Fall 2010

Diagnosis Of VLSI circuit defects: defects in scan chain and circuit logic

Xun Tang
University of Iowa

DIAGNOSIS OF VLSI CIRCUIT DEFECTS:

DEFECTS IN SCAN CHAIN AND CIRCUIT LOGIC

by

Xun Tang

An Abstract

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

December 2010

Thesis Supervisor: Professor Sudhakar M. Reddy

ABSTRACT

Given a logic circuit that fails a test, diagnosis is the process of narrowing down the possible locations of the defects. Diagnosis to locate defects in VLSI circuits has become very important during the yield ramp up process, especially for 90 nm and below technologies where physical failure analysis machines become less successful due to reduced defect visibility by the smaller feature size and larger leakage currents. Successful defect isolation relies heavily on the guidance from fault diagnosis and will depend even more for the future technologies.

To assist a designer or a failure analysis engineer, the diagnosis tool tries to identify the possible locations of the failure effectively and quickly. While many defects reside in the logic part of a chip, defects in scan chains have become more and more common recently as typically 30%-50% logic gates impact the operation of scan chains in a scan design. Logic diagnosis and scan chain diagnosis are the two main fields of diagnosis research. The quality of diagnosis directly impacts the time-to-market and the total product cost. Volume diagnosis with statistical learning is important to discover systematic defects. An accurate diagnosis tool is required to diagnose large numbers of failing devices to aid statistical yield learning. In this work, we propose techniques to improve diagnosis accuracy and resolution, techniques to improve run-time performance.

We consider the problem of determining the location of defects in scan chains and logic. We investigate a method to improve the diagnosability of production compressed test patterns for multiple scan chain failures. Then a method to generate special diagnostic patterns for scan chain failures was proposed. The method tries to generate a complete test pattern set to pinpoint the exact faulty scan cell when flush tests tell which scan chain is faulty.

Next we studied the problem of diagnosis of multiple faults in the logic of circuits. First we propose a method to diagnose multiple practical physical defects using

simple logic fault models. At last we propose a method based on fault-tuple equivalence trees to further improve diagnosis quality.

Abstract Approved: _____
Thesis Supervisor

_____
Title and Department

_____
Date

DIAGNOSIS OF VLSI CIRCUIT DEFECTS:

DEFECTS IN SCAN CHAIN AND CIRCUIT LOGIC

by

Xun Tang

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 2010

Thesis Supervisor: Professor Sudhakar M. Reddy

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

PH.D. THESIS

_____

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

Xun Tang

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 2010 graduation.

Thesis Committee: _____
Sudhakar M. Reddy, Thesis Supervisor

_____
Wu-Tung Cheng

_____
Soura Dasgupta

_____
Jon G. Kuhl

_____
Hantao Zhang

_____
David R. Andersen

To My Family

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my academic advisor, Professor Sudhakar M. Reddy, for his excellent guidance and solid management throughout this research. Without his guidance, this work would not be even possible. Equal amount of thanks are given to Dr. Wu-Tung Cheng and Dr. Ruifeng Guo for their constructive advices and knowledgeable explanations. I also want to thank my committee members Prof. Kuhl, Prof. Dasgupta, Prof. Zhang, and Prof. Andersen for serving on my committee and giving valuable suggestions.

Many thanks to my friends and co-workers in my research: Huaxing Tang, Yu Huang, Manish Sharma, Liyang Lai, Elham Moghaddam Chris Schuermyer, Xin Dou.

Finally I would like to thank my parents, family and close friends for their great support and encouragement along this journey.

ABSTRACT

Given a logic circuit that fails a test, diagnosis is the process of narrowing down the possible locations of the defects. Diagnosis to locate defects in VLSI circuits has become very important during the yield ramp up process, especially for 90 nm and below technologies where physical failure analysis machines become less successful due to reduced defect visibility by the smaller feature size and larger leakage currents. Successful defect isolation relies heavily on the guidance from fault diagnosis and will depend even more for the future technologies.

To assist a designer or a failure analysis engineer, the diagnosis tool tries to identify the possible locations of the failure effectively and quickly. While many defects reside in the logic part of a chip, defects in scan chains have become more and more common recently as typically 30%-50% logic gates impact the operation of scan chains in a scan design. Logic diagnosis and scan chain diagnosis are the two main fields of diagnosis research. The quality of diagnosis directly impacts the time-to-market and the total product cost. Volume diagnosis with statistical learning is important to discover systematic defects. An accurate diagnosis tool is required to diagnose large numbers of failing devices to aid statistical yield learning. In this work, we propose techniques to improve diagnosis accuracy and resolution, techniques to improve run-time performance.

We consider the problem of determining the location of defects in scan chains and logic. We investigate a method to improve the diagnosability of production compressed test patterns for multiple scan chain failures. Then a method to generate special diagnostic patterns for scan chain failures was proposed. The method tries to generate a complete test pattern set to pinpoint the exact faulty scan cell when flush tests tell which scan chain is faulty.

Next we studied the problem of diagnosis of multiple faults in the logic of circuits. First we propose a method to diagnose multiple practical physical defects using

simple logic fault models. At last we propose a method based on fault-tuple equivalence trees to further improve diagnosis quality.

TABLE OF CONTENTS

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER

CHAPTER 1. INTRODUCTION 1

CHAPTER 2. REVIEW OF DEFECT DIAGNOSIS ALGORITHMS 3

2.1 Review of Scan Chain Diagnosis 4
2.1.1 Tester-Based Scan Chain Diagnosis 7
2.1.2 Hardware-Based Scan Chain Diagnosis 8
2.1.3 Software-Based Scan Chain Diagnosis 10
2.2 Review of Logic Diagnosis 18
2.2.1 Single Stuck-At Defect Diagnosis Using the Stuck-At Fault Model 18
2.2.2 Multiple Fault Diagnosis 21

CHAPTER 3. IMPROVING COMPRESSED PATTERN GENERATION FOR MULTIPLE SCAN CHAIN FAILURE DIAGNOSIS 38

3.1 Introduction 38
3.2 Preliminaries 39
3.2.1 Test response compactors 39
3.2.2 Problem formulation 41
3.3 A Method to Improve Diagnostic Resolution of Production Tests 44
3.3.1 Normal test generation flow 44
3.3.2 Proposed chain selection procedure 45
3.4 Experimental Results 48

CHAPTER 4. IMPROVING DIAGNOSTIC TEST GENERATION FOR SCAN CHAIN FAILURES USING MULTI-CYCLE SCAN PATTERNS 54

4.1 Motivation 54
4.2 Previous Work 54
4.3 Preliminaries 55
4.3.1 Analysis of the test generation algorithm in [Guo 07] 57
4.4 Procedures to Improve Diagnostic Resolution 60
4.4.1 Differentiate Scan Cells through Data Input Capture 61
4.4.2 Sequential Depth for Procedure 1 63
4.4.3 Practical Considerations 65
4.4.4 Extension to Timing Failures 67
4.5 Improve Runtime Performance of Diagnostic Test Generation 68
4.5.1 The Key Idea behind the Speed-up Flow 69
4.5.2 The Proposed Diagnostic Flow for Speeding-up 71
4.6 Experimental Results 73

CHAPTER 5. DIAGNOSIS OF MULTIPLE PHYSICAL DEFECTS USING LOGIC FAULT MODELS 80

5.1 Introduction 80
5.2 Diagnosis Method Based On SLAT 83
5.2.1 Preliminaries 83
5.2.2 Multiple Physical Defect Diagnosis 84
5.2.3 Experimental Results for Multiple Physical Defect 94
5.3 Diagnosis Method Based on FTET 99
5.3.1 Analysis of Diagnosis Method in [Ye 10] 100
5.3.2 Method to Identify Faulty Locations Based On FTET 103
5.3.3 Experimental Results for Multiple Physical Defects 108

CHAPTER 6. CONCLUSIONS 114

6.1 Conclusion of Completed Research 115
6.2 Future Research 117

REFERENCES 119

LIST OF TABLES

Table 1: Two-pattern Flush Tests 5
Table 2: The Statistics and Pattern Counts of 4 Designs 49
Table 3: Experimental Results for Stuck-at Faults on 2 Faulty Chains 50
Table 4: Experimental Results for Stuck-at Faults on 3 Faulty Chains 51
Table 5: Experimental Results for Timing Faults on 2 Faulty Chains 51
Table 6: Design Profile 75
Table 7: Diagnostic Test Generation for Design 1 75
Table 8: Diagnostic Test Generation for Design 2 76
Table 9: Diagnostic Test Generation for Design 3 76
Table 10: Diagnostic Test Generation for Design 4 78
Table 11: Diagnostic Test Generation for Design 5 78
Table 12: An example of suspect selection (α=0.4).....93
Table 13: Updated table 94
Table 14: Design Characteristics 95
Table 15: Experimental Results for Two Physical Defects 97
Table 16: Experimental Results for Three Physical Defects 97
Table 17: Faulty Location Identification Using Two Methods 112
Table 18: Diagnosis Results Using Three Methods.....113
Table 19: Diagnosis Results Using Three Methods.....114

LIST OF FIGURES

Figure 1: An Example Scan 4
Figure 2: Identification of upper bound (UB) and low bound (LB) 11
Figure 3: Jump Simulation 12
Figure 4: Dynamic learning and single-excitation ATPG 13
Figure 5: Fault Detected at SO of Good Chain 17
Figure 6: Proposed Flow 22
Figure 7: Site Selection Example 24
Figure 8: Multiple Fault Pattern Types 29
Figure 9: Overview of Diagnosis Method in [Ye 10] 37
Figure 10: A Space Compactor with 3 Output Channels 39
Figure 11: Scan Chain Selection Logic 40
Figure 12: Illustrating Multiple Chain Failures 42
Figure 13: Normal Chain Selection Procedure 45
Figure 14: The Proposed Chain Selection Procedure 46
Figure 15: Illustration of Scan Cell N Impacting Cell N-1 59
Figure 16: Illustration of Using Multi-Cycle Scan Patterns 60
Figure 17: An Example with a Stuck-at-0 Fault 61
Figure 18: Scan Cell Constraints 69
Figure 19: Diagnostic Test Generation Flow 72
Figure 20: Failing Patterns Types 83
Figure 21: Diagnosis Flow Overview 84
Figure 22: Example of Bit Union 86
Figure 23: Net-open Defects 88
Figure 24: Illustration of Diagnosis Challenge 89
Figure 25: Comparison for Two Defects 95

Figure 26: Comparison for Three Defects 96
Figure 27: A Faulty Circuit under Pattern abc(101) 100
Figure 28: Diagnosis methodology overview [Ye 10].....101
Figure 29: Example of FTET Construction [Ye 10].....102
Figure 30: An example of conflict 104
Figure 31: Faulty S27 Circuit under Pattern P1 105
Figure 32: FTET of Pattern P1 for Faulty S27 Circuit 106
Figure 33: FTET of Pattern P4 for Faulty S27 Circuit 107

CHAPTER 1. INTRODUCTION

Following the so-called Moore's Law, the scale of Integrated Circuits (ICs) has doubled every 18 months. The term "VLSI" originally was used for chips having more than 100,000 transistors but now is used to refer to chips with hundreds of millions of transistors. The semiconductor industry goal of shipping out quality devices continues to become more challenging as the design complexity increases and the feature size of transistors keeps decreasing. The purpose of fault diagnosis is to determine the cause of failure in a manufactured defective chip. To assist a designer or a failure analysis engineer, the diagnosis tool tries to identify the possible locations of the failure effectively and quickly. While many defects reside in the logic part of a chip, defects in scan chains have become more and more common recently as typically 30%-50% logic gates impact the operation of scan chains in a scan design. Logic diagnosis and scan chain diagnosis are the two main fields of diagnosis research. The quality of diagnosis directly impacts the time-to-market and the total product cost. In volume diagnosis, diagnosis is performed on a large number of failing chips to find yield-limiting systematic defects and design issues. Due to the increasing difficulty of physical inspection of today's multi-layer deep sub-micron design and the increasing cost of the physical analysis process, diagnosis becomes a very important step in the process of silicon debugging, yield ramp-up and field return analysis.

