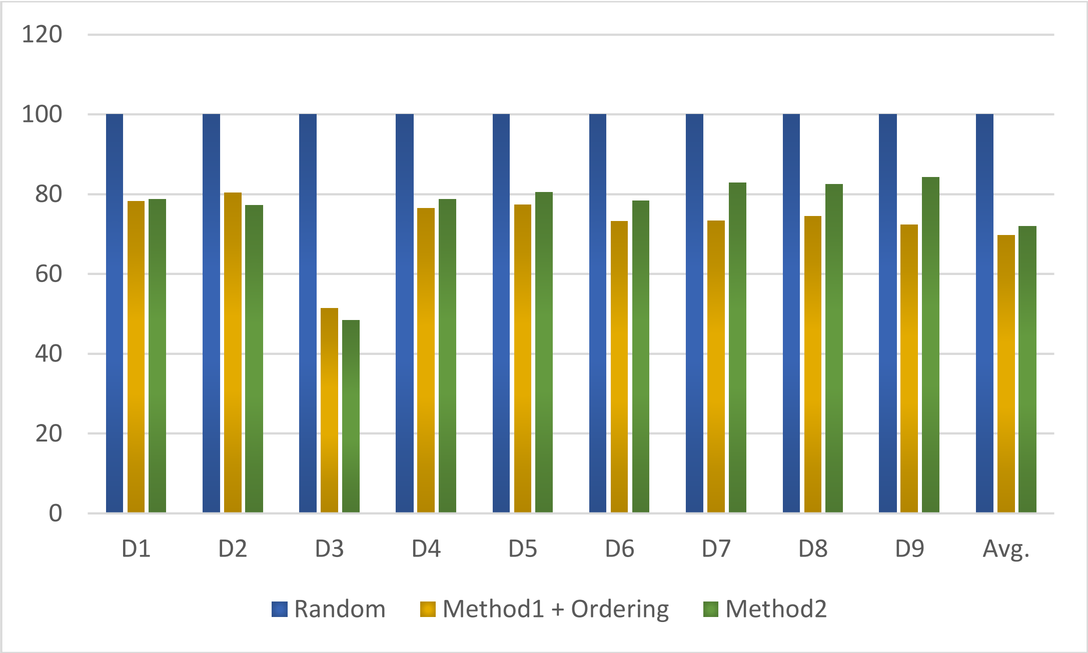


Figure 3.14: Test length comparison for BIST ready designs

In practice, Method-2 is faster than all other techniques as shown in Figure 3.15 for non BIST ready designs. Run-time for the default ATPG is assigned a value of 100; other run times are normalized in comparison with the default ATPG accordingly. Method1 + new fault ordering increases run-time by 12%; Method-2 decreases run time by 11% on average.

3.4.3 Results for Transition Faults

In this section, we discuss the results for the transition fault (launch-on-capture[2]) ATPG discussed in this chapter. In Table 3.12, we compare the results of test generation for transition fault model using random decisions, with proposed

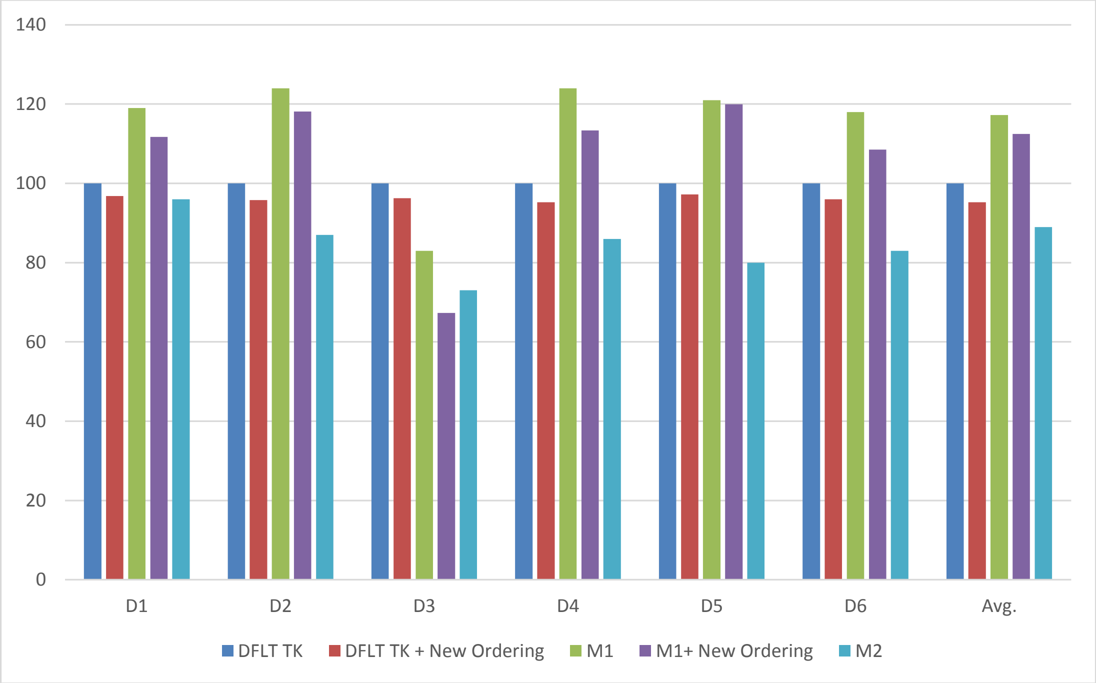


Figure 3.15: Runtime comparison for non BIST ready designs

Table 3.12: Results for transition fault ATPG

Ckt	FC	RandomTL	Proposed Trans	%Red	[5] TL	%Red	[6] TL	%Red
D1	94.43	25277	19053	24.6	21822	13.7	21288	15.8
D2	93.63	20839	15804	24.2	17759	14.8	17411	16.5
D3	93.14	26535	11850	55.3	17276	34.9	15950	39.9
D4	57.92	2478	1562	37.0	1933	22.0	1996	19.4
D5	96.54	21213	13852	34.7	17842	15.9	16663	21.5
D6	88.15	19313	14682	24.0	16901	12.5	15839	18.0
D7	89.29	7052	5074	28.0	5677	19.5	6024	14.6
D8	93.46	13052	9680	25.8	10829	17.0	10234	21.6
D9	94.6	20628	14834	28.1	17749	14.0	15940	22.7
Avg.		156387	106391	32.0	127787	18.3	121346	22.4

Table 3.13: Results using only dynamic measures

Ckt	FC	RandomTL	Dynamic Measures TL	%Red
D1	94.43	25277	22630	10.5
D2	93.63	20839	18884	9.4
D3	93.14	26535	22742	14.3
D4	57.92	2478	2198	11.3
D5	96.54	21213	19030	10.3
D6	88.15	19313	17731	8.2
D7	89.29	7052	6347	10.0
D8	93.46	13052	11116	14.8
D9	94.6	20628	17212	16.6
Avg.		156387	137891	11.8

Table 3.14: Results with only static measures

Ckt	FC	RandomTL	Static Measures TL	%Red
D1	94.43	25277	21619	14.5
D2	93.63	20839	17426	16.4
D3	93.14	26535	18762	29.3
D4	57.92	2478	1950	21.3
D5	96.54	21213	18394	13.3
D6	88.15	19313	16959	12.2
D7	89.29	7052	5783	18.0
D8	93.46	13052	10985	15.8
D9	94.6	20628	16593	19.6
Avg.		156387	128471	17.9

Table 3.15: Results with only fault ordering

Ckt	FC	RandomTL	New Flt Ordering TL	%Red
D1	94.43	25277	24185	4.3
D2	93.63	20839	20018	3.9
D3	93.14	26535	24346	8.3
D4	57.92	2478	2402	3.1
D5	96.54	21213	20127	5.1
D6	88.15	19313	18378	4.8
D7	89.29	7052	6722	4.7
D8	93.46	13052	12535	4.0
D9	94.6	20628	19692	4.5
Avg.		156387	148405	5.1

transition fault ATPG, the Method-1 ATPG, and Method-2 ATPG for line justification and D-propagation. After the circuit name, we give the fault coverage obtained and test lengths using random decisions, followed by the values when the proposed transition fault ATPG is given. In the next column, we give the percentage reduction in test length obtained by the new transition fault ATPG. Test lengths obtained by Method-1 ATPG, followed by percentage reduction and Method-2 ATPG are also given. New transition fault ATPG reduces test length by 32% in comparison to random decision ATPG; only 18% and 22% reduction is achieved by Method-1 and Method-2 ATPG.

Later we study individual components of the proposed transition fault ATPG and their impact on pattern count reduction; we individually study the impact of dynamic and static measures as well as fault ordering on the final pattern count. In Table 3.13 we show that pattern count reduction reduces to 12% if only dynamic measure was used for ATPG guidance; no fault ordering was used. Pattern count reduces by 18% if only static measure was used for ATPG guidance as shown in Table 3.14. Impact of fault ordering is given in Table 3.15; justification and propagation decisions were made randomly in this case. Fault ordering reduces pattern count by 5.1%.

CHAPTER 4
ISOMETRIC TEST COMPRESSION

Test compression schemes are commonly characterized by their compression ratios which can reach, at their best, the value of f^{-1}, where f is the number of specified positions, on average, that need to be deterministically determined in a scan cycle or fill rate. In this chapter, we demonstrate that on-chip test data compression can be taken to a new level by synergistically engineering both test generation (ATPG) and encoding of tests. We propose a new test compression scheme that has the following features:

- Elevates compression ratios to values beyond what is achievable through the conventional reseeding.
- Reduces the switching activity during scan shift-in.
- Provides a programmable option to control test power, without hardware modifications.
- Low hardware overhead, in comparison to previous low power compression schemes.

The new scheme deploys an on-chip power-aware test data decompressor, the corresponding test cube encoding method, and a compression-constrained ATPG that allows loading scan-chains with patterns having low toggling (count of change in the content of scan cells), while encoding a significant number of specified bits produced by an ATPG in a compression-friendly manner. As a result, the new solution offers

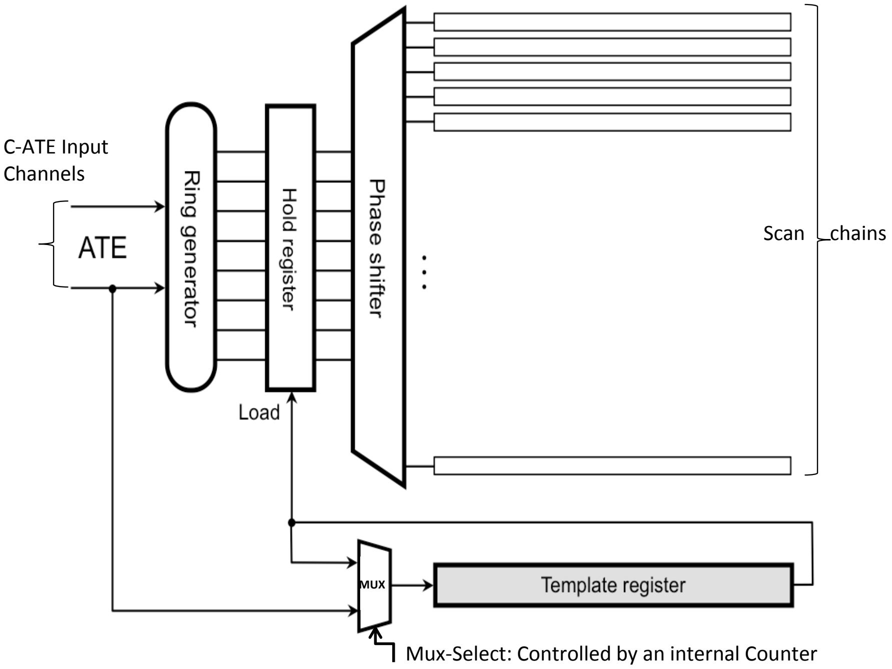


Figure 4.1: Isometric Decompressor

very high compression ratios. Moreover, the new scheme avoids periods of elevated toggling in scan-chains and reduces scan unload switching activity due to unique test stimuli produced by the new technique, leading to a significantly reduced shift toggle rates for the entire circuit under test.