The following three merits are usually used to evaluate the effectiveness of an advanced diagnosis tool:

- High diagnosis resolution: The number of reported candidate locations is defined as diagnosis resolution and it should be as small as possible. If the reported

candidate set size is too large, it will make physical failure analysis extremely difficult. The cost of time and resources for the failure analysis process would be high.

- High diagnosis accuracy: The set of candidate locations reported should be as close as possible to the set of real defects. Low accuracy wastes time and resources in the physical failure analysis process because the reported candidate set has low correlation with the real defects.
- High runtime efficiency: The speed of performing quality diagnosis should be high. This is important especially for volume diagnosis. With deep-submicron processes, especially 65nm design and below, systematic defects have turned to be dominant. In order to locate systematic defects, a large volume of failed chips need to be diagnosed and the diagnosis results need to be used for the statistical analysis. To diagnose a large number of failed chips in a reasonable time, the run time of the diagnosis must be relatively short.

The objective of our diagnosis research is to develop techniques to improve diagnosis resolution and accuracy as well as improve diagnosis runtime performance. In Chapter II, we will briefly review previous works on defect diagnosis and in Chapter III we propose a technique to improve compressed test pattern generation for multiple scan chain failures. Chapter IV will discuss new methods to improve diagnostic test generation for scan chain failures using multi-cycle scan patterns. In Chapter V we propose a diagnosis method for multiple random physical defects residing in the logic part of a circuit using logic fault modes and multiple clustered defects. Chapter VI concludes the research.

CHAPTER 2. REVIEW OF DEFECT DIAGNOSIS ALGORITHMS

In this chapter, we review previous research in the area of defect diagnosis. In Section 2.1 we first review previous works on scan chain diagnosis and in Section 2.2 we review previous works on logic diagnosis.

Physical defects can behave in many different ways. We use fault models to model the effect of a defect for diagnosis. Currently, logical fault models are widely used due to the speed of simulation and simplicity. A logic fault model describes the faulty behavior of a defect at the logic level. Model-based defect diagnosis is a procedure to identity defects by using fault model simulations. Classic fault models include: the stuck-at fault model, the bridge fault model, the open fault model, the gate delay fault model and the path delay fault model.

- The stuck-at fault model: This is the simplest and most widely used model. The faulty behavior of a large number of physical defects can be effectively described using stuck-at fault models. For the stuck-at fault model, a node in the circuit is stuck at a fixed logic value, either 0 (stuck-at 0) or 1 (stuck-at 1). Stuck-at 0 could be the result of a short to the ground line. Stuck-at 1 could be the result of a short to the power supply line.
- The bridge fault model: The bridge fault model is used to describe the logic behavior of two nodes that are shorted in the circuit. Common bridge fault models are the wired-AND/OR fault model, the dominate fault model, 4-way bridge fault model. The wired-AND/OR bridge model assumes that the faulty node of the bridge always has the logic value 0(1). The dominate bridge model assumes that one node of the bridge always dominates the other node by imposing its logic

value on the other one. The bridge fault model is an important model since bridging is a common defect in circuits.

- The open fault model: The open fault model attempts to model the open defects, such as electrical open, break, and disconnected vias in a circuit. Opens can result in state-holding, intermittent, and pattern-dependent fault effects; thus open models are more complicated than stuck-at and bridge fault models.
- The delay fault model: To represent timing related defects, the gate delay model and path delay model are used. The gate delay model assumes the defect-induced delay is only between a single gate input and output. The path delay model spreads the total delay along a circuit path from a circuit input to a circuit output.

2.1 Review of Scan Chain Diagnosis

Scan-based designs have been considered a cost-effective method for achieving good test coverage in digital circuits and are widely adopted in nowadays VLSI designs. The amount of die area consumed by scan elements, chain connections, and control circuitry varies with different designs. As 30%-50% of logic gates on a typical chip impact the operation of scan chains [Kundu 94], scan chain failures are the cause of a substantial proportion of failing chips. Accurate diagnosis of scan chain failures is important for physical failure analysis and yield analysis.

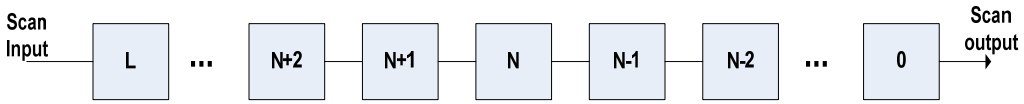


Figure 1: An Example Scan

We give the following terminologies here since they are used in much of the research on scan chain diagnosis. Each scan cell in a scan chain is given an index. As shown in Figure 1, the scan cell connected to the scan chain output has index 0 and the scan cells are numbered incrementally from scan chain output to scan chain input. Downstream scan cells of a scan cell N are the scan cells between scan cell N and the scan chain output. Upstream scan cells of a scan cell N are the scan cells between scan cell N and the scan chain input. In a list of consecutive scan cells in a scan chain, the scan cell with the highest index is called the upper bound (UB) and the scan cell with the lowest index is called the lower bound (LB).

Table 1: Two-pattern Flush Tests

Pattern	Pattern 1	Pattern 2
Scan Input (SI)	11110000	00001111
Expected Scan Output (SO)	11110000	00001111
Faulty SO (SA0)	00000000	00000000
Faulty SO (SA1)	11111111	11111111
Faulty SO (STR)	11100000	00001111
Faulty SO (STF)	11110000	00011111
Faulty SO (FTR)	11111000	00001111
Faulty SO (FTF)	11110000	00000111
Faulty SO (HT)	11111000	00000111

A chain pattern (also called a flush pattern) is a pattern consisting of shift-in and shift-out operations without pulsing capture clocks. The purpose of chain patterns is to

test scan chain integrity. A scan pattern is a pattern consisting of a shift-in operation, one or multiple capture clock cycles, and a shift-out operation. The purpose of a scan patterns is to test system logic. Classic scan chain fault models include stuck-at faults (stuck-at-0, stuck-at-1), slow faults (slow-to-rise, slow-to-fall, and slow), and fast faults (fast-to-rise, fast-to-fall, and fast). Setup-time violations can result in slow faults, and hold-time violations can result in fast faults. Slow and fast faults are also called timing faults. The faulty behavior of a physical defect can also behave as a permanent fault, which occurs in all cycles, or an intermittent fault, which occurs in a subset of cycles.

Usually flush tests are used to detect scan cell defects and to determine the fault type in a defective scan chain. Two flush test patterns [Chuang 08] shown in the second row of Table 1 can be used to determine the following fault types: stuck-at-0 (SA0), stuck-at-1(SA1), slow-to-rise(STR), slow-to-fall(STF), fast-to-rise(FTR), fast-to-fall(FTF), hold-time(HT).

The flush tests are shifted in and out of scan chains without using capture cycles. Assuming that there is a single fault in a faulty scan chain, the response to flush tests (the shifted-out values) for seven types of faults are shown in rows 4 through 10 of Table 1 and the fault free response is shown in row 3. It can be seen that the fault responses distinguish the fault types. After determining the fault type based on flush tests, the chain diagnosis is to locate the defective scan cell in the faulty scan chain.

Chain patterns alone are usually sufficient to determine the faulty chain and faulty type as described above but insufficient to pinpoint the index of a failing scan cell. This is the fundamental motivation of scan chain diagnosis, which is the process of identifying one or multiple defective scan cells in one or multiple scan chains or defective scan-

enable or clock signals. Diagnosis techniques of scan chain failures have been investigated in some previous works. These techniques can be classified into three categories: tester-based techniques, hardware-based techniques and software-based techniques. Next we review these techniques. We will focus on the software-based techniques as our research on scan chain diagnosis also falls into this category.

2.1.1 Tester-Based Scan Chain Diagnosis

Tester-based diagnosis techniques use a tester to control scan chain shift operations and use physical failure analysis (PFA) equipment to observe defective responses at different locations in order to identify failing scan cells. These techniques normally provide good diagnosis resolution. However, they require expensive, time-consuming, and often destructive sample preparation, and they provide visibility only through a small peephole. Hence, you must know where to look with your PFA equipment.

In [De 95] the authors propose a tester-based technique in which they apply a chain pattern of alternating 0s and 1s and use an electron-beam to detect the toggles. A binary-search scheme is applied to detect a stuck-at fault at a cell where the toggles start to disappear.

Song et al. propose a diagnostic method based on light emission due to off-state leakage current (LEOSLC) [Song 04]. They apply two chain patterns: all 0s and all 1s. They compare two emission images of a cell for both chain patterns. If there is no difference, a stuck-at fault could be on this cell or its upstream cells. This procedure is repeated until it reaches the first cell that shows a different emission image for all 0s and 1s. Applying a binary search can speed up the process.

If passing or failing conditions for a scan shift operation, such as power supply, reference voltages, or clock speed can be identified, then passing or failing conditions can be used to shift in a chain pattern and change the test environment to the opposite condition for shift out. The location where failures start to appear or disappear is the defect location. Motika et al. identify the passing or failing shift speed to diagnose slow faults [Motika 03]. By varying operating parameters, Motika, Nigh, and Song trigger one or more latches downstream from the fault location to change the state from the stuck-at fault value [Motika 06].