4.1 Isometric Architecture

Figure 4.1 recalls a generic architecture of our on-chip test data decompressor originally presented in [60] called EDT, which we modify in this work. It consists of a ring generator and a phase shifter driving the scan chains. Compressed test patterns are delivered to the decompressor through c external channels in such a way that

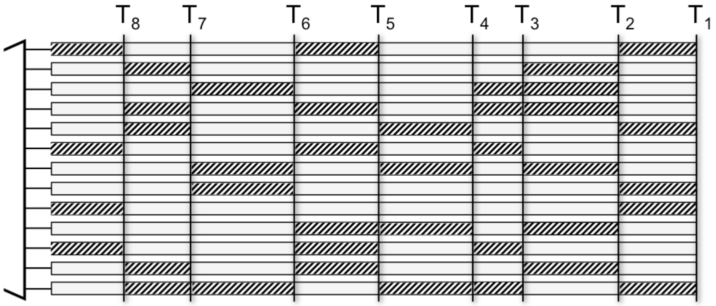


Figure 4.2: Low power template

a new C-bit vector is injected into the ring generator every scan shift cycle. The hold register in the EDT placed between the ring generator and the phase shifter is reused. It captures and saves certain states of the ring generator. As a result, the toggling-free data can be provided to the scan-chains for a number of continuous shift cycles by taking the shift data from the hold register while the ring generator keeps advancing to the next states needed to encode another group of specified bits in future scan cycles.

4.1.1 Overview of Isometric Compression Scheme

In this work, we demonstrate that the proposed architecture opens a new direction in test data compression when used in conjunction with a compression-constrained ATPG. An additional circular (or template) register is added, which facilitates operations of the decompressor as illustrated in Figure 4.1.

The template register size matches the longest scan-chain. Initially, a tester, using a single channel, uploads a control pattern to the circular register. This is done only once for a number of patterns. Next, from the template register, data is applied to the hold register to control whether it is updated from the ring generator or not, while the ATE channels upload successive compressed test patterns in the scan-shift mode. As a result, a control bit is delivered to the hold register every shift cycle in order to indicate whether the hold register is to be updated with the current content of the ring generator. If the hold register is updated, such a time will be referred to as a toggle point. Two successive toggle points determine a hold period.

The scheme of Figure 4.1 allows partitioning a given test cube into several transition-free hold periods, each comprising a certain number of consecutive slices of scan cycles. One can therefore repeat a given decompressor state many times in succession by using the hold register, storing a state that the ring generator entered at the beginning of the hold period. Locations of all toggle points form a test template (see Figure 4.2), where two colors indicate the logic values of 0 and 1 applied to successive scan cells. As a result, the contents of the template register can be used to control the operation of the decompressor. Only the first toggle value in a consecutive sequence of toggles is required to be deterministically determined. The rest of the scan-cell values in the chain are determined by the decompressor hold register till the next toggle. However, new values called free-variables enter the ring generator, and which can be used in later scan cycles.

The actual fill rate (the number of specified bits) does not have to be either a compression-limiting factor or a low switching activity constraint. For example, having the same logic value assigned to several cells hosted by the same scan chain

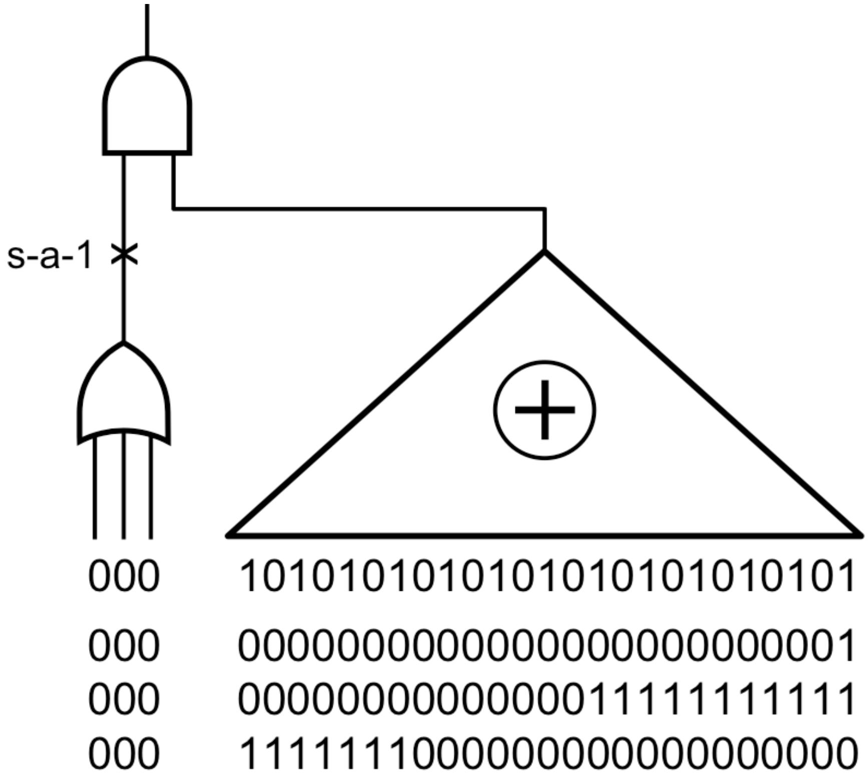


Figure 4.3: Circuit test cubes

may ease problems related to both compression and toggling. Indeed, only specified bits occurring at toggle points must be encoded, whereas bits of the same value make it possible to deliver identical test data to scan chains for a number of shift cycles, thereby reducing the number of transitions in scan cell values.

Consider the example shown in Figure 4.3. Let a circuit under test feature 28 inputs. A part of this circuit is an XOR tree. In order to detect an indicated stuck-at-1 fault, one can apply test patterns listed in the figure. The number of specified bits that have to be encoded within the first vector is equal to $3 + 25 = 28$, whereas to encode one of the remaining patterns it suffices to target only 3 + 2 specified bits. Indeed, the specified values of 0 (1) occur here in sequences not interspersed with 1-bits (0-bits), and thus can be obtained by using the hold functionality since the decompressor outputs can be sustained to deliver the identical test data for a number of shift cycles. Clearly, it is easier to encode just 5 values (and update the hold register) than to handle 28 specified locations.

For the reasons stated above, the scheme presented in this thesis differs from earlier test data compression schemes in a fundamental way:

1. It does not work with test cubes generated prior to the encoding process.
2. Nor does it attempt to determine hold cycles based on locations of the specified bits.
3. Instead, after forming a test template, it employs the template to guide the ATPG as to the most suitable sites of specified bits such that the resultant test patterns are highly compressible, while scan toggling remains within acceptable limits.
4. Because of their high fill rates and special form (many bits of the same value populating the same scan chain), these vectors will be further referred to as *isometric patterns*.

All the free-variables entering the continuous flow decompressor during hold periods can be used during subsequent toggle cycles to encode any peaked in the specified bits that need to be encoded. The above mentioned problem of uneven peaks in specified bits of a test cube, we believe, is usually as a result of bad scan

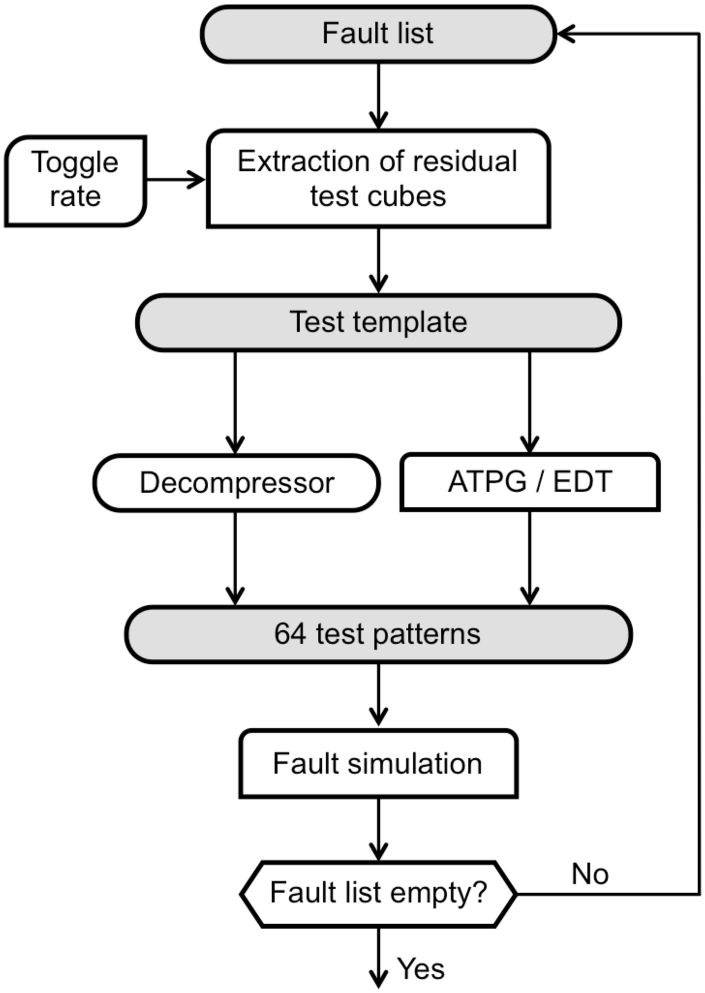


Figure 4.4: Isometric compression flow

chain stitching and the problem can be solved by re-stitching the scan-chains, but most designers would not allow it.

4.2 Isometric Compression Flow

The isometric test compression scheme can be broadly divided into two phases details of which are described below:

- *Phase 1:* Deals with the generation of templates, hold and toggle sequence, required to control the hold register.
- *Phase 2:* Deals with communicating constraints due to test template to EDT solver (mechanism for encoding any test vector). The same constraints are also passed on to the ATPG. Subsequently, K patterns are generated using the same template. In this work, we set K to 64.

In Figure 4.4, we illustrate isometric test compression flow. During template generation process, at first we start by generating test cubes for N faults. Cube generation is followed by storing specified positions that occur in test cubes with very high probability; irrespective of how ATPG decisions were made during the test cube generation, other assignments are not stored in this minimized (called residual cube) cube and are discarded.

Generation of residual cube is followed by the generation of the template. Some important properties of an effective test template are:

- The template should help in controlling the toggling activity during scan shift.
- It should encode a large number of specified scan cells by hold; i.e. without requiring any free variables.
- The template is able to encode cases when a particular scan-slice has a very large number of specified bits by placing the appropriate number of holds before it.
- The template should not lower fortuitous detection of faults.

The first step during the template generation process is the merging of residual cubes. The merging step is very similar to commonly used cube merging procedures during the test compaction procedure described in Chapter 2.

One major difference is that we allow conflicts during the merge. Since a template is used to generate 64 patterns, a conflict based merge allows a scan-cell to be assigned a value of 0 for one vector and 1 for some other vector, hence a test for two faults with conflicting requirements can be generated at toggle points, and this increases the effectiveness of the generated template for test generation.