2.1.2 Hardware-Based Scan Chain Diagnosis

Hardware-based methods use special scan chain and scan cell designs to facilitate diagnosis. These techniques are effective in isolating scan chain defects. However, because they typically require extra hardware overhead, they are not acceptable in many products. In addition defects could occur in the extra control hardware, which makes diagnosis more complicated.

In [Schafer 92] the authors proposed to connect each scan cell's output to a scan cell in another scan chain so that its value could be observed from the other scan chain (partner chain) in diagnostic mode. For example, assume there is one stuck-at-0 at the output of cell 2 of scan chain 1, and chain 1 has four scan cells. After shifting in 1111, chain 1 should have 1100. Then the circuit is turned into diagnostic mode, and the data in chain 1 is transferred to its partner chain. Assuming the partner chain is a good chain, 1100 is observed from this chain, and it can be deduced that the defect must be in the middle of chain 1.

In [Edirisooriya 95] the authors propose to insert XOR gates between scan cells to enhance chain diagnosis. The proposed scheme will always identify the fault closest to the scan output if there are multiple faults. The scheme makes a trade-off between the number of XOR gates added and the diagnostic resolution.

Narayanan and Das proposed to add simple circuitry to a scan flip-flop to enable its scan-out port to be either set or reset [Narayanan 97], [Narayanan 99]. They presented a global strategy based on the set/reset feature to account for disparities in defect probabilities and controllability and observability attributes of flip-flops in a scan chain. They also presented an algorithm to optimally modify a subset of the flip-flops to maximize diagnostic resolution. One solution is that each adjacent pair of flip-flops consists of a flip-flop whose scan output can be reset to 0, and a flip-flop whose scan output can be set to 1. Hence, any single stuck-at fault can be diagnosed down to a pair of flip-flops.

In [Wu 98] a special circuit is proposed to flip, set, or reset scan cells to identify defective cells. After shifting in a chain pattern, the circuit can invert, set, or reset each flip-flop's state. The faulty cell is located via the observed unloading value. Song proposed a bidirectional scan chain architecture in which the scan chain performs both forward and backward scan shifts to diagnose scan faults [Song 00].

Tekumulla and Lee propose partitioning scan chains into segments and bypassing segments that contain hold-time violations [Tekumulla 07]. When a hold-time violation is located on a scan chain segment, the flip-flop in that segment is bypassed and new test patterns are created.

2.1.3 Software-Based Scan Chain Diagnosis

Software-based techniques use diagnosis algorithms to identify faulty scan cells. Compared with hardware-based techniques, software-based techniques do not need modification of the conventional scan design and are more widely adopted in industry for general designs. Software-based fault diagnosis algorithms include cause-effect and effect-cause methodologies. Effect-cause diagnosis uses fault injection and simulation. The simulated responses are compared with the observed tester results to identify the faulty scan cells. This approach is useful when complex fault models are used. Cause-effect diagnosis is based on pre-calculated fault signature, which is also called fault dictionary.

2.1.3.1 Effect-Cause Methods

In [Stanley 01] fault injection and simulation are used to find faulty scan cells. One fault in a cell is injected for each run. Because all scan cells on a faulty chain are candidates, the method is time-consuming for a scan chain with many cells. To speed up the diagnosis procedure, researchers have proposed several techniques.

Guo and Venkataraman proposed an algorithm that identifies an upper bound (UB) and lower bound (LB) for a faulty cell [Guo 01]. Figure 2 shows an example to explain the algorithm. Assume there is a single fault in the faulty chain. First, they faulty chain's simulated loading values are all changed to Xs. After the capture clock pulses, assume the simulated captured values on this faulty chain are XX10XXX0XX1X. That means cells 8 and 4 will capture 0s no matter what values were loaded to the faulty chains. Suppose the observed values on the tester are actually 111111001010. Because the observed value at scan cell 8 is 1, a stuck-at-1 fault must be downstream of cell 8. Thus cell 8 is an upper

bound (UB) of the faulty scan cell. Meanwhile, because the observed value at cell 4 matches the simulated value 0, the stuck-at-1 fault must be upstream of cell 4. Thus cell 4 is a lower bound (LB) of the faulty cell. Ranking the suspect cells within the bounded range further improves the diagnosis resolution. This method may not provide good-enough diagnostic resolution for some scan chain failures and it is hard for it to handle multiple faults in one scan chain.

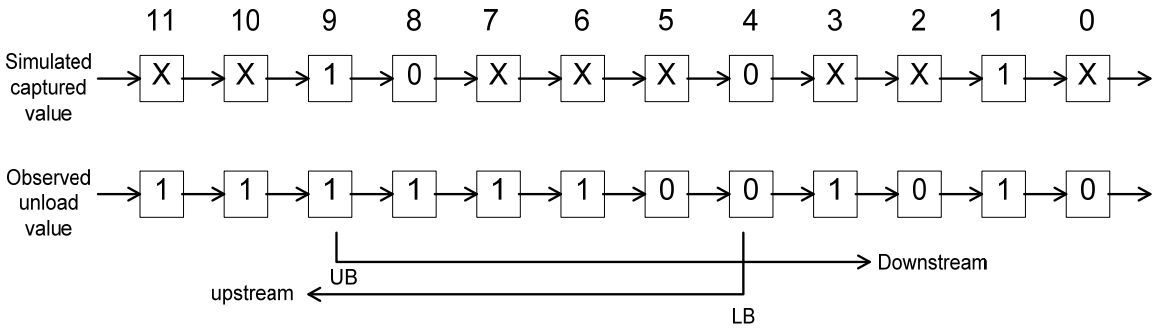


Figure 2: Identification of upper bound (UB) and low bound (LB)

Figure 3 illustrates another method of speeding up effect-cause scan chain diagnosis. Kao, Chuang, and Li proposed jump simulation to diagnose a single chain fault [Kao 06]. For each failing pattern, a simulator performs multiple simulations to quickly determine the UB or LB. After the range is finalized, a detailed simulator performs parallel pattern simulation for every fault in the final range. Suppose there is a stuck-at-1 fault on a scan chain and the current UB=27 and the current LB=20. The scan cells from the UB to the LB are evenly divided into three parts and the boundary scan cells (22, 24, and 26) are chosen as jump bits. In searching for a new UB, the algorithm assumes the fault is upstream from the jump bit. It changes all 0s downstream from the jump bit to 1s, and all

0s between the jump bit and the UB to Xs. If a simulation mismatch occurs in a jump bit, say, cell 24, the algorithm deduces that the stuck-at-1 fault is actually downstream from jump bit 24. It therefore moves the new UB to scan cell 23. It searches for the LB in a similar way. This method can provide better diagnostic resolution compared with method proposed in [Guo 01]. However, the diagnostic resolution is still bounded by the production test set and may not be able to provide good-enough resolution needed for failure analysis.

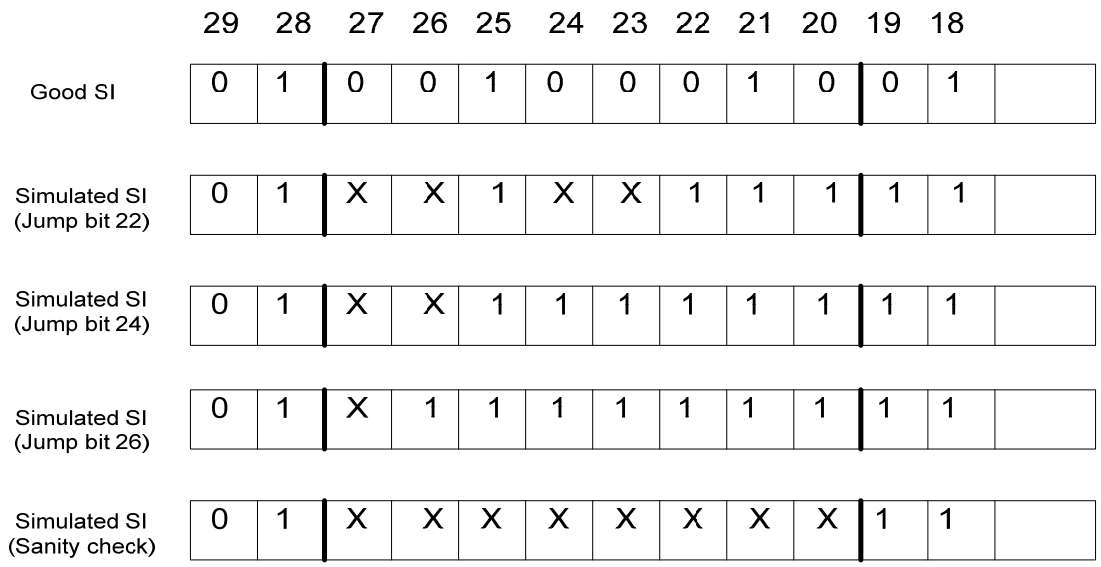


Figure 3: Jump Simulation

In [Huang 07] an effect-cause method was proposed using dynamic learning. This method was based on several learning rules. These rules analyzed the circuit, patterns, and mismatched bits and backtraced the logic cones to determine which cells should be simulated in the next iteration. As a result, only a few cells need to be simulated to find suspects instead of simulating every scan cell within a range. Figure 4 shows an example

of a technique that updates the LB. A fault is injected at the current LB at cell 1. If there is a simulation mismatch on the cell of a good chain, we can backtrace the fault from the mismatched cell. Assuming this cell is driven by cells 4 and 3 on the faulty chain, we learn that either cell 4 or cell 3 or both carried wrong loading values in the previous simulation. Therefore, the new LB is updated to scan cell 3. This process can be iterated several times until the actual defective scan cell is found.

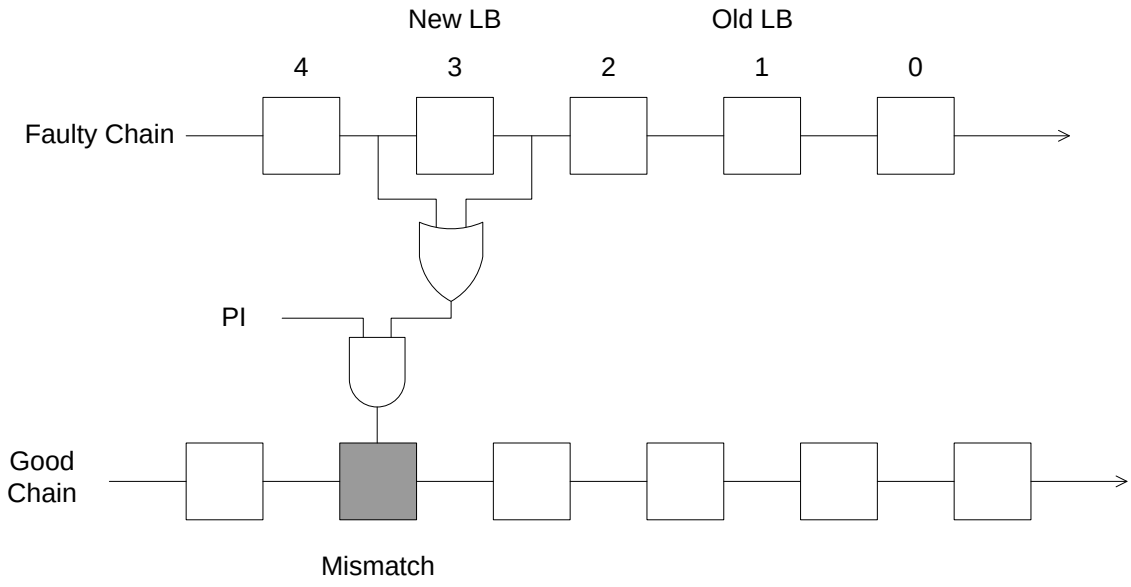


Figure 4: Dynamic learning and single-excitation ATPG

Huang et al. investigate the diagnosis of intermittent hold-time faults and propose an algorithm based on X simulation in which intermittent loading and unloading behavior is modeled with Xs [Huang 03]. Huang, Cheng, and Crowell present case studies to illustrate the problems of using a fault model to diagnose real chain defects [Huang

05_1]. They propose a fault model relaxation flow in which chain fault models are adaptively selected according to fault model relaxation rules and simulation results.

To reduce test data volume, encoded tests and compacted test responses are now widely used in industry. Scan chain diagnosis on devices with embedded compression techniques is a new challenge as the errors in responses due to defects which are captured in scan cells are not observed directly. Huang, Cheng, and Rajski proposed a methodology that enables existing chain diagnosis algorithms to be used with compressed test data [Huang 05_2]. The tester's storage capacity and test time restrict the total number of failing bits that can be collected, thus negatively affecting the diagnosis resolution. Huang et al. proposed three methods of running chain diagnosis with limited failure information: static pattern reordering, dynamic pattern reordering, and per-pin-based diagnosis [Huang 06].

2.1.3.2 Cause-Effect (Dictionary-based) Methods

In [Guo 07] a dictionary-based diagnosis technique for scan chain failures is proposed. In this technique, differential signatures are stored in fault dictionaries to reduce the fault signature redundancy of adjacent scan cell faults. The differential signatures serve to diagnose single stuck-at faults, timing faults and some multiple stuck-at faults in a single scan chain. The cost of large memory and more complicated fault models could make the method less practical.

2.1.3.3 Diagnostic ATPG for Scan Chain Failures

In software-based diagnosis, usually the production test patterns used to detect defective chips are used to collect test failures for diagnosis. Since the aim of production tests is only fault detection, they may not provide sufficient information for diagnosis

purpose, leading to poor diagnosis. In [Huang 06], it is reported that for only 23% of failing dies, diagnosis based on production tests gave less than or equal to three scan cell suspects, which is regarded as the minimum desirable resolution in our research work. Methods to generate additional test patterns to improve diagnostic resolution have been proposed.

Kundu proposed a scan chain diagnosis algorithm that focuses on generating test patterns for stuck-at faults [Kundu 93] [Kundu 94]. It creates test patterns either to capture desired values in specific scan cells or to propagate fault effects to good scan chains for failure observation. The method is summarized as follows:

For each scan cell N in faulty chain (from cell L-1 to cell 0)

1. Add scan cell constraints (N, N-1…0) = (vv...v)

2. Run stuck-at ATPG to generate a test pattern for a stuck-at-v fault at the data input of scan cell N

3. If the fault is detected, save the test for cell N and continue with next cell at Step 1;

4. Otherwise, generate a test for the stuck-at-v fault on data output of scan cell N:

a. If a fault is detected, save the test for cell N and continue with the next cell at Step 1;

b. Otherwise, continue with the next cell from Step 1

This diagnostic test generation algorithm targets each scan cell one at a time starting with cell (L-1), which is the scan cell closest to scan chain input. For each target scan cell N, it constrains scan cell N and its downstream scan cells to value v. These constraints model the impact of a stuck-at fault on scan cell N during scan load process (i.e., scan cell N and all its downstream scan cells will get value v during the scan load process due

to the stuck-at-v fault on cell N). With all these constraints, the algorithm first tries to create a test pattern to capture a logic value opposite to the stuck-at fault value into the target scan cell N. If successful, it moves on to the next scan cell. Otherwise, it tries to create a test pattern to detect a stuck-at-v fault at the data output of the target scan cell. However, this algorithm doesn't constrain the scan cell appropriately to provide sufficient conditions to guarantee that the scan cell N can be differentiated from scan cell N-1.

In [Yang 05] the authors proposed using functional test patterns for scan chain failure diagnosis. Their procedure selected patterns to randomize the signal probability of scan cells. By comparing the observed signal profile on a tester and the expected signal profile along a faulty scan chain, the faulty scan cell can be identified. Profiling based methods can quickly identify some faulty scan cells, but it may fail to identify some faulty scan cells which can be easily pinpointed using other methods.

Some researchers proposed diagnosis algorithms that include two parts [Hsu 06] [Tzeng 07_1] [Tzeng 07_2]. First, use diagnostic ATPG to obtain scan patterns that don't use chain-loading procedures so that the impacts of chain defects come only from chain unloading procedures. Then, apply heuristics to analyze test failures and identify defective scan cells. The heuristics include signal profiling, best alignment, delay insertion, and image recovering.

Li [Li 05_1] [Li 05_2] proposes a single-excitation technique to generate diagnostic patterns. Single-excitation patterns have one sensitive bit that can be flipped by the fault. This technique converts the diagnosis problem into a single-stuck-at-fault ATPG problem, which existing tools can easily solve. Figure 5 [Li 05_1] shows an example. Suppose that a stuck-at-0 chain fault exists. The single-excitation pattern 00100 shifts

into the faulty scan chain, making cell 2 the sensitive bit. Hence, this technique detects a fault in the same way as it would detect a stuck-at-0 fault in combinational logic.

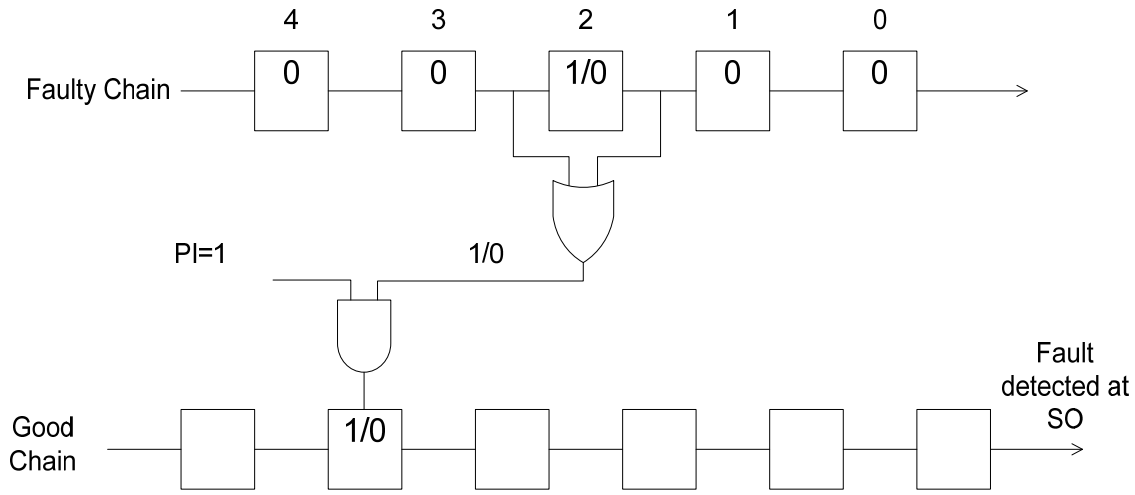


Figure 5: Fault Detected at SO of Good Chain

In [Crouch 05] it is suggested that fault effects be propagated to as many primary outputs and good scan chains as possible. He also proposed adding shift cycles between capture clocks, which can be helpful for diagnosing multiple chain faults. In [Sinanoglu 07] Sinanoglu and Schremmer proposed generating test stimuli, such as 0s and 1s, that are immune to hold-time violations on the faulty chain and randomly changing stimuli on the good chains.

Guo, Huang and Cheng proposed a complete test set generation technique for scan chain fault diagnosis [Guo 07]. This technique attempts to create test patterns that uniquely identify any faulty scan cell. The authors extended the algorithm to handle multiple failing scan chains and designs with test compression logic. During test

generation, the algorithm carefully analyzed constraints on scan cell controllability and obsevability if there are logic correlations between scan cells on the same scan chain. If all the scan cells in the target range can be targeted using the procedure, a complete test set can be obtained to guarantee each scan cell will have a unique faulty behavior. This method, however, cannot provide sufficient conditions for multi-cycle scan patterns, and the test generation flow can be very time consuming when the whole scan chain needs to be targeted, especially for very large designs.

In [Wang 08], a diagnostic test generation method is proposed to target scan chain defects in the presence of defects in the logic part of the DUT. It targets a scan cell using only fault propagation conditions, (i.e., specifically targeting scan cell input is not used).

2.2 Review of Logic Diagnosis

In this section, we give a brief review of diagnosis methods of defects in the logic part of circuits.

2.2.1 Single Stuck-At Defect Diagnosis Using the Stuck-At Fault Model

The stuck-at fault model is most widely used to model the faulty behavior of physical defects, and many defects can be covered using stuck-at fault ATPG patterns. Most diagnosis is also based on stuck-at fault model. In most cases, we don't know what model the defect belongs to before we perform diagnosis. Therefore stuck-at diagnosis is usually performed first. Based on the stuck-at diagnosis result, we can determine if complex fault models, such as the bridge fault, the net open fault, or the cell fault etc., can be constructed to better explain the failures.

Logic diagnosis can also be classified into two categories: One is Effect-cause diagnosis that back traces from erroneous PO/PPO to find candidates and then simulates the candidates to find the one that best matches the failure responses observed on the tester. The other is Cause-effect diagnosis, which uses a pre-simulated fault dictionary and the observed failure response on the tester to determine the defect in a faulty device.