The actual template is generated using a variant of a set covering problem[45]. Toggles are added incrementally after every residual cube is merged by running a covering algorithm. A toggle locking mechanism of previously generated toggles, ensures that any fault for which a test can be generated can be targeted even after the merging of a few other residual cubes further down in the process.

Post template generation, the template constraints are given to an EDT solver, a mechanism to check if the patterns generated can actually be successfully encoded by the hardware controlled by the current template.

The template hold and toggle constraints are given to the ATPG. Only the toggle cycles are specified by new values from the input. Any group of hold cycles followed by toggle cycles result in copying the value from the immediately previous toggle.

The ATPG communicates these constraints by adding an equivalence relationship between all consecutive hold cycle scan cells in a chain preceded by a toggle. Adding equivalence relations helps the ATPG decision making process and avoids unnecessary backtracks.

4.3 Residual Test Cubes Generation

In this section, we discuss the first step in the isometric compression flow, namely the generation of a pool of residual test cubes for N faults. The vast majority of test data compression schemes work with the concept of a test cube for a fault. Recall that this is a partially specified vector with 0s, 1s, and don't cares. Any test vector contained in such a test cube is a test for the fault. A large body of experimental data shows that test cubes, even those generated with a dynamic compaction targeting multiple faults and performing multiple clock compression, have fill rates anywhere in the range of 0.2% to 5% [28]. Typically, however, locations of only 10% (or less) of the specified bits in a test cube are essential to detect a fault, i.e., these specified values cannot be replaced with specified values in other locations. The remaining 90% of the positions are flexible, i.e., the ATPG may assign specified values to mutually exclusive subsets of sites, and one can take advantage of this phenomenon to further improve test compression.

The scheme presented in this paper uses residual test cubes. Given a fault, the residual test cube is a partially specified vector that retains only those positions of the regular test cube that correspond to primary inputs and scan cells whose likelihood of being specified is greater than a specified threshold. In particular, specified values of all essential scan cells become unconditional parts of a residual test cube. As can be observed, not every vector contained in a residual cube is a test because of not yet specified bits that can be determined in several other ways. However, if a fault is testable, then the ATPG will eventually find a test vector within its residual cube

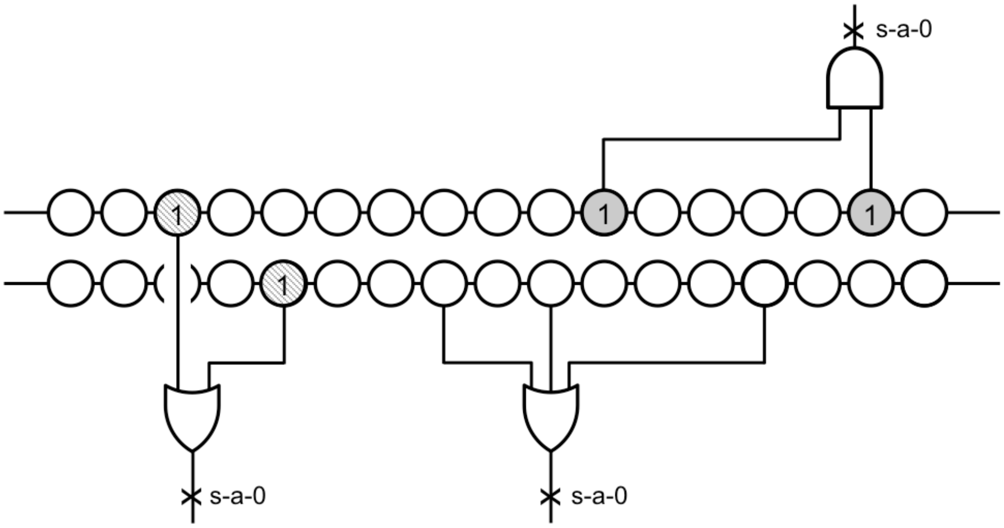


Figure 4.5: Extraction of residual test cube

by replacing some don't care bits with specified values.

4.3.1 Extraction of Residual Test Cube

Example: Consider two scan chains shown in Figure 4.5; in particular, they drive three logic gates with stuck-at-0 faults on their outputs. Clearly, two gray scan cells connected to the AND gate are essential to detect the fault, and hence they are an unconditional part of the residual test cube. Furthermore, setting only one driver of the 2-input OR gate to 1 is necessary to detect another fault. The ATPG can make each assignment with a probability of 0.5. If the threshold is set to 0.5, either of the specified values will make it to the residual test cube. Just as in the previous case the ATPG will likely assign a single logic value of 1 to the 3-input OR gate. This time, however, the probability of setting a given input to 1 turns out to equal 0.33,

Require: *Input:* Fault f and circuit G. *Output:* Probability p1 and p0 for every line in netlist

Procedure is a two pass procedure over the netlist, one forward pass starting from fault location and one backward pass starting from PO/Scan cells.

We use an intermediate queues, forward queue and mechanism to mark gates.

Step 1: Add fault location to forward queue, with polarity opposite to that of fault.

Step 2: Also, faulty gate is marked to be processed during backward trace, magnitude of marking is the polarity opposite to that of a fault.

Step 3: Any forward decision to propagate fault effect to PO/SO is equi-probable. Side-inputs (gate inputs other than by which fault effect is propagating) are marked to be processed during backward trace, with non-controlling value depending on the gate type and magnitude equal to fault effect probability value.

Step 4: In a backward trace, start from PO/SO. In case gate has been marked during forward trace, in case of implication copy magnitude to its inputs (if no inversions else invert values at the input). Inputs are again marked to be processed in a levelized manner.

Step 5: For decision nodes equally divide values at the gates output's to its inputs (taking care of inversions). Inputs are again marked to be processed when reached during backward processing.

Figure 4.6: Calculation of probabilities for a fault

and thus none of these specified bits is included in the residual test cube.

4.3.1.1 Procedure to Calculate Probability

The procedure to calculate probabilities for a fault is described in Figure 4.6. The first step to determine a residual test cube is the computation of the corresponding probabilistic test cube profile. Here, we inject a fault and compute signal probabilities along its propagation paths. Clearly, as long as there is a unique fault propagation path, the corresponding probabilities are set to 1. Probabilities associ-

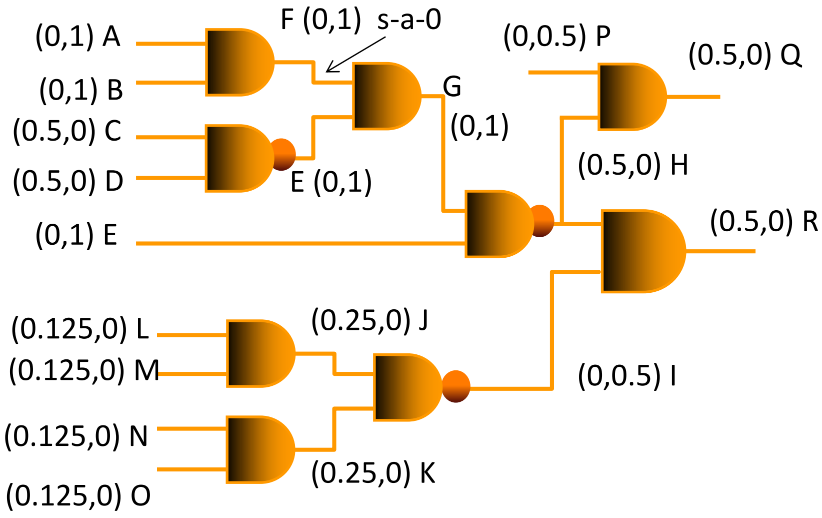


Figure 4.7: Computing probabilities

ated with branches of a fan-out stem are equal fractions (inversely proportional to the number of branches) of the probability that the fault can reach the stem. In a backward implication process, inputs assume the values based on the corresponding outputs. In the off-path implication phase, inputs assume the same non-controlling value as that of the fault propagation. Finally, in the backward justification, the number of inputs divides the output signal probability.

Example: Every line has a probability for 0(1), named p0(p1) corresponding to a fault f. Probabilities p0 is the measure of necessity of using a given value at a line for detection of fault f. In Figure 4.7 line F has a s-a-0 fault f. Detection of f requires justification of 1 on line F. Lines A and B which are input's to AND gate feeding line F need to be set to 1 to justify F as 1. Propagation of fault effect requires line E (off path gate) to be set to non-controlling value with magnitude equal to probability by which fault effect reaches the D-frontier gate E. Line E is set to probabilities, p1=1 (as F is 1) and p0=0. Fault effect reaches gate G with probability equal to input on-path probability, taking inversions into account. Hence, G is justified assigned a probability value of (p0,p1) as (0,1). Justifying E to inputs either one of the inputs C or D needs to be 0, since every decision is equip-probable we divide probability of E into two parts. C and D both are assigned probability of (0.5,0). In a similar manner, probabilities for other lines can be calculated.

From the test cubes and the resultant probabilistic test cube profiles, we obtain the residual test cubes by keeping only specified bits with the corresponding signal probabilities greater than a certain threshold. Once a certain number of residual test cubes become available, a residual cube merging process combines these individual residual cubes generated earlier and yields a multi-fault residual cube. However, this process terminates when the number of toggle points associated with the resultant residual test cube exceeds a user-defined threshold. As mentioned earlier, these toggle points are employed to form a test template whose creation is detailed in Section 4.4.

4.3.2 Typical Residual Cube Profile

Figure 4.8 illustrates an example of a specified bits profile for a large industrial design. The diagram is a cumulative histogram. On x-axis on the graph represents probabilities, p in descending order and on y-axis we show number of inputs with probability greater than or equal to p. The various steps required to generate the

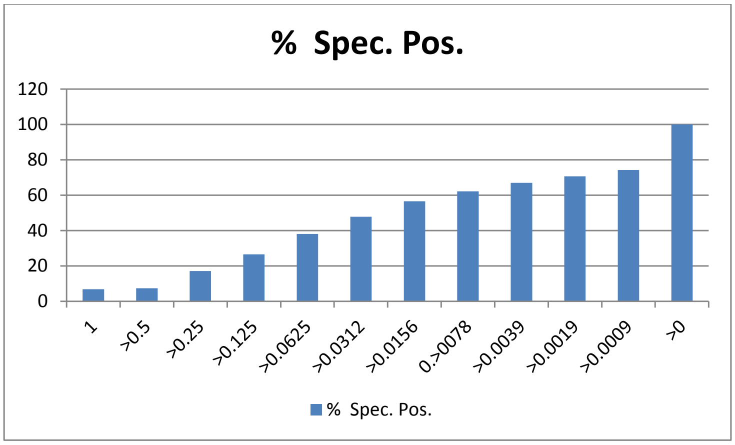
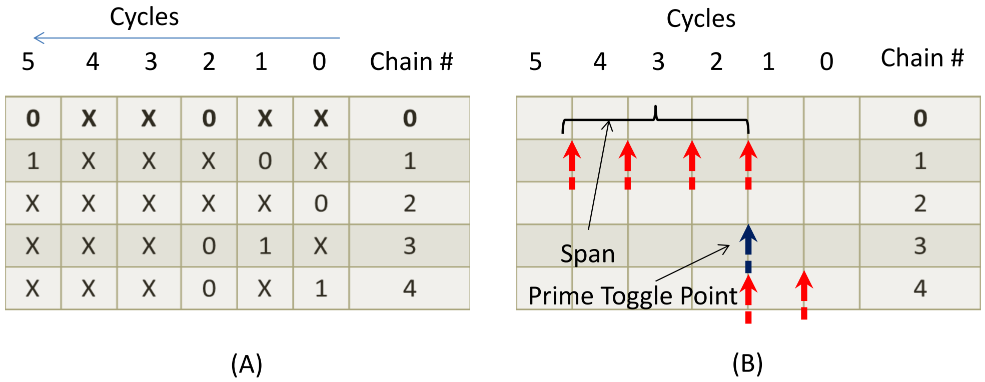


Figure 4.8: Specified bits profile

histogram are:

1. Randomly pick 1K faults. Start with a fault f, the first fault in the list.
2. Generate a test for fault f using an ATPG.
3. Calculate probability values corresponding to fault f, as described in Figure 4.6.
4. Overlap specified scan-cells with the corresponding probability values.
5. Collect data for every specified scan-cell.
6. Generate a graph as shown in Figure 4.8

H(p) is the column height in Figure 4.8, where entry H(p) in the histogram indicates the average percentage of bits that the ATPG may assign a specified value with the probability greater than or equal to p. It is obtained for 1,000 test patterns having 29,473 specified bits. For example, H(0) = 8 means that 8% of all specified bits are essential, whereas H(0.125) = 27 indicates that 27% of positions in total can



5	4	3	2	1	0	Chain #
0	**X**	**X**	**0**	**X**	**X**	**0**
1	X	X	X	0	X	1
X	X	X	X	X	0	2
X	X	X	0	1	X	3
X	X	X	0	X	1	4

(A)

(B)

Figure 4.9: Toggle points and spans

be assigned specified values with the probability greater than or equal to 0.125.

4.4 Test Template Generation

The isometric decompressor of Figure 4.1 offers an inherent ability to encode a test cube in a manner that results in low toggling patterns by selecting toggle points in places where some scan chains change their content values. Recall that the toggle point requires that the hold register be updated from the ring generator. If two adjacent scan cells require different values in their residual test cube, then a toggle point is uniquely determined and is referred to as a prime toggle point. If two successively specified values in a residual cube are separated by scan cells with don't care values, then a toggle point can be placed anywhere between these scan-cells, provided they have different values. Otherwise, there is no need to add any toggle point. A span is a set of locations where a toggle points can be placed between a consecutive 1(0) and 0(1) in a scan chain. For a prime toggle point the span is of length 1, as only one such toggle position exists.

Example: Consider the residual test cube of Figure 4.9. As can be seen, there is one prime toggle point (blue arrows). Other non-prime possible toggle locations are shown in red. However, locations of two other toggle points remain to be determined, as they can be placed anywhere between the values of 0 (1) and 1 (0), respectively. A set of arrows between consecutive 0(1) and 1(0) consists of a *span*.

A span is called a *covered* span if a toggle point is inserted in the test template at any one of the possible toggle locations of the span. In Figure 4.9, the prime toggle point will cover all the spans.

4.4.1 Covering Algorithm

We discuss covering algorithm in the Toggle_gen sub procedure in Figure 4.10. Covering algorithm's aim is to minimize the number of toggles that are being assigned, while generating a test template which successfully detects a large number of faults. Covering algorithm is a variant of a set cover problem. We can only add a definite number of toggles to a test template to comply with the low shift toggle requirements. Prime toggle locations need to be covered to generate a successful test for a fault. Algorithms first add toggles that cover all prime toggle locations, as shown in Figure 4.11; after adding prime toggle location to the test template, we cover another span of length 3 was covered by it. We are left with two spans. The number of spans covered by each possible location is added, possible toggle locations covering the largest number of spans are added to the test template and covered spans are removed.

GENERATE TEMPLATE PROCEDURE

Require: Input: Circuit netlist, a set of residual cubes. **Output:** Possible toggle locations for template.

STEP 1: Initialize merged residual test cube, MR={}, Total toggles=0.

STEP 2: If Total_toggles > allowed switching rate. Calculate toggle positions using sub-procedure Toggle_gen(MR)

STEP 3: If Total_toggles < allowed switching rate. Merge next residual cube by saving(locking) existing toggle locations.

- Overwrite any conflicting PI/PPI values of new residual cube to the corresponding values in MR, with position in MR being overwritten by values in the residual cube currently being merged.
- If both MR and residual cube being merged have same value, do nothing
- If PI/PPI in MR is not specified and residual cube being merged has a specified value, copy value from residual cube being merged to MR.
- If PI/PPI in MR is specified and residual cube being merged is unspecified, do nothing.
- Both residual cube being merged and PI/PPI are unspecified, do nothing

Go back to step 2.

STEP 4: Finally, output toggle positions for the template.

TOGGLE_GEN SUB PROCEDURE

Require: Input: Circuit netlist, merged residual cube MR, locked toggle positions, list of toggles T={} **Output:** A set of toggle positions T

STEP 1: Generate PT spans for MR. Add all locked toggle locations to list of toggles T and remove all covered spans.

STEP 2: Add toggles to cover single length prime toggle point spans and remove other covered spans.

STEP 3: For each cycle i estimate number of spans covered by addition of toggle at i

STEP 4: Add toggle at maximum sum cycle.

STEP 5: If all spans not covered return to step 2.

STEP 6: Finally output toggle position for intermediate processing

Figure 4.10: Procedure to generate test templates

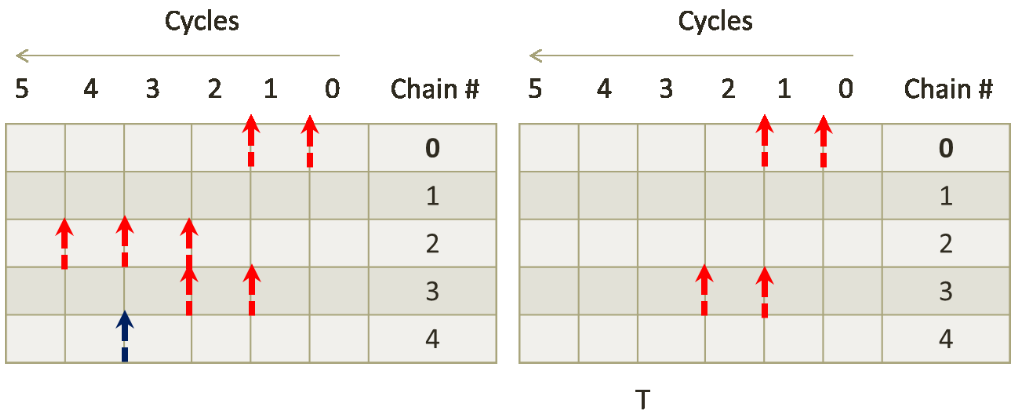


Figure 4.11: Covering Algorithm

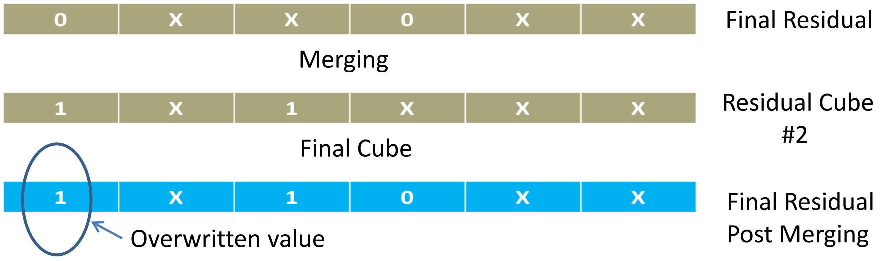


Figure 4.12: Conflict Based Merging

In Figure 4.11, a toggle between cycle 1 and cycle 2 covers the remaining two spans.

4.4.2 Rules for Conflict Based Merging

As discussed previously, conflict based merging is an important concept introduced in this work for the generation of a test template. The rules for merging final merged residual cubes and a new residual cube from the buffer are as follows:

- Overwrite any conflicting PI/PPI values of a new residual cube to the corresponding values in MR, with the position in MR being overwritten by values in the residual cube currently being merged. All the previous toggles before adding the conflicting new residual cube are always added; any previously merged fault still has all toggles needed to generate a test vector to detect that fault.

- If both MR and residual cube being merged have the same value, keep the older value, as it is required for detection of more than one fault.
- If PI/PPI in MR is not specified and the residual cube being merged has a specified value, copy the value from the residual cube being merged to MR as it is required to detect faults corresponding to the residual cube being merged.
- If PI/PPI in MR is specified and the residual cube being merged is unspecified, do nothing.
- If both the residual cube being merged and PI/PPI are unspecified, do nothing.

Figure 4.12 illustrates an example of a conflict based merging.

4.4.3 Assigning ATPG Constraints and Equivalences

In addition to controlling the decompressor, a test template directs the ATPG to produce highly compressible test cubes. Figure 4.7 sketches out the isometric test compression flow. As can be seen, the main loop includes the extraction of residual test cubes for randomly selected faults and forming the corresponding test template. These steps continue until a predetermined number of toggle points are reached. The test template is subsequently passed to the ATPG in order to make it aware of additional constraints that have been recently added. The test template divides scan chains into hold periods. Within each period only one polarity value is used. Hence, all scan cells within a period will have the same value; they all will have exactly the same linear equation, and ATPG will consider all of them equivalent. The ATPG and EDT-based compression iterate until 64 (this is an implementation dependent factor) test patterns compliant with the newly created test template are produced, or a user-defined abort limit is reached. These test patterns are fault simulated, and the fault list is updated accordingly. If there are still faults on the list, the method goes back to pick new faults and to create new residual test cubes. Otherwise, the whole procedure stops.

Having instantiated a test template, one can easily determine all hold periods where the scan cells assume the same value, i.e., they become ATPG equivalent. If the ATPG assigns a certain logic value to one of the scan cells within a given period, it subsequently copies that value to the remaining cells of the same period, and propagates these values back to the circuit. Moreover, the EDT solver assumes that the same equation is associated with these cells. As a result, only the first cell of the period becomes the subject of encoding; this is exactly the only cycle when the hold register can be reloaded with the new content of the ring generator.

As soon as a scan-cell is specified as a result of an ATPG implication or decision, all the equivalent scan-cells are also assigned the same value. Assigning ATPG equivalences help in the following manner:

- Larger number of possible conflicts are immediately detected, and hence it reduces the number of ATPG backtracks.

Shift 10	Shift 9	Shift 8	Shift 7	Shift 6	Shift 5	Shift 4	Shift 3	Shift 2	Shift 1	Scan-Chain
H	H	T	T	H	H	H	H	T	T	H/T
0	0	0	1	1	1	1	1	1	0	Value
Not Enc.	Not Enc.	Enc.	Enc.	Not Enc.	Not Enc.	Not Enc.	Not Enc.	Enc.	Enc.	Enc./Not Enc.

Figure 4.13: EDT solver and the test template

- Number of EDT aborts are reduced, as now large number of additional implications are generated due to equivalence relations, which leads to better performance during dynamic compaction.
- Reduces the validating effort of the EDT solver as equivalence relation does not allow the ATPG to put two opposite values in a set of equivalent cells, which may otherwise result in a backtrack.

4.4.4 EDT Solver and Test Template

EDT solver requires test template, to constraint the solver. When test template is used a toggle hold sequence occurs, only the toggle cycles needs to be encoded (consumes free -variables) not the consecutive holds. An example shown in Figure 4.13 illustrates the concept.

Example: Given a scan-chain of length 10, top most row shows shift cycle number, send row shows -if a toggle or a hold occurs for the particular cycle, third row gives the value assigned to scan-cell and last row shows if the scan-cell is encoded or not (we assume no Xs right now in the above scenario). The above example clearly shows that only toggle cycles are encoded.