2.2.1.1 Cause-Effect Diagnosis

A fault dictionary is a record of the errors that the modeled faults in the circuit are expected to cause [Abramovici 90]. It stores a mapping from the modeled fault to a simulated faulty response. The procedure of fault dictionary diagnosis is to look in the pre-calculated dictionary to find the suspect that causes the faulty behavior. The fault candidate whose expected faulty signature matches best with the observed faulty signature will be chosen as the final fault candidate. If we assume a single stuck-at defect, there should be an exact match between the expected signature of the fault candidate and the observed faulty signature.

There are three types of dictionaries: pass-fail dictionaries, complete dictionaries and compressed signature dictionaries. A pass-fail dictionary only stores a single bit (pass or fail) of failure information for each fault per test pattern. Since it omits useful information about where the failing bit is, it renders distinguishing some faults impossible. A complete dictionary is a full-response dictionary, which stores all circuit outputs in the presence of each fault for each test pattern. The number of bits required to store a complete dictionary equals F*V*O, where F is the number of faults, V is the number of test patterns, and O is the number of primary outputs. The downside of a complete dictionary is the storage it requires is huge for designs with multi-million gates.

A compressed signature dictionary is obtained by feeding the output information through a 32 or a 64 bit multiple input signature register (MISR) to get a compressed signature. There is the problem of aliasing: that two different output responses may be compressed to the same failure signature. But by choosing a MISR with more bits, the chance of aliasing is slim. Compressed signature dictionaries save storage space and provide about the same diagnosis resolution as the complete dictionary.

A number of methods were proposed in order to reduce the memory requirement for a complete dictionary.

2.2.1.2 Effect-Cause Diagnosis

In [Waicukauski 89] Waicukauski and Lindbloom proposed an effect-cause diagnosis procedure. The procedure can be summarized as following.

1. Use the path-tracing technique to obtain an initial candidate list. Initial fault candidates must satisfy the requirements below in order to reduce the search space and improve diagnosis efficiency:
    - The fault must reside in the input cone of a failing PO of the given pattern
    - There must exist a parity-consistent path from the faulty site to the failing PO
    - If a failing pattern affects more than one PO, that candidate fault must reside in the intersection of all the input cones of those failing POs. This is based on a single defect assumption: that for a failing pattern, only one defect is activated and propagated.
2. Simulate each fault on the initial candidate list to see if it perfectly explains any of the failing patterns. If it does, assign it a weight equal to the number of patterns it

explains on the current list. Store the candidate fault with the greatest weight, and remove the failing pattern explained by it.

3. After explaining the entire failing-pattern list, or when the candidate list has all been examined, terminate the algorithm, and report the possible candidate sites. Rank candidates using their weights obtained in Step 2.

2.2.2 Multiple Fault Diagnosis

As design complexity grows and the feature size of transistor decreases, more and more physical defects cannot be modeled using the classic single fault model. These defects behave as multiple faults. Much research has been done on multiple fault diagnosis. Multiple-fault diagnosis mainly has the following difficulties:

If the multiple-fault problem is addressed directly, the error space grows exponentially. Error space = (# of lines) $^{(\#\ \text{of defects(errors)})}$ [Veneris 99], where # of lines is the number of signal lines in the circuit. This would be expensive to explore exhaustively. Assumptions and heuristics are proposed to address this problem. Multiple-faults may be activated at the same time and interact with each other to create fault masking, which makes the multiple fault diagnosis difficult.

2.2.2.1 Diagnosis Using SLAT Paradigm [Bartestein 01]

In [Bartestein 01] a diagnosis method based on Single-Location-at-a-Time (SLAT) was investigated. The single-location-at-a-time paradigm assumes that there exist many SLAT patterns, in which, only one activated fault is observed. There can be more than one activated fault, but only one is propagated to the primary output or scan flip-flops. The algorithm is actually location-based diagnosis. During the first phase, it performs logic diagnosis with stuck-at faults assumption. It back-traces the failing pins to

find candidates and if a candidate explains a pattern, the pairs consisting of the failing pattern and the locations of the successful faults are stored in a table. It collapses two faults into one fault if two faults happen at the same location but with opposite faulty values. If two faults occur at the same location, there is a high possibility that the two faults are caused by the same defect.

In the second phase, a recursive minimum set covering procedure is called to find the minimum number of faults which can explain all the failing SLAT patterns. The failing patterns that are explained by the least number of faults are considered first. The result is a multiplet, which is a minimal collection of faults such that each SLAT pattern is explained by at least one fault in the multiplet. A multiplet is the basic unit of a SLAT diagnosis. It is the simplest explanation of all the SLAT patterns.

2.2.2.2 Diagnosis Methodology Using Error Propagation Analysis [Yu 08_1] [Yu 08_2]

The proposed flow is shown in Figure 6 [Yu 08_2] below.

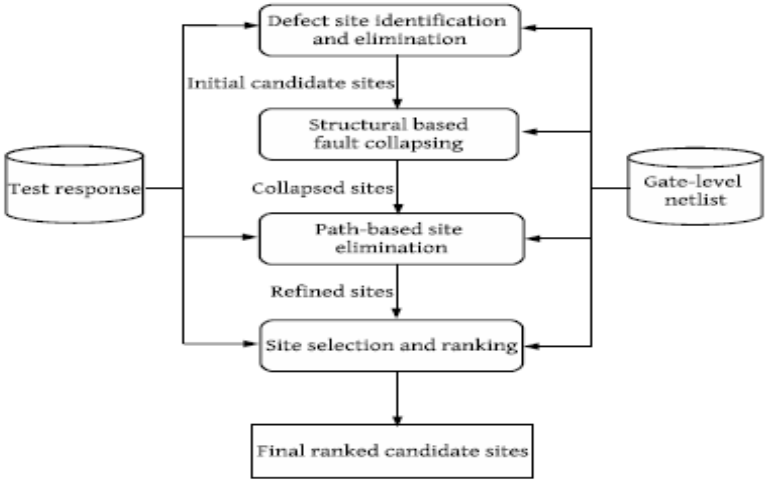


Figure 6: Proposed Flow

Three definitions:

- For pattern t_k, if site f_i propagates to observation point out_j if all the side inputs of on-path gates have fault-free values, f_i is said to "output-explain" out_j for t_k.
- For pattern t_k, if site f_i propagates to out_j only if some side inputs of on-path gates have faulty values, f_i is said to "partially output-explain" out_j for t_k.
- Let $f_i - J_i$ be the longest sub-path such that f_i can propagate to J_i when all the side inputs of on-path gates have fault-free values. The remaining sub-path $J_i - out_j$ is called the "under-explained" sub-path

Fault elimination:

Perform conservative implications and eliminate false faults using passing bits.

Site grouping:

- For each failing pattern and each defect site that explains any observed failing observation points, we find the "out-explained" out_j and also the "under-explained" sub-path for each site.
- Group the sites that have the same "out-explained" observation points and the same "under-explained" sub-paths together

Site selection:

- Each time, select the site set that output-explains the most observed failing points that have not been explained

The site selection algorithm is shown in Figure 7 below [Yu 08_2].

Algorithm 2 Site Selection.

S_i = {site sets from site grouping}
S_o = {the selected sites}
FO = {failing observable points *not* output-explained by S_o}
Initialize $S_o = \emptyset$ and $FO =$ all failing observable points
while $FO \neq \emptyset$ **do**
  Select set $F \in S_i$ that output-explains the most points $\in FO$
  $S_o = S_o \cup F$
  $FO = FO -$ {points jointly output-explained by sites in S_o}
end while

Figure 7: Site Selection Algorithm

Ranking:

- Rule 1: A site that is an element of a smaller site set is ranked higher
- Rule 2: If rule 1 results in a tie, a site that output-explains more failing points is ranked higher
- Rule 3: If rule 2 results in a tie, randomly choose one site

This method has its downside. The fault elimination step may not be able to eliminate many sites, which leaves a large number of sites for the following analysis.

2.2.2.3 Multiple error diagnosis based on Xlists

In [Boppana 99], the authors assume the logical errors are locally bounded. They use Xlist to mark a region with X (don't care) and perform a 3-value simulation (0,1,X) to see if X is propagated to the output. If there is mismatch at the output, this region cannot contain all the faults. This method has good computation at speed and general good resolution when the faults are in clusters. However, in real designs, faults may scatter and are not related. Using this method will not be effective to localize the fault locations.

Xlist: A set of nodes whose actual values would be replaced during simulation by the value X, and the X-values propagated to subsequent logic levels by 3-valued logic simulation is called an Xlist.

Two forms of Xlist-Error models are defined. They are topologically bounded errors and region based error. Let the circuit have n nodes. Let T = (1,2,…,n) be a topological order of the nodes of the circuit.

Topologically bounded error: If the logic functions at the nodes in set E = {e_1, e_2, …, e_k} are erroneous and satisfy the following: Exist i, j, (1 <= i <= j <= n) such that {i<=e_q<=j, for any q, 1 <= q <= k}. The integers i, j are the lower and upper bounds within which the error lies.

Region based error: If the logic functions at the nodes in set E = {e_1, e_2, …, e_k} are erroneous and satisfy the following: For any q, 1 <= q <= k, Structural Distance (e_q, p)<=r.

When diagnosing topologically bounded errors, if the error is assumed to be bounded by k topologically, then choosing overlapping Xlists will guarantee that there exists an Xlist containing all the error nodes: (1,2,…,2k), (k+1,k+2,…,3k),(2k+1,…,4k),…((roof(n/k)-2)k+1,…,n). The problem is again if the faults are scattered, then no diagnosis is possible.

When diagnosing region-based errors, if the errors are assumed to be bounded by a radius r, by choosing a region-based Xlist at each node in the circuit that includes every node within a radius of r from that node will be guaranteed to contain the fault region.

The Xlists are simulated and compared with observed PO and PPO. If an Xlist produces a mismatch{(0,1) or (1,0)} (match{(0,0) or (1,1)}, or partial match {(X,0) or

(0,X)}, the potential of that Xlist containing the error nodes is reduced (increased, increased slightly), and the Xlist is scored accordingly. Xlists are ranked according to the scores.

Symbolic variables can also be used to improve the accuracy of the diagnosis procedure. Binary decision diagram (BDD) representation of the symbolic function simulated removes the losses in 3-valued simulation. However, there are circuits for which efficient BDD representation is hard to obtain.

2.2.2.4 Multiple Fault Diagnosis Using n-Detection Tests

This work [Wang 03_2] investigates the multiple-fault diagnosability of the n-detection tests.

- For bridge faults, n-detection tests increase the possibility of detecting them.
- If P_a is the test set that detect fault a and P_b is the test set that detect fault b, n-detection tests decrease the possibility that P_a=P_b or P_a is a subset of P_b, which is hard for multiple diagnosis.