4.4.5 Algorithmic Overview

Generation of a template is an important step and centerpiece of isometric compression scheme. The template generated is good for 64 patterns, therefore it needs to be flexible enough to generate a significantly different pattern for each of the 64 patterns. Inefficient templates can lead to pattern count explosion due to reduced fortuitous detection and fault coverage drop.

Our present definition of *template* captures some of the required criteria for a good template. We define template as a set of toggle points, a value is not assigned to any scan-cell or a set of scan-cells. The advantage of the above definition lies in the following fact:

- Template is less constrained; if values were added, all 64 patterns will have the same value for a particular set of scan-cells, resulting in a loss of flexibility and fortuitous detection.
- The ATPG has the flexibility to generate patterns for the current set of undetected faults.
- A test for incompatible faults may be generated, as now the same scan-cell can be specified differently in two patterns.
- Patterns adhere to current switching constraints given by the user.

The procedure to generate template works illustrated in Figure 4.10 works incrementally by picking up a residual cube and merging it to an empty MR cube. Conflict based merging is carried out using the rules described in Section 4.1. After merging, the covering algorithm described in Toggle_Gen_Sub in Figure 4.10 is run to assign toggle points set t_1 for the current specified positions of the MR. Next, another residual cube is merged to MR; if the specified value of any of the scan-cells in MR and second residual cubes conflict, overwrite the value in MR with the value of the scan cell in the second residual cube.

In Figure 4.12, we show an example of a merged cube and merged residual cube post merging of another residual cube. The left most scan-cell has a 0 in an initial merged residual cube. A new residual cube with 1 in the left most position is merged using conflict based merging procedures and 0 is overwritten by 1. Merging with conflicts of residual cubes has its benefits as discussed before. Post merging all the toggles in set t_1 are added to MR. All covered spans are removed, and covering algorithm is run to add additional toggle points to cover spans that are still not covered. Let the new additional toggle set be t_2. Now the new set of locked toggle points are t_1 and t_2; when another residual cube is added all, t_1 and t_2 are first added to remove all covered spans and remaining spans are covered by additional toggles generated using the covering algorithm.

This incremental procedure of merging additional residual test cubes to final MR using the conflict based merging technique continues until we do not overrun our allowed of toggle budget for a given switching activity. Once we have crossed the allowed toggle budget the final test template is given to the EDT solver to change the appropriate equations as well as the ATPG simultaneously to add equivalence relations among scan-cells. Sixty four ATPG patterns are generated using this new template.

4.5 Multiple Template

Handling a large number of scan chains with a single test template may compromise both the quality of compression and a desired toggling rate. It appears that a test data decompressor deploying multiple template registers is proving easier to scale up, and thus to handle pattern counts targeting more aggressive reductions of toggling.

4.5.1 Hardware Architecture

The basic architecture of a revised test data decompressor is shown in Figure 4.14. It is comprised of a few template registers (two in the figure) associated with disjoint groups of scan chains. Due to their segmented structure, a hold register and a phase shifter can drive these scan groups in an independent fashion. Since each hold register is reloaded regardless of other registers, the number of toggling points can be effectively reduced and assigned more flexibly while maintaining compression capabilities of the whole scheme.

4.5.2 Overall Flow

Multiple test template scheme results in a flow slightly different from the usual flow. The first step in the flow consists of grouping scan-chains. Scan-chains are divided into disjoint groups. A well formed group should adhere to the following properties:

- Scan-chains which set up excitation and propagation conditions for the same set of faults should be in the same group.

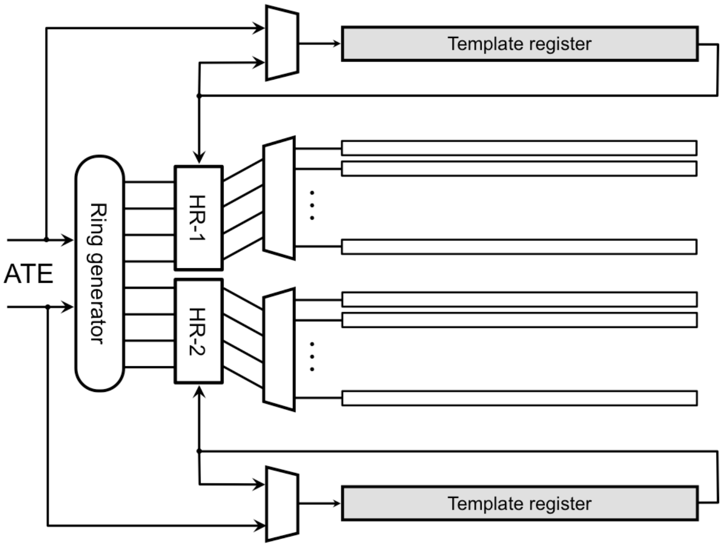


Figure 4.14: Multiple test template usage

- Groups should be selected in a manner that chains in the group, on average, have a low probability of opposite value requirements among scan-cells (of the same scan chain) in the adjacent neighborhood.

Single template flow, described in Figure 4.8, needs to be modified to accommodate multiple test templates for test generations. Major modifications in the flow are enumerated below:

1. Given a fault list, the flow starts with dividing scan-chains into various groups.
2. Conflict based merging of residual cubes occurs in a similar fashion as a single template case.
3. Templates for each k scan-chain groups are extracted individually, after the merging of residual cubes.

4. Decompressor and the ATPG are constrained using k test templates.
5. Switching threshold used as a stopping criteria is now an average of the number of toggles for all k scan-groups.

4.5.3 Procedure to Generate Scan Groups:

Scan groups are generated using a randomized algorithm. The procedure is a variation of well known k-mean clustering algorithm. Before delving into the actual procedure we introduce some new terminology that will be used in the future. Residual Weight: For every scan cell, two values RW0 and RW1, i.e. residual weight for zero and one, are stored for each specified position in a residual cube. In this case, the specified value is 1 followed by a 0 in consecutive scan-cells in the scan chain and it is a prime toggle point, then RW1=1 and RW0=0 for a scan cell with 1 and RW0=1 and RW1=0 for the scan-cell with 1, for a given fault. In this case, the toggle point is a possible toggle point instead, RW value is distributed equally among all possible toggle locations. Sum of Residual Weights SRW: Sum of residual weights consist of two components: SRW0 and SRW1 for each scan cell. It is the sum of RW0 and RW1 for all faults in the targeted fault list. The distance between Scan Chains i and j,

$$D(I,j)=\sum_{a=0}^{a<(longestchain)}(|SRWi(1)-SRWj(1)|+|SRWi(0)-SRWj(0)|)$$

The various steps of the procedure to partition scan-chains into disjoint groups are given below:

1. Select a sampled list FS of 5% of all faults from the set N faults. Compute the residual cubes for one fault in FS, as described before in Section III.

2. While generating the residual cubes, calculate the individual residual weights. Calculate SRW values for targeted fault set FS. Calculate distance D(I,j) between all scan chain pairs.
3. Assign k chains as seed cluster centers and assign each chain to a cluster, depending on the lowest distance cluster center.
4. Calculate the new cluster center by taking the mean of all elements in the cluster and choosing the minimum distance node from the mean as the new cluster center.
5. Repeat step E unless the result from two consecutive iterations results in the same cluster center, or the number of iterations exceed the maximum number of allowed iterations.
6. Scan through the chain list and re-assign the chain to the closest cluster center that is non-empty, i.e. number of chains in a group< (Number of chains)/ k or else assign chain to closest non-empty cluster.
7. Output the group number for each individual scan chain.
8. Once k scan chain groups of equal size are formed, the template is generated for each individual group, separately.

4.6 Isometric Patterns and Compression

Isometric compression has an advantage that only the first toggle among a set of consecutive toggle hold sequences needs to be encoded. The rest of the scan-cells in the chain carry the same equation as the first scan cell in the sequence, resulting

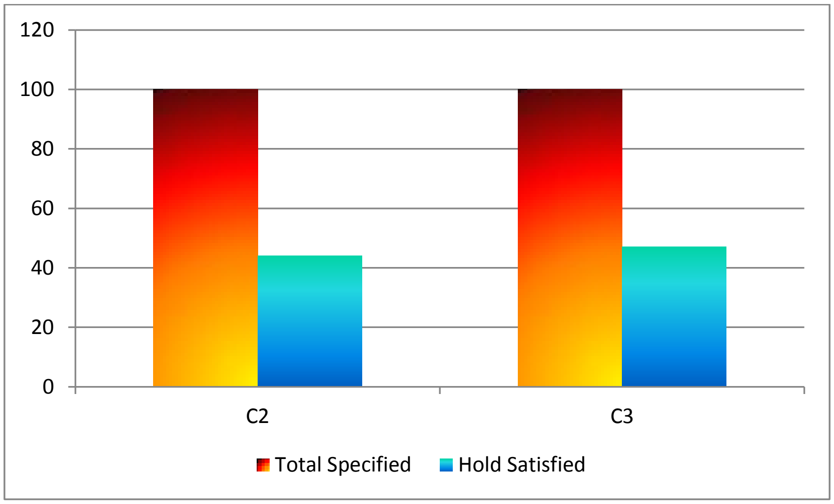


Figure 4.15: Specified position histogram

in the specification of some additional scan-cells without the need of any input data. The above phenomena can help reduce test lengths significantly. Isometric patterns encode a larger number of specified positions on average than the total number of free-variables (input channel bits) available via the tester. We conducted a small experiment to analyze the number of specified positions, required by all deterministically targeted faults that are satisfied by the hold.

The various steps involved in our experimental setup are as follows:

1. Generate isometric test patterns for a given set of k faults.
2. Take each isometric test pattern and set of faults that were specifically targeted.
3. Relax (make X) each specified bit individually one by one, if all deterministically targeted faults are still detected, permanently change the specified position to X, else, revert back to the original values.

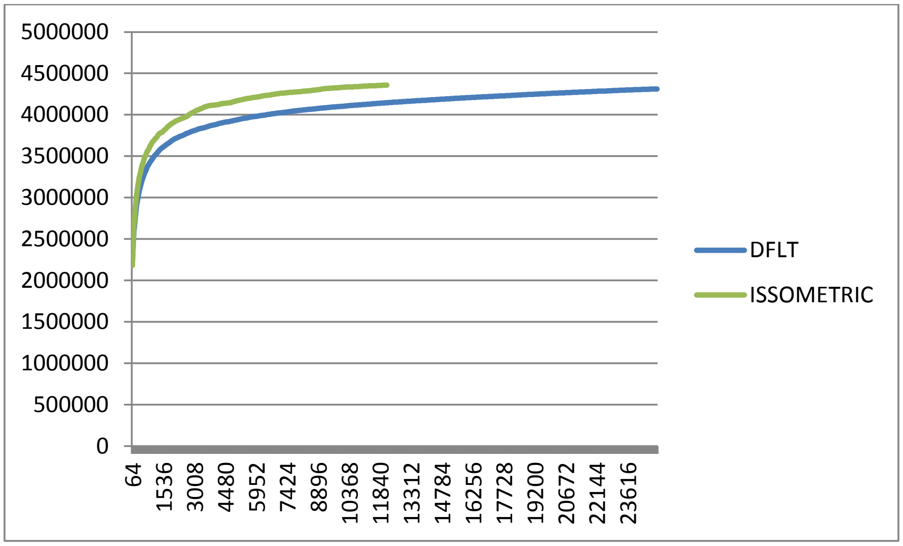


Figure 4.16: Number of faults detected

4. Count the number of specified sequences of the toggle followed by consecutive holds and the total number of specified positions.

In Figure 4.15, we compare the total number of specified positions with the number of positions satisfied by the hold for two circuits; the total number of required specified positions are normalized to 100. Approximately 45% of the specified positions are encoded by the hold and do not require any input data to encode them, resulting in significantly higher compression.

Later in Figure 4.16, we compare the cumulative number of faults detected until a given pattern count, for baseline and isometric compression schemes; isometric scheme detect the same number of faults by 9K patterns in comparison to 25K patterns required by the baseline compression scheme.

Table 4.1: Circuit Properties

CKT	Size	FF Num	Num of Chains	Longest Chain	Input Channels	Output Channels	Decompressor Size
D1	1.1M	75K	382	190	2	5	32
D2	3.3M	73K	244	300	3	3	32
D3	1.3M	52K	209	250	3	3	32
D4	107K	1640	15	115	1	1	16
D5	500K	28K	400	71	4	8	28
D6	400K	41K	400	104	2	5	22
D7	2.2M	167K	1200	142	16	16	65
D8	1.2M	75K	400	202	2	5	32
D9	1.3M	68K	658	104	3	6	64

4.7 Experimental Results

The isometric test compression has been verified by conducting a series of experiments with nine industrial designs. The basic data regarding the designs, such as the number of gates, the number of scan cells, the number of scan chains, the size of the longest chain, the number of EDT input channels, and the size of decompressor are listed in Table 4.1. All experiments were performed for stuck-at tests. The same table reports the fault coverage numbers for a default EDT scheme with approximately a 50% switching ratio for scan-in shift modes as well as three different toggling values used in all experiments presented in this section. A single test session consists of two steps that are repeated several times: (1) shifting in a next test template, and (2) applying 64 test patterns corresponding to this template.

4.7.1 Results for Stuck-at Faults

The results of the experiments are summarized in Tables 4.2 4.5. It is assumed that the number of toggle points per test template is selected in such a way that the resultant scan shift toggling rates are equal to 25% (here the number of toggling

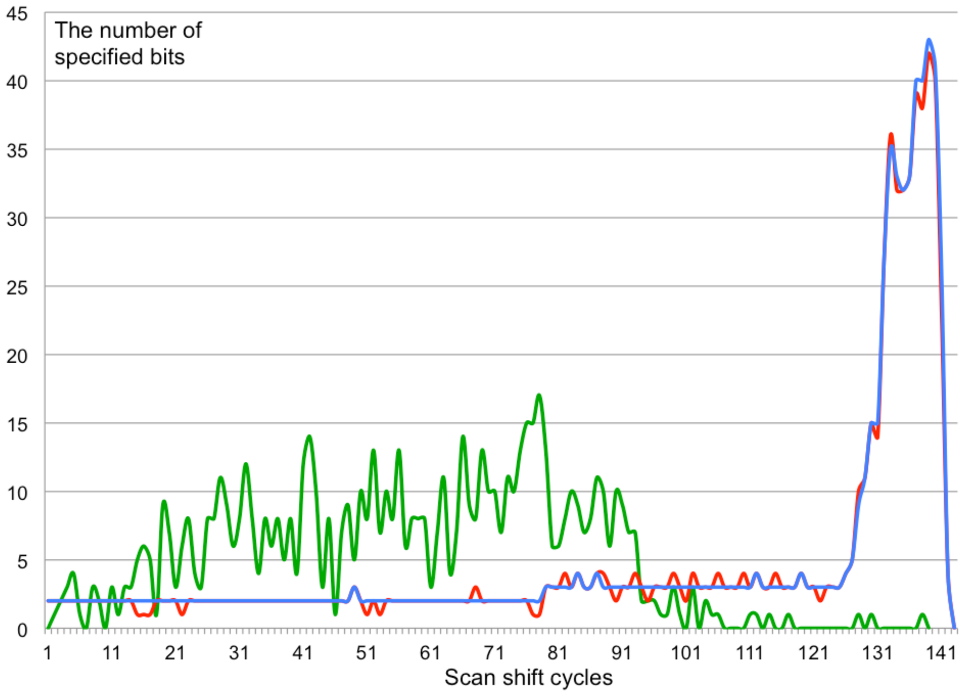


Figure 4.17: Distribution of specified bits

points equals approximately c/2, where c is the number of scan shift cycles), 17%, and 12%. Table 4.2 reports the number of test patterns. Note that the test coverage (see Table 4.2) remains unaffected in each test case. Furthermore, all experiments were performed for 1, 2, 3, and 4 test templates.

Table 4.3 presents the corresponding reduction of input data volume over the conventional EDT-based solution. Note that the data volume for the new method includes the test template data. The results of Table 4.2 clearly indicate that application of the new scheme produces remarkable results. The isometric test compression compares favorably with earlier test data reduction solutions as far as pattern and compression numbers are concerned. In all test cases, compression rates are significantly higher than those of the standard EDT. For example, the observed volume reduction relative to the EDT, even with the test templates overhead (circa 2% as every template is used for 64 successive test patterns), varies from 28% to 77%, and its average value computed across the nine examined industrial designs is equal to 43.95%. Clearly, the proposed scheme elevates compression ratios to levels beyond what is achievable through the conventional static and dynamic reseeding. Furthermore, as shown by the last column of Table 4.2, in all examined test cases the switching activity is reduced to 25%, 17%, and 12% compared to the reference value of 49.31% obtained as the average weighted transition metric over all nine designs for the standard EDT scheme. The resultant scan-in shift induced switching activity, measured by the normalized weighted transition metric (WTM) [59], is shown in Table 4.4, Table 4.5 delivers additional data regarding test power dissipation observed during all experiments. In particular, one can find all WTM metrics for scan shift-out operations in Table 4.5. Design D7 is of special interest, as it represents a test case where specified bits occurring in test cubes are highly clustered with abnormally high, though local, fill rates. Figure 4.17 illustrates a distribution of specified bits for three different test patterns across successive shift cycles. As can be seen, two patterns (represented by blue and red curves) have very high fill rates between 130 and 140 shift cycles. As a result, earlier compression schemes would either resort to increasing pattern counts because of test cube low compressibility or declare a compression abort. On the contrary, the isometric test compression addresses this issue seamlessly with no negative impact on the test data volume and test time.

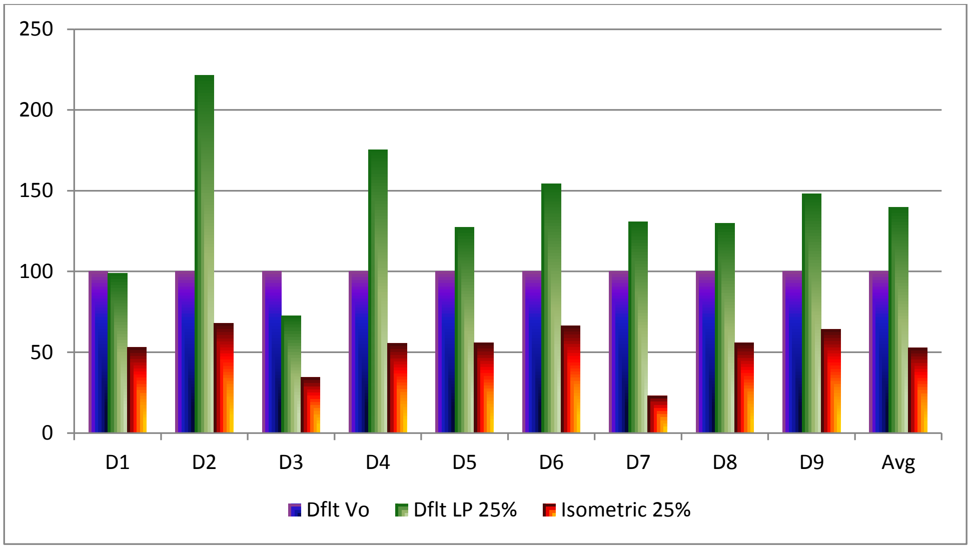


Figure 4.18: Comparison with default low power scheme

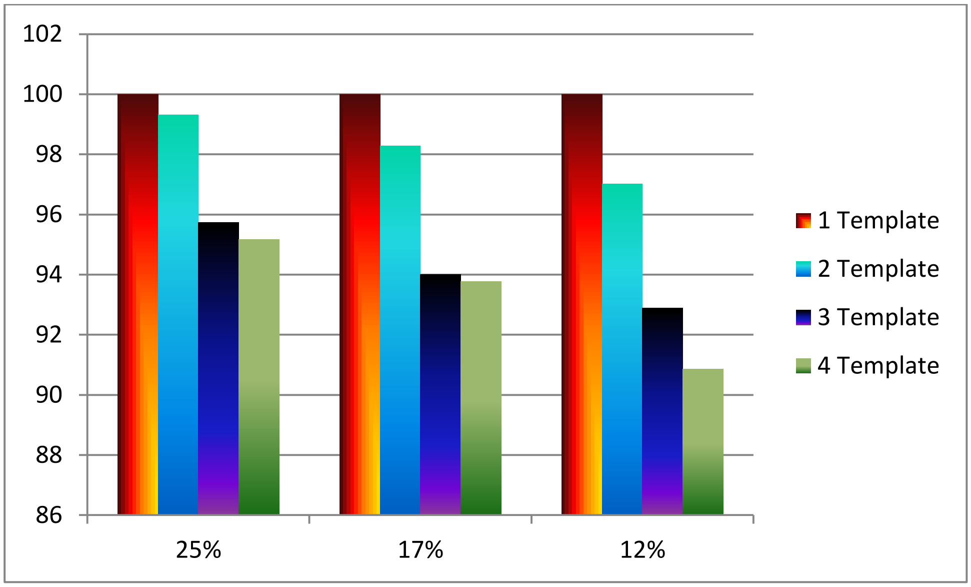


Figure 4.19: Single and multiple templates for various toggle rates

Table 4.2: Test Length for Various Switching Rates and Multiple Templates

CKT	Cov	TL	25 %: Num. of Scan Grps				TL	17 %: Num. of Scan Grps				TL	12 %: Num. of Scan Grps			
		DFLT	1	2	3	4	DFLT	1	2	3	4	DFLT	1	2	3	4
D1	96.71	15040	8512	8320	8000	7963	15040	9344	8832	8576	8448	15040	11136	10389	10116	9853
D2	93.31	5696	3968	3904	3891	3872	5696	4072	4135	4058	4074	5696	4979	5028	4702	4638
D3	94.56	20032	6592	7173	6810	6896	20032	7015	7357	6829	7153	20032	7361	7459	7307	7126
D4	95.31	4992	2,896	2846	2835	2769	4992	3628	3563	3407	3347	4992	4215	4026	3980	3932
D5	99.26	7938	4736	4652	4476	4431	7938	5028	4943	4732	4758	7938	5596	5307	5110	4921
D6	98.63	2048	1472	1457	1382	1361	2048	1641	1637	1593	1576	2048	1994	1920	1816	1778
D7	91	2496	576	593	568	572	2496	837	806	814	793	2496	1025	1018	960	942
D8	97.04	8064	5,120	4827	4602	4516	8064	5834	5624	5257	5114	8064	6015	5853	5405	5318
D9	95.94	9359	6471	6291	6059	6012	9359	7002	6738	6472	6370	9359	7492	7328	6875	6753