Two diagnosis methods in [Bartestein 01] and [Wang 06] are used to demonstrate the improvement of multiple-fault diagnosability using n-detection tests.

2.2.2.5 Curable Vectors and Curable Outputs

In [Huang 01], the author proposed a diagnosis procedure with measures of two matching mechanisms: curable vectors (vector match) and curable outputs (failing PO match). Each possible candidate error is ranked according to these two matches. When single defect diagnosis does not explain the behavior, double defect diagnosis is considered and a heuristic for multiple (more than two) defect diagnosis is proposed.

Curable output: Assume that the response of a primary output Z_i is failing with respect to an input vector v. It is called a curable output of a signal f with respect to test vector v if v is able to sensitize a discrepancy path from f to Z_i. By notation, Z_i belongs to curable output (f,v).

This is essentially a fault injected at f that matches the failing output Z_i for vector v. A curable output based heuristic is outlined as follows: First, the curable outputs of each signal with respect to each failing input vector are calculated, either by fault simulation, back propagation [Kuehlmann94], or an observability measure [Veneris97]. The signals are sorted based on their total curable outputs for all the failing vectors. The signal with a large number of curable outputs is regarded to be a more likely defect candidate.

Curable vector: An input vector v is curable by a signal f if the output response can be fixed by replacing f with a new Boolean function (re-synthesize). Partially curable vector: An input vector is partially curable by a signal f if the response of every failing output reachable by f can be fixed by re-synthesizing f.

The diagnosis procedure will first assume only a single defect exists. If there are signals that pass the curable output and curable vector filtering, then the process stops. Otherwise, double faults are assumed and every signal pair is enumerated to check against the curable vector based criterion. In [Huang 99] the procedure of double-defect diagnosis is investigated. If any valid candidate is found, then the process stops. Otherwise, it moves on to the next stage to apply a heuristic to generate a signal list sorted by their defect possibilities. The heuristic algorithm uses 2 rules: rule 1 records the total number of partially curable vectors, while the rule 2 records the total number of

curable outputs. The signals are first sorted according to rule 1. If rule 1 results in a tie, the signals are sorted according to rule 2.

2.2.2.6 Multiple and single fault simulation in diagnosis

[Takahashi02]

The authors use four heuristics to repeatedly add and remove faults into/from the candidate list. At the beginning, all collapsed stuck-at faults are in the candidate list.

In this work the authors have the following assumptions:

1. Multiple faults do not mask each other.
2. All single stuck-at faults detected by a failing test are candidate faults for inclusion in the set suspect faults (SF).
3. Faults from SF that cause inconsistency between the simulated value of the circuit injected with these faults and the observed value in response to a passing test must be removed from SF.
4. A smaller fault set is more probable than a larger fault set

The following four heuristics are applied in the following order to add faults into the candidate list and remove faults from the candidate list:

1. If a fault is detected by a passing test, then the fault is removed from SF.
2. A failing pattern t_f is simulated on the circuit injected with all the faults in the candidate list. If there is mismatched output (i.e., the observed value is different from the simulation) then all single faults detected at any mismatched output are added into SF.

3. A passing pattern is simulated on the circuit injected with all the faults in the candidate list. All single faults detected by passing patterns at mismatched outputs are removed from set SF.
4. Try removing any single fault if the reduced SF without this fault still matches the observed faulty response.

This method requires too many multiple fault simulations and is hard to handle more complicated fault models that may only excite under certain conditions.

2.2.2.7 Incremental diagnosis and PO partition [Wang03]

[Wang06]

A fault is defined as a hard fault when only one pattern detects it. The only pattern that detects a hard fault is called the essential pattern of this fault. Suppose n faults can perfectly explain some failing patterns. These n faults as a group are called the n-perfect candidate for those explained patterns.

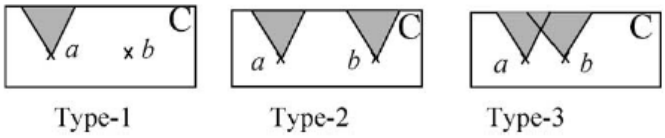


Figure 8: Multiple Fault Pattern Types

Failing pattern types:

- Type-1: SLAT pattern.

- Type-2: More than one activated fault is observed. Their fault effects are not correlated. PO partition techniques can be used to deal with Type-2 failing patterns.
- Type-3: More than one activated fault is observed and their fault effects have interactions on some observation points. Type-3 failing patterns are hard to diagnose.

$$FC(g) = \frac{\sum_{i=1}^{M} \left(\sum_{j=1}^{N} \mathrm{Ob}_{ij} \right)}{M}.$$

Functional congestion (FC) of a gate g is defined as above. M is the number of faults in g's fan-in cone; N is the total number of patterns in test T. If the fault i under pattern j can be observed by g, Ob_{ij} is 1; otherwise 0. Gates with high functional congestion are often the cause of incomplete or wrong candidate fault sites, since multiple faults are more likely to interact at functional congestion locations. N-detection test sets will improve the diagnosis results since more patterns will be Type-1 or Type-2 than in the single detection test set.

The A_single Algorithm [Waicukauski89] is used as the base algorithm:

1) Initialize the fault-candidate list using a path-tracing technique. Initial fault candidates satisfy the following requirements to reduce the search space and improve diagnosis efficiency:
    a. The fault must reside in the input cone of a failing PO of the given pattern.
    b. There must exist a parity-consistent path from the faulty site to the failing PO.

c. If a failing pattern affects more than one PO, that candidate fault must reside in the intersection of all the input cones of those failing POs (SLAT assumption).

2) Simulate each fault on the initial candidate list to see if it perfectly explains any of the failing patterns. If it does, assign it a weight equal to the number of patterns it explains on the current list. Store the candidate fault with the greatest weight, and remove the failing pattern explained by it.
3) After explaining the entire failing-pattern list, or when the candidate lists have all been examined, terminate the algorithm, and report the possible candidate sites. Sort candidate faults using their weights obtained in Step 2.

n-perfect Algorithm:

1) Find a 1-perfect fault candidate: n=1. Apply A_single. Eliminate the explained patterns.
2) Inject each n-perfect candidate into the circuit and perform steps 3 and 4 until all n-perfect candidates have been tried.
3) For each unexplained failing pattern, initialize the possible fault candidates.
4) Perform A_single on the modified circuit and construct (n+1)-perfect candidates based on the targeted fault model.
5) Determine the (n+1)-perfect candidates that can further explain some failing patterns not yet explained by those (1 through n)-perfect candidates.

6) Rank and weight the (n+1)-perfect candidates based on failing and passing information. Eliminate those failing patterns that can be explained by (n+1)-perfect candidates from the failing pattern list. Increase n by 1.
7) Perform steps 2-6 for the remaining unexplained failing patterns until no fault candidate can be found, or until all failing patterns have been explained.
8) Post process all possible k-perfect candidates (1<=k<=n) to eliminate the candidates that cause many passing patterns to fail when multiple-fault candidate are injected into the circuit. This is to eliminate wrong candidates assumed in the very beginning, which may imply other wrong candidates.

Not all n-perfect candidates that explain all failing patterns will be found. In the final step, the algorithm will find the minimum cardinality group of faults as the fault candidates that can explain the most failure responses.

Failing PO partition algorithm:

1) Back trace from each failing PO. If back tracing from POi finds a possible fault candidate, we mark it as reachable from POi.
2) For each failing pattern Pi, create failing PO connectivity graph, gPi. Each PO corresponds to a vertex in gPi. If two failing POs can reach the same fault, there is an edge between them.
3) Collect gPi to form Gp. Gp has vertices corresponding to POs. There is an edge in Gp if there is an edge between these vertices in any of the gPi. Assign a weight to an edge in Gp equal to the number of corresponding gPi's that contain that edge. If Gp is a disjoint graph, the heuristic stops. Perform the diagnosis separately on each sub graph without losing any failing-pattern information.

4) Partition Gp by removing low-weight edges.
5) Use each group of failing POs induced by the partition to filter out the original failure responses, and generate the new failure responses.
6) Diagnose each group of POs separately and obtain the fault candidates.

This method must find the exact fault in the first run to guarantee further diagnosis, and it may be hard to inject the fault when the fault identified is a defect that can only be modeled using a complex fault model.

2.2.2.8 Adaptive Diagnostic Test Pattern Generation

In [Lin 07], two methods were proposed to generate diagnostic test patterns to improve diagnosis resolution. They are called SO-SLAT patterns and multiple-capture anti-detecting (MC-AD) patterns. The authors assume that the initial diagnosis report includes all the injected faults.

SO-SLAT patterns:

- For each fault in the candidate list, propagate the fault to one single observation point while inactivating all the other faults that can propagate to this single observation point.
- If it's impossible to generate a test for one fault, apply the test patterns that can be generated and try generating again. So it's an iterative process that requires access to the tester multiple times.

MC-AD patterns:

- For each faulty line in the candidate list, insert a multiplexer **(**MUX), the select line of MUX can set the line to faulty value or fault-free value.

- Apply MC-AD patterns and use Satisfiability **(**SAT) solver to decide the values at these candidate faulty lines to match tester outputs.
- MC property: It has multiple-cycle captures, so it can maximize the number of activated faults for each pattern.
- AD property: The AD test for fault f_i is test patterns that can detect all the other candidate faults but not f_i.

These methods need to access the tester multiple times, which makes them less practical to use in some environments. It is also difficult for these methods to handle intermittent faults.

2.2.2.9 Adaptive Debug and Diagnosis without Fault Dictionaries [Holst 07]

Holst and Wunderlich proposed a method to identify faulty regions based on its input/output behavior and independent of a fault model.

They define the evidence of a fault f for each test pattern t ϵ T.

$$e(f,t) = (\Delta\sigma_t, \Delta l_t, \Delta\tau_t, \Delta\gamma_t)$$

- $\Delta\sigma_t$ is the number of failing outputs where both the device under diagnosis (DUD) and the fault machine (FM) match. They are also usually called sftf (simulation fail tester fail).
- Δl is the number of outputs which fail in FM but are correct in DUD. They are also called sftp (simulation fail tester pass).
- $\Delta\tau_t$ is the number of outputs which fail in DUD but are correct in FM. They are also usually called sptf (simulation pass tester fail).

- $\Delta\gamma_t$ is the minimum of $\Delta\sigma_t$ and Δl.

So the evidence of a fault f for a test set T is

$$e(f,T) = (\sigma_t, l_t, \tau_t, \gamma_t), \text{ with}$$

$$\sigma_t = \sum_{t\in T}\Delta\sigma_t,\ l_t = \sum_{t\in T}\Delta l_t,\ \tau_t = \sum_{t\in T}\Delta\tau_t,\ \gamma_t = \sum_{t\in T}\Delta\gamma_t.$$

The collapsed stuck-at faults are ranked using the following criteria:

- First, evidences are ranked by increasing γ_t.
- Secondly, evidences with equal γ_t are ranked by decreasing σ_t.
- Finally evidences with equal γ_t and σ_t are ranked by increasing l_t.

It was proposed in this work that if the diagnostic resolution is not good enough, more test patterns can be generated based on the analysis of the evidences.

For multiple fault diagnosis, it is difficult for this method to provide sufficient results for further failure analysis since many false locations could be ranked before some real faulty locations that only produce few failing bits. It also fails to address the issues in the diagnosis of more complicated multiple defects, which could be derived from a group of simple stuck-at faults to better explain the failures.

2.2.2.10 A Diagnosis Algorithm for Extreme Space Compaction [Holst 09]

Holst and Wunderlich proposed a diagnosis algorithm which is especially suited for BIST with a test response compaction scheme. The idea is similar to what was proposed in [Holst 07]. For each fault, a signature in the form of an m-bit parity bit stream P_m^f is produced to compare with observed parity bits P_m. They use the following definitions:

- σ_m^f is the number of bit positions faulty both in the response stream P_m as well as in the syndrome P_m^f.
- l_m^f is the number of bit positions correct in P_m but faulty in P_m^f.
- τ_m^f is the number of bit positions faulty in P_m but correct in P_m^f.

The faults are ranked using the following criteria:

- First the faults are ranked by decreasing σ_m^f.
- Then the faults with equal σ_m^f are ranked by increasing l_m^f.

For diagnosis of multiple faults, this method has the same deficiencies as pointed out in the review of [Holst 07].

2.2.2.11 A Method to Diagnose Multiple Arbitrary Faults

[Ye 10]

Ye, Hu and Li proposed a multiple-fault diagnosis method. The method has no assumption on fault models. To cope with the multiple-fault mask and reinforcement effect, two key techniques of construction and scoring of fault-tuple equivalence trees (FTET) are introduced to choose and rank the final candidates. In Figure 9, it show the overview of the diagnosis method.

For each failing pattern, the method traces from all failing observation point towards the primary input to construct a FTET. The tracing for each failing patterns starts from the first fault-tuple, which are all the failing observation points. The tracing starts from the first tuple. A score is assigned to each node in the tree. After a FTET is constructed for each failing pattern, the sites are selected based on the score until all the FTETs are pruned.

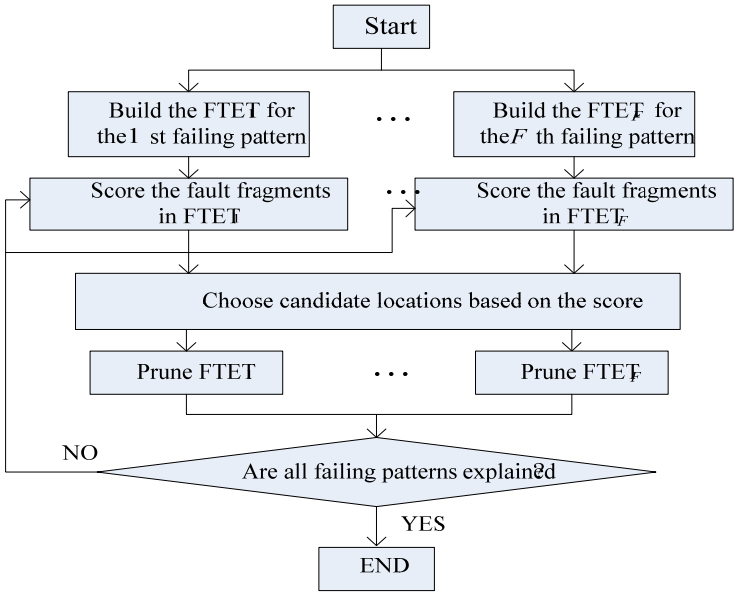


Figure 9: Overview of Diagnosis Method in [Ye 10]

The method shows good diagnosability and resolution. It has better diagnosis results compared with the latest diagnosis work [Yu 08_1]. However it is assumed that multiple faults are randomly present in the faulty circuit and did not investigate the issue when multiple faults are present in a bounded local region, which makes the diagnosis more challenging.

CHAPTER 3. IMPROVING COMPRESSED PATTERN GENERATION FOR MULTIPLE SCAN CHAIN FAILURE DIAGNOSIS

3.1 Introduction

The cost of test of manufactured VLSI circuits using scan based structural tests is determined by test application time and tester memory costs. Several methods to reduce these costs have been recently developed and are used in industry [Lee 99], [Ham 00], [Raj 04], [Bar 01], [Who 03]. All these methods divide the scan cells in to a large number of scan chains and use compressed tests which are decompressed using on-chip decompression logic. This allows use of a small number of tester channels to load the tests. Test response data is also compacted typically by a linear circuit and observed through a small number of tester channels. Compacted test responses negatively impact fault diagnosis due to reduced observability. Earlier methods to improve fault diagnosis for circuits using test response compaction include the use of bypass of compaction circuits, use of additional tests beyond production tests used to only detect defects [Kun 94], [Li 05], [Guo 07]. Bypassing compaction requires additional on-chip circuits and increased test data volume. Using additional tests to improve diagnosis can be done in two ways. One is to augment production tests. However since this approach increases test application time it is typically not used. The other approach is to use production tests first to detect defects and then use additional tests for diagnosis purpose. Additional tests may be based on diagnosis using the production tests [Guo 07]. However this method may require mounting the failing chips on the testers a second time. Use of additional tests to improve diagnostic resolution may not be applicable in volume diagnosis used for yield learning and yield ramp up [Chu 08]. Yield learning for very-large-scale integrated

circuits (VLSI) designed at current and future technology nodes of 45 nm and below will require diagnosis of tens to hundreds of thousands of failing chips [Sha 08]. The ideal solution for improved diagnosis is to improve the diagnosis achieved by production tests without increasing pattern counts by much. In this work we provide a simple method to achieve this goal.

3.2 Preliminaries

3.2.1 Test response compactors

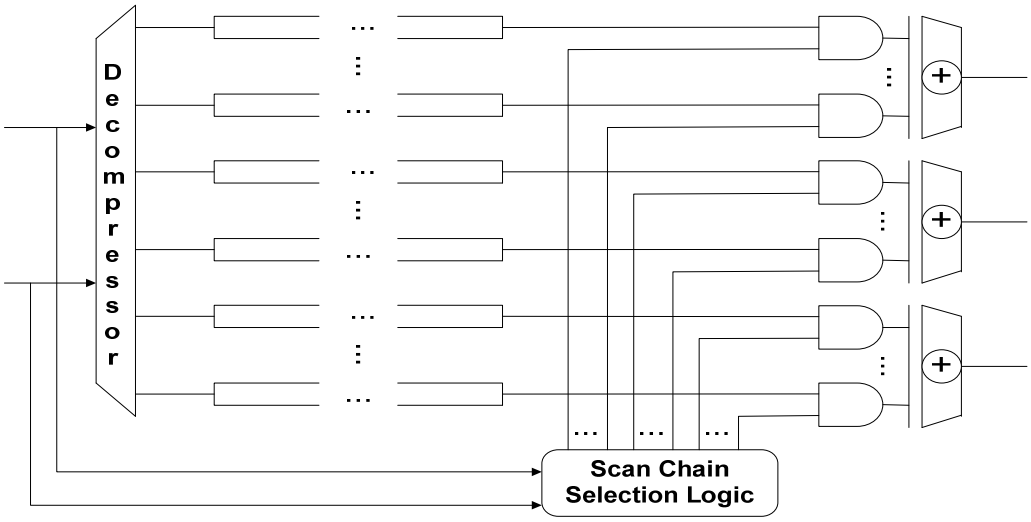


Figure 10: A Space Compactor with 3 Output Channels

Typically test responses are compacted either using multiple input signature registers (MISRs) [Elg 97] or trees of Exclusive ORs (XORs) called space compactors [Sal 83]. If MISRs are used then one has to prevent unknown values entering the compactor. Using XOR trees permits unknown values in test responses entering the

compactors. In this work we consider circuits using space compactors. Figure 10 shows a typical space compactor which compacts several scan chains into 3 output channels.

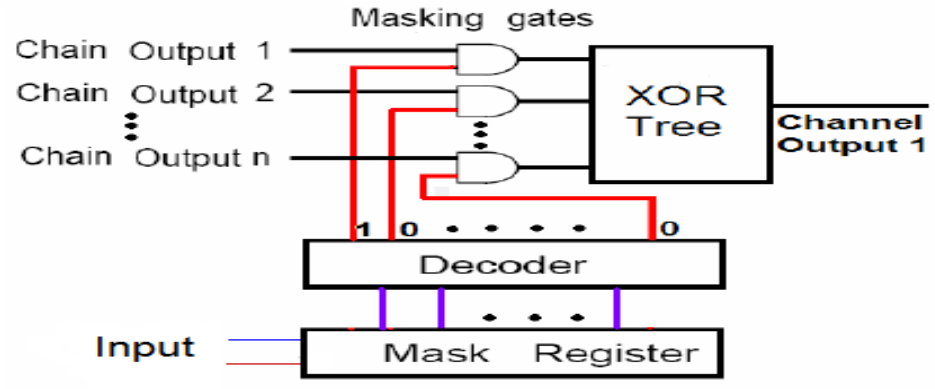


Figure 11: Scan Chain Selection Logic

The test response data in scan chains go through the XOR trees and are scanned out using the output channels. For space compactors, unknown values are permitted to enter the compactor. However, an unknown value corrupts all the test response data in the same scan cycle of the scan chains connected to the same output channel, thus masking the faults which are propagated to these scan cells. In order to maximize the detection ability of a test pattern and minimize the effect of unknown values, scan chain selection logic is usually utilized. Scan chain selection logic includes a mask register and a decoder as shown in Figure 11 for one output channel. The mask bits, determined separately for each pattern, are delivered through the test inputs to the mask register and the decoder is usually a linear logic circuit to decode the mask bits into masking signals. The masking signals drive inputs to AND gates. Usually the number of mask bits of a mask register is smaller than the number of masking signals. To reduce test data volume, mask bits are

determined for each test pattern and the decoded masking signal for each scan chain doesn't change during scan unload. As shown in Figure 11, all the scan cells on chain1 are observed through XOR tree and all the scan cells on the other chains are masked since the scan chain selection logic sets all except the input to the first AND gate to 0.