Table 4.3: Volume for Various Switching Rates and Multiple Templates

CKT	Cov	VO	25 %: Num. of Scan Grps				VO	17 %: Num. of Scan Grps				VO	12 %: Num. of Scan Grps			
		DFLT	1	2	3	4	DFLT	1	2	3	4	DFLT	1	2	3	4
D1	96.71	7.04	3.98	3.89	3.74	3.73	7.04	4.37	4.13	4.01	3.95	7.04	5.21	4.86	4.74	4.61
D2	93.31	5.51	3.84	3.78	3.76	3.75	5.51	3.94	4.00	3.93	3.94	5.51	4.82	4.86	4.55	4.49
D3	94.56	15.02	4.94	5.38	5.11	5.17	15.02	5.26	5.52	5.12	5.36	15.02	5.52	5.59	5.48	5.34
D4	95.31	0.69	0.40	0.39	0.39	0.38	0.69	0.50	0.49	0.47	0.46	0.69	0.58	0.56	0.55	0.54
D5	99.26	3.14	1.87	1.84	1.77	1.75	3.14	1.99	1.96	1.87	1.88	3.14	2.21	2.10	2.02	1.95
D6	98.63	0.59	0.42	0.42	0.40	0.39	0.59	0.47	0.47	0.46	0.45	0.59	0.57	0.55	0.52	0.51
D7	91	6.34	1.46	1.51	1.44	1.45	6.34	2.13	2.05	2.07	2.01	6.34	2.60	2.59	2.44	2.39
D8	97.04	3.90	2.48	2.33	2.23	2.18	3.90	2.82	2.72	2.54	2.47	3.90	2.91	2.83	2.61	2.57
D9	95.94	4.26	2.95	2.86	2.76	2.74	4.26	3.19	3.07	2.95	2.90	4.26	3.41	3.34	3.13	3.07

Table 4.4: Toggling Rates (WTM) Scan Shift-in

CKT	LP 25%	25 %: Num. of Scan Grps				LP 17%	17 %: Num. of Scan Grps				LP 12%	12 %: Num. of Scan Grps			
	DFLT	1	2	3	4	DFLT	1	2	3	4	DFLT	1	2	3	4
D1	23.94	21.31	22.29	22.09	22.7	16.76	16.72	16.53	16.27	16.74	14.13	11.63	11.87	12.37	12.15
D2	24.06	23.07	23.17	23.81	24.09	16.07	15.78	16.42	16.6	17.01	13.01	10.72	12.01	12.19	12.31
D3	24.16	20.56	21.08	21.49	21.92	17.44	16.83	16.66	16.28	16.96	12.76	11.44	11.49	12.06	12.2
D4	23.73	22.69	22.98	22.85	22.87	17.49	17.15	16.12	16.51	16.85	12.59	11.09	11.14	11.72	12.01
D5	23.77	23.18	23.32	23.64	24.66	16.58	16.24	16.07	16.53	16.58	12.83	10.23	11.15	11.93	11.85
D6	22.94	21.54	21.96	22.74	23.13	16.5	16.4	15.39	16.04	16.61	11.64	11.26	11.25	11.98	11.63
D7	23.5	23.07	23.8	24.08	24.91	16.83	16.63	16.41	16.71	17.08	13.88	10.13	10.68	12.14	11.88
D8	23.01	20.76	21.23	21.46	22.08	17.29	16.72	16.01	16.14	16.62	12.09	10.63	11.44	11.51	11.8
D9	24.22	24.13	23.09	24.25	24.42	17.32	16.94	16.72	16.74	16.95	12.41	11.19	11.73	11.41	12.15

Table 4.5: Toggling Rates (WTM) Scan Shift-out

CKT	LP 25%	25 %: Num. of Scan Grps				LP 17%	17 %: Num. of Scan Grps				LP 12%	12 %: Num. of Scan Grps			
	DFLT	1	2	3	4	DFLT	1	2	3	4	DFLT	1	2	3	4
D1	24.26	21.85	22.45	22.52	23.16	16.76	16.98	16.62	16.34	16.87	14.16	11.74	12.41	12.54	12.23
D2	24.68	23.21	23.20	24.66	24.27	16.07	16.93	16.95	16.80	17.05	13.07	11.40	12.37	12.85	12.58
D3	24.24	21.22	21.27	21.71	22.68	17.44	15.88	17.19	16.64	17.61	13.18	11.87	11.64	12.12	12.35
D4	24.11	23.34	23.27	23.05	23.47	17.49	15.94	16.20	17.16	16.97	12.69	11.55	11.62	12.02	12.44
D5	23.90	24.65	23.42	23.96	25.08	16.59	16.60	16.59	16.61	17.24	13.00	10.91	11.29	11.92	12.42
D6	23.24	21.79	22.55	22.82	23.25	16.50	16.18	15.57	16.94	16.85	12.16	11.89	11.55	12.39	12.20
D7	24.00	23.62	23.99	24.39	25.22	16.84	16.08	16.67	16.81	17.74	14.00	10.81	10.83	12.76	12.19
D8	23.04	21.74	21.72	21.95	22.69	17.29	16.36	16.40	16.68	17.20	12.65	11.16	11.82	11.78	12.39
D9	24.44	24.13	23.57	24.34	24.49	17.32	16.57	16.83	16.93	17.57	12.77	11.45	11.81	12.69	12.53

Table 4.6: Results for non-BIST ready designs

	Random ATPG		Method1 + New Ordering		Method2		Isometric	
CKT	FC	TL	TL	%Red	TL	%Red	TL	%Red
D1	96.71	15040	11016	26.8	12212	18.8	7963	47.1
D2	93.31	5696	4385	23.0	4437	22.1	3872	32.0
D3	94.56	20032	9672	51.7	11667	41.8	6896	65.6
D4	95.31	4992	4232	15.2	4033	19.2	2769	44.5
D5	99.26	7938	6389	19.5	6933	12.7	4431	44.2
D6	98.63	2048	1706	16.7	1802	12.0	1361	33.5
D7	91	2496	1893	24.2	1914	23.3	572	77.1
D8	97.04	8064	5918	26.6	6273	22.2	4516	44.0
D9	95.94	9359	7626	18.5	7624	18.5	6012	35.8
Avg.		75665	52837	30.2	56893	24.8	38392	49.3

4.7.2 Comparison With a Low-Power Compression Scheme

In Figure 4.18 we compare the results for data volume obtained by the default baseline ATPG with a 50% toggle rate, a commercial low power compression scheme with 25% toggling during shift, and the proposed isometric compression scheme with 25% toggling and using multiple template flow (four templates were used). The data volume values are normalized with respect to the default ATPG data volume, which is assigned a value of 100. Isometric compression results in a 47% reduction in data volume over the default ATPG and the default low power by 62%.

4.7.3 Impact of Using Multiple Templates

In Figure 4.19, we compare normalized test lengths obtained by average test length obtained by 1 template case assigned a value of 100 with 2, 3, and 4 templates for 25%, 17%, and 12% toggle rates. As illustrated in the figure, multiple templates are most effective for lower toggle rates. A 12% toggle rate using 4 templates results in a reduction of 10% test length in comparison to test lengths obtained by 1 template.

A toggle rate of 25% results in 5% reduction in the test length.

4.7.4 Comparison With Other ATPG Techniques

We compare the results obtained by Method-1 combined with test enumeration based fault ordering and Method-2 discussed in Chapter 3 with isometric compression approach in Table 4.6. Test lengths are normalized with respect to those obtained by the default ATPG which is given a test length of 100. The isometric scheme out performs all ATPG decision schemes with a 49% reduction in pattern count using 4 templates in comparison to the baseline compression scheme.

CHAPTER 5
CONCLUSIONS

This thesis focused on two major issues thst incurred while testing digital circuits using compression: pattern count and low power test.

First, we propose methods to guide line justification and D-propagation decisions in ATPG. Static and dynamic measures, which are updated during test generation, are used to achieve substantial reduction in test set sizes. A method to select between static and dynamic measures, based on the level of the decision node, is also given. Experimental results on several industrial designs show that, on average, the proposed methods reduce by 27% the sizes of the test sets generated by a state of the art commercial ATPG.

Further, we propose a new ATPG which uses a novel FFR based fault ordering technique and guidance measures for justification and fault effect propagation. Experimental results on several industrial designs show that for BIST ready designs on average, the proposed ATPG reduces test set sizes by 24% in comparison to a state of the art commercial ATPG for non-BIST ready designs and 28% for BIST ready designs. Transition faults are difficult to test and often result in longer test lengths, we propose a new fault ordering technique based on test enumeration. This ordering technique when combined with Method-1 [5] for stuck-at fault, resulted in a 30% reduction in test length. A new guidance approach was also proposed for transition faults. Test set sizes were reduced by 32% for transition faults, using the proposed methods.

With the size of test patterns growing at alarming rates, the isometric test compression proposed in this thesis provides a coherent way to generate and apply test patterns compressed beyond the limits achievable by existing solutions, while substantially reducing power dissipation during scan loading. The proposed scheme is non-intrusive to the core logic, and offers a programmable means to control the scan toggling rates. Furthermore, it has the ability to trade off the test power consumption and the resultant compression. As the new compression preserves all benefits of continuous flow decompression, it provides a smooth migration path to the next-generation test data compression solutions while maintaining a very high quality of test, minimizing design, and manufacturing cost impacts. On average, the test set sizes were reduced by 49%, in comparison to a commercial ATPG tool.

5.1 Future Work

Traditional fault models like stuck-at and transition fault models define fault sites at the boundary of a cell. However, 50% of the faults are estimated to be present internally in cells. Defect-oriented fault models were proposed in [14]. In [14], new user defined fault models were developed depending on the cells internal defects; the approach is known to detect real defects in the field. However, compact test generation for defect-aware fault models is difficult as:

- Fortuitous detection of faults is low, as a definite combination of inputs needs to be applied to detect user-defined faults.
- The test has a higher number of specified positions per fault, hence are not

easily compressible in a compression environment.

The generation of compact test generation using isometric test compression scheme is still an unexplored area. Consecutive hold and toggle cycles used in isometric schemes results in additional constraints, leading to reduced fortuitous detection.

In isometric test compression scheme, multiple template registers were used to provide flexibility. However, a given assignment of groups of scan-chains permanently to a template register may work only for few faults. A scheme that assigns scan-chains to a scan-group depending on the current set of undetected faults will provide more flexibility during test generation, leading to compact test sets.

REFERENCES

[1] M. Abramovici, A. Friedman, and M. Breuer, *Digital Systems Testing and Testable Design*. Electrical engineering, communications and signal processing, IEEE, 1994.

[2] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability (Systems on Silicon)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006.

[3] X. Lin, E. K. Moghaddam, N. Mukherjee, B. Nadeau-Dostie, J. Rajski, and J. Tyszer, “Power aware embedded test,” in *ATS*, pp. 511–516, 2011.

[4] X. Lin and J. Rajski, “On utilizing test cube properties to reduce test data volume further,” in *Proc. of ATS*, 2012.

[5] A. Kumar, J. Rajski, S. M. Reddy, and C. Wang, “On the generation of compact test sets,” in *ITC*, 2013.

[6] A. Kumar, J. Rajski, S. M. Reddy, and T. Rinderknecht, “On the generation of compact deterministic test sets for bist ready designs,” in *ATS*, 2013.

[7] A. Kumar, M. Kassab, E. Moghadham, N. Mukherjee, J. Tyszer, J. Rajski, S. M. Reddy, and C. Wang, “Isometric test compression with low toggling activity,” in *ITC*, 2014.

[8] A. Crouch, *Design-for-test for Digital IC's and Embedded Core Systems.* No. v. 1 in Design-for-test for Digital IC's and Embedded Core Systems, Prentice Hall PTR, 1999.