3.2.2 Problem formulation

When space compactors are used, internal scan chains are not observed directly at the output channel. The reduced observability of internal scan chains and the interaction between them can adversely impact scan-based diagnosis.

Scan chain failures are the cause for a substantial proportion of failing chips. As 30%-50% of logic gates of a typical chip impact the operation of scan chains, it is very likely that scan chain operations will be impacted by random and/or systematic defects. Meanwhile failures on multiple scan chains are observed much more frequently than they were before. [Bas 08] reported that 32% of 529 units with scan chain failures contained multiple chain failures. Note that with the space compactor, the number of scan chains is much larger than that of traditional scan designs, thus the probability of having multiple scan chain failures is even higher in a modern scan compression designs than traditional scan designs. Multiple scan chain failures can be caused by either independent defects that land on different scan chains or by defects on a global signal driving multiple scan chains. For example, a delay fault in a buffer in a clock tree can cause hold-time violations on multiple chains. Diagnosis of multiple chain failures with space compactors is challenging when multiple scan chains of the same output channel fail.

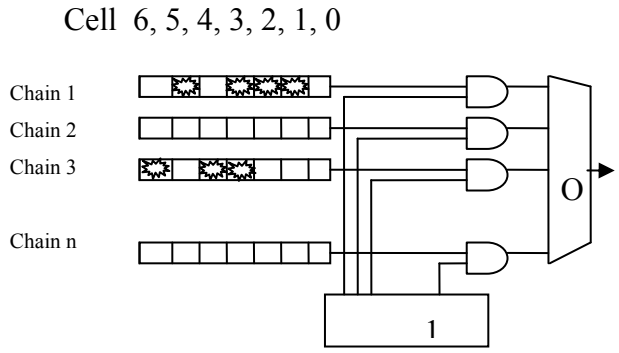


Figure 12: Illustrating Multiple Chain Failures

Figure 12 shows an example of how a space compactor can mess up the failures at an output channel when more than one scan chain among those observed through the output channel are faulty.

From Figure 12, it can be seen that in the failing chain “chain1” there are 4 failing bits at scan cells 1, 2, 3 and scan cell 5. In the failing chain “chain3” there is 3 failing bit at scan cells 3, 4, 6. The masking signals from the chain selection logic are shown in the figure. Failing chain “chain1” and failing chain “chain3” are both observed through the space compactor with single output. So after the space compactor due to compaction there are 5 failing bits at scan cells 1, 2, 4, 5 and scan cell 6. For diagnosis purposes, given the 5 failing bits, it becomes difficult for the diagnosis tool to know which scan chain is causing which failing bit. If two failing chains have failing bits at the same scan shift cycle, the two failing bits cancel each other, which also makes the chain diagnosis difficult. As shown in Figure 12, failing bits at scan cell 3 of “chain1” and scan cell 3 of “chain3” cancel each other. In the example illustrated in Figure 12, if “chain3” is masked

then the failures at the output will be caused only by the errors captured in “chain1” and this aids diagnosis. This example also illustrates the key idea behind the simple strategy we are proposing to improve diagnosis by production patterns. The idea is to mask or not observe at least one selected scan chain among those that are driving a single output of the compactor. However, in order to keep the pattern counts close to those normally obtained, we do this only when all the scan chains are selected for observation by the normal test generation flow. Thus we attempt to minimally disturb the normal test generation flow. Details of this procedure are given in Section 3.3.

In a regular production test pattern set, in the chain-test step, flush tests are applied and they include the tests which observe only one chain in each output channel at one time. Thus, which chains are faulty can be readily identified [Hua 05]. However, identifying which scan cells are faulty on faulty chains is very challenging. For regular production test patterns the scan chain selection logic observes all scan chains when no unknown value is in the scan cells in order to maximize the detection ability and minimize the pattern count. As discussed above, diagnosis of failing scan cells is made difficult when more than one chain observed through a single compactor output is faulty. In [Guo 07], a diagnostic pattern set was proposed for diagnosing failing scan cells in designs with space compactors. These additional patterns are generated based on the results of diagnosis using production patterns that detected the failing chip. Since the approach requires testing the failing chips a second time, it may be too expensive, if not impossible, to use in some test flows. The optimal solution and our goal in this work is to improve the diagnostic resolution achievable by production tests without increasing pattern counts or requiring additional test time.

3.3 A Method to Improve Diagnostic Resolution of Production Tests

We first give definitions of some terms used in this work. These terms define the nature of tests with respect to how the responses to them are observed through the output channels of a space compactor. **Non-masking pattern** denotes the patterns whose responses in all scan chains are observed through compactor outputs. **Partial-masking pattern** denotes the patterns whose responses in more than one chain are observed and the rest of the chains are masked by the chain selection logic. **1-hot pattern** denotes the patterns whose response in only one scan chain in each output channel is observed and all the other chains are masked.

3.3.1 Normal test generation flow

In Figure 13, a normal scan chain selection algorithm for a test pattern is shown. After a test pattern is generated, the procedure enters the chain selection flow.

First the pattern is fault simulated to determine fault propagation sites and unknown value locations in the scan chains. Based on the results of fault simulation, masking signals are assigned to one scan chain at a time and the masking signals are encoded together with the test or in additional bits scanned in after the test. Mask assignments and encoding is repeated until the encoding capability is reached. The number of mask bits of a mask register is less than the number of masking signals. So encoding capability is reached when all the mask bits of the mask register are determined, at which time the remaining masking signals are determined by the mask bits.

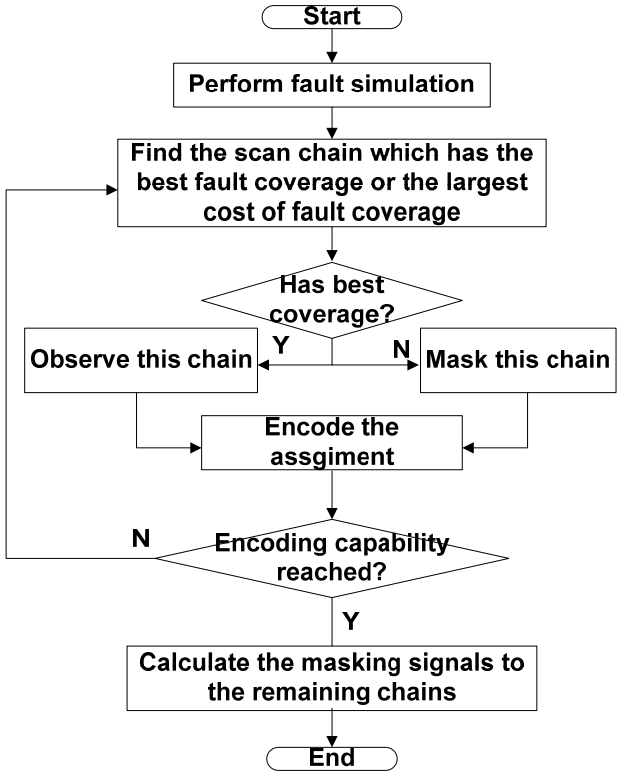


Figure 13: Normal Chain Selection Procedure

Detection of undetected faults is the priority for the normal chain selection procedure. For this reason, for most test patterns the scan chain selection logic selects all chains for observation when X (unknown values) ratio is not high. However these patterns may not provide efficient information for diagnosis.

3.3.2 Proposed chain selection procedure

In order to improve the diagnostic resolution by production patterns with minimal pattern count increase, we propose a new chain selection procedure in the test generation flow. While the normal chain selection procedure only considers the detection ability of each pattern and observes all scan chains when there are no unknown values in scan

chains that affect fault coverage, the proposed chain selection procedure considers the diagnosability of each pattern and masks a small number of chains without much loss of detection ability of a test pattern. This difference is the basic idea for an effective and simple method to increase the diagnosis capability of production patterns.

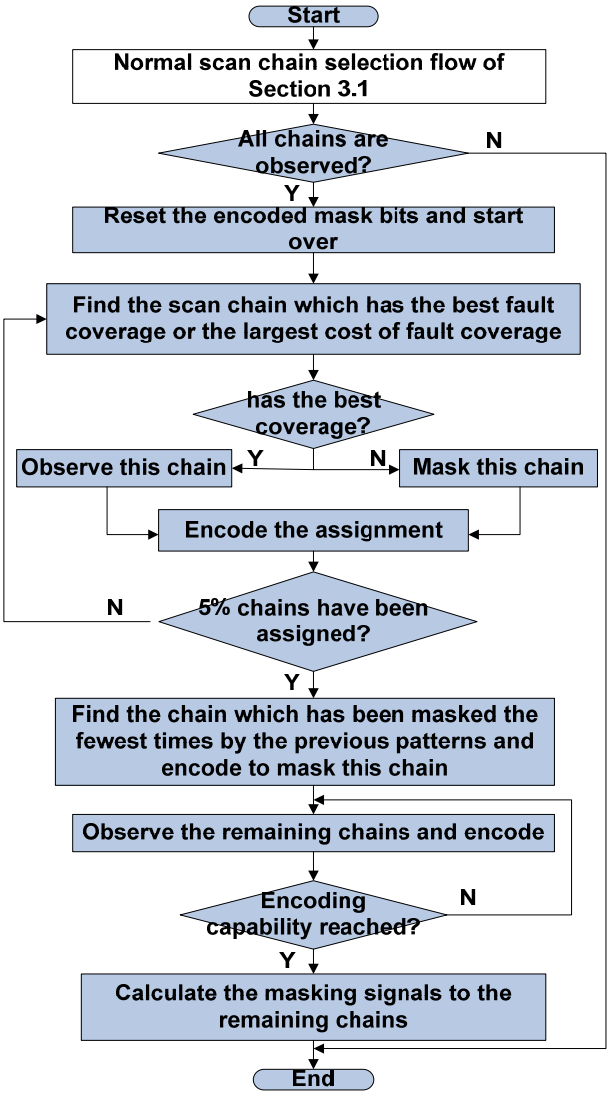


Figure 14: The Proposed Chain Selection Procedure

The proposed scan chain selection procedure is shown in Figure 14. It is explained below.After a test pattern is generated, first the normal chain selection flow is performed. At the exit of the normal flow, only if all the scan chains are observed, the proposed flow is entered.

If the proposed scan chain selection flow is entered, first the mask bit encoding is rolled back. In order to keep the detection ability of the pattern, we start assigning the masking signals as in the normal flow until 5% of all the scan chains are assigned masking signals. The 5% threshold can be changed. However our experiments suggest that in general this threshold is good for most circuits.

Next, we go through all the scan chains which have not been assigned masking signals and find the chain which has been masked the fewest times by the previous