[9] J. P. Roth, “Diagnosis of automata failures: a calculus and a method,” in *IBM Journal of Research & Devlopment*, vol. 10, pp. 278 –291, july 1966.

[10] P. Goel, “An implicit enumeration algorithm to generate tests for combinational logic circuits,” *IEEE Trans. on Computers*, vol. C-30, pp. 215 –222, march 1981.

[11] E. B. Eichelberger and T. W. Williams, “A logic design structure for lsi testability,” in *Papers on Twenty-five years of electronic design automation*, 25 years of DAC, (New York, NY, USA), pp. 358–364, ACM, 1988.

[12] J. Waicukauski, E. Lindbloom, B. K. Rosen, and V. Iyengar, “Transition fault simulation,” *Design Test of Computers, IEEE*, vol. 4, no. 2, pp. 32–38, 1987.

[13] G. L. Smith, “Model for delay faults based upon paths,” *International Test Conference*, pp. 342–349, 1985.

[14] F. Hapke, R. Krenz-Baath, A. Glowatz, J. Schloeffel, H. Hashempour, S. Eichenberger, C. Hora, and D. Adolfsson, “Defect-oriented cell-aware atpg and fault simulation for industrial cell libraries and designs,” in *International Test Conference*, pp. 1–10, 2009.

[15] K. Mei, "Bridging and stuck-at faults," *IEEE Transactions on Computers*, vol. C-23, no. 7, pp. 720–727, 1974.

[16] F. Ferguson and T. Larrabee, "Test pattern generation for realistic bridge faults in cmos ics," in *International Test Conference*, pp. 492–, 1991.

[17] E. Trischler, "Incomplete scan path with an automatic test generation methodology," in *Procs. of ITC*, pp. 153–162, 1980.

[18] A. Mudlapur, V. Agrawal, and A. Singh, "A random access scans architecture to reduce hardware overhead," in *IEEE International Test Conference*, pp. 358–366, 2005.

[19] C. Glover and M. Mercer, "A method of delay fault test generation," in *Design Automation Conference*, pp. 90–95, 1988.

[20] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-in test for VLSI: pseudorandom techniques*. New York, NY, USA: Wiley-Interscience, 1987.

[21] N. Tamarapalli and J. Rajski, "Constructive multi-phase test point insertion for scan-based bist," in *ITC*, pp. 649 –658, oct 1996.

[22] G. Hetherington, T. Fryars, N. Tamarapalli, M. Kassab, A. Hassan, and J. Rajski, "Logic bist for large industrial designs: real issues and case studies," in *ITC*, pp. 358 –367, 1999.

[23] H. Tang, C. Wang, J. Rajski, S. M. Reddy, J. Tyszer, and I. Pomeranz, "On efficient x-handling using a selective compaction scheme to achieve high test response compaction ratios," in *VLSI Design*, pp. 59–64, 2005.

[24] M. Geuzebroek, J. van der Linden, and A. van de Goor, "Test point insertion that facilitates atpg in reducing test time and data volume," in *ITC*, pp. 138 – 147, 2002.

[25] S. Remersaro, J. Rajski, T. Rinderknecht, S. Reddy, and I. Pomeranz, "Atpg heuristics dependant observation point insertion for enhanced compaction and data volume reduction," in *Symp. on Defect and Fault Tolerance of VLSI Systems*, pp. 385 –393, oct. 2008.

[26] J.-S. Yang, N. Touba, and B. Nadeau-Dostie, "Test point insertion with control points driven by existing functional flip-flops," *IEEE Trans. on Computers*, vol. 61, pp. 1473 –1483, oct. 2012.

[27] B. Koenemann, "Lfsr-coded test patterns for scan designs," *European Test Conf*, pp. 237–242, 1991.

[28] J. Rajski, J. Tyszer, M. Kassab, N. Mukherjee, R. Thompson, K. Tsai, A. Hertwig, N. Tamarapalli, and G. Mrugalski, "Embedded deterministic test for low cost manufacturing test," in *Proc. of ITC*, pp. 301–310, 2002.

[29] P. Goel and B. Rosales, "Test generation and dynamic compaction of test," in

Annu Test Conference, pp. 260 – 268, 1979.

[30] I. Pomeranz, L. Reddy, and S. Reddy, "Compactest: a method to generate compact test sets for combinational circuits," *IEEE Trans. on CAD*, vol. 12, pp. 1040 –1049, jul 1993.

[31] B. Ayari and B. Kaminska, "A new dynamic test vector compaction for automatic test pattern generation," *IEEE Trans. on CAD*, vol. 13, pp. 353 –358, mar 1994.

[32] S. Kajihara, I. Pomeranz, K. Kinoshita, and S. Reddy, "Cost-effective generation of minimal test sets for stuck-at faults in combinational logic circuits," *IEEE Transactions on CAD*, vol. 14, pp. 1496 –1504, dec 1995.

[33] K. Miyase and S. Kajihara, "Xid: Don't care identification of test patterns for combinational circuits," *IEEE Trans. on CAD*, vol. 23, pp. 321–326, Nov. 2006.

[34] I. Pomeranz and S. Reddy, "Forward-looking reverse order fault simulation for -detection test sets," *IEEE Trans. on CAD*, vol. 28, no. 9, pp. 1424–1428, 2009.

[35] I. Hamzaoglu and J. Patel, "Test set compaction algorithms for combinational circuits," in *Procs. of ICCAD*, pp. 283 – 289, 1998.

[36] M. Konijnenburg, J. van der Linden, and A. van de Goor, "Compact test sets for industrial circuits," pp. 358 –366, 1995.

[37] Z. Wang and D. Walker, “Dynamic compaction for high quality delay test,” pp. 243 –248, 2008.

[38] S. Remersaro, J. Rajski, S. Reddy, and I. Pomeranz, “A scalable method for the generation of small test sets,” in *Proc. of DATE*, pp. 1136 –1141, 2009.

[39] P. Wohl, J. Waicukauski, and T. Finklea, “Increasing prpg-based compression by delayed justification,” in *Proc of ITC*, pp. 1 –10, nov. 2010.

[40] H. Fujiwara, “Fan: A fanout-oriented test pattern generation algorithm,” in *ISCAS*, 1985.

[41] M. Schulz, E. Trischler, and T. Sarfert, “Socrates: a highly efficient automatic test pattern generation system,” *IEEE Transactions on CAD*, vol. 7, no. 1, pp. 126–137, 1988.

[42] W. Kunz and D. Pradhan, “Recursive learning: a new implication technique for efficient solutions to cad problems-test, verification, and optimization,” *IEEE Transactions on CAD*, vol. 13, no. 9, pp. 1143–1158, 1994.

[43] M. Henftling, H. Wittmann, and K. Antreich, “A single-path-oriented fault-effect propagation in digital circuits considering multiple-path sensitization,” in *ICCAD*, pp. 304–309, 1995.

[44] L. Reddy, I. Pomeranz, and S. Reddy, “Rotco: a reverse order test compaction technique,” in *Euro ASIC*, pp. 189–194, 1992.

[45] D. Hochbaum, “An optimal test compression procedure for combinational circuits,” *IEEE Trans. on CAD*, vol. 15, no. 10, pp. 1294–1299, 1996.

[46] I. Pomeranz and S. M. Reddy, “Forward-looking fault simulation for improved static compaction,” *IEEE Trans. on CAD*, vol. 20, pp. 1262–1265, Nov. 2006.

[47] S. Hellebrand, B. Reeb, S. Tarnick, and H.-J. Wunderlich, “Pattern generation for a deterministic bist scheme,” in *ICCAD*, pp. 88–94, 1995.

[48] M. Sauer, S. Reimer, I. Polian, T. Schubert, and B. Becker, “Provably optimal test cube generation using quantified boolean formula solving,” in *ASP-DAC*, pp. 533–539, 2013.

[49] S. Hillebrecht, M. A. Kochte, D. Erb, H.-J. Wunderlich, and B. Becker, “Accurate qbf-based test pattern generation in presence of unknown values,” in *DATE*, pp. 436–441, 2013.

[50] 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,” in *ITC*, pp. 1098–1104, 2003.

[51] S. Almukhaizim and O. Sinanoglu, “Peak power reduction through dynamic partitioning of scan chains,” in *ITC*, pp. 1–10, 2008.

[52] Q. Xu, “Pattern-directed circuit virtual partitioning for test power reduction,” in *Proc. IEEE International Test Conference (ITC) 2007, paper 25.2.*

[53] V. Iyengar and K. Chakrabarty, "Precedence-based, preemptive, and power-constrained test scheduling for system-on-a-chip," in *VTS*, pp. 368–374, 2001.

[54] Y. Bonhomme, "Efficient scan chain design for power minimization during scan testing under routing constraint," in *ITC*, pp. 488–493, 2003.

[55] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S. Reddy, "Techniques for minimizing power dissipation in scan and combinational circuits during test application," *IEEE Trans on CAD*, vol. 17, no. 12, pp. 1325–1333, 1998.

[56] S. Remersaro, X. Lin, Z. Zhang, S. Reddy, I. Pomeranz, and J. Rajski, "Preferred fill: A scalable method to reduce capture power for scan based designs," in *ITC*, pp. 1–10, 2006.

[57] W. Li, S. M. Reddy, and I. Pomeranz, "On reducing peak current and power during test," in *ISVLSI*, ISVLSI '05, (Washington, DC, USA), pp. 156–161, IEEE Computer Society, 2005.

[58] E. K. Moghaddam, J. Rajski, S. M. Reddy, and J. Janicki, "Low test data volume low power at-speed delay tests using clock-gating," in *ATS*, pp. 267–272, 2011.

[59] J. Rajski, E. K. Moghaddam, and S. M. Reddy, "Low power compression utilizing clock-gating," in *ITC*, pp. 1–8, 2011.

[60] D. Czysz, G. Mrugalski, N. Mukherjee, J. Rajski, P. Szczerbicki, and J. Tyszer, "Low power compression of incompatible test cubes," in *ITC*, pp. 1–10, 2010.

[61] D. Czysz, M. Kassab, X. Lin, G. Mrugalski, J. Rajski, and J. Tyszer, "Low-power scan operation in test compression environment," *IEEE Trans of CAD*, vol. 28, no. 11, pp. 1742–1755, 2009.

[62] M. Filipek, Y. Fukui, H. Iwata, G. Mrugalski, J. Rajski, M. Takakura, and J. Tyszer, "Low power decompressor and prpg with constant value broadcast," in *ATS*, pp. 84–89, 2011.

[63] J. Lee and N. Touba, "Low power test data compression based on lfsr reseeding," in *ICCD*, pp. 180–185, 2004.

[64] L. Goldstein and E. Thigpen, "Scoap: Sandia controllability/observability analysis program," in *DAC*, pp. 190 – 196, june 1980.

[65] P. Wohl, J. A. Waicukauski, F. Neuveux, and E. Gizdarski, "Fully x-tolerant, very high scan compression," in *DAC*, pp. 362–367, 2010.

[66] N. A. Touba, "Survey of test vector compression techniques," *IEEE Design and Test of Computers*, vol. 23, pp. 294–303, 2006.

[67] I. Pomeranz and S. M. Reddy, "The accidental detection index as a fault ordering heuristic for full-scan circuits," in *DATE*, pp. 1008–1013, 2005.

[68] P. Krauss and M. Henftling, "Efficient fault ordering for automatic test pattern generation for sequential circuits," in *ATS*, pp. 113 –118, nov 1994.

[69] M. Messing, A. Glowatz, F. Hapke, and R. Drechsler, "Using a two-dimensional fault list for compact automatic test pattern generation," in *LATW*, pp. 1 –6, march 2009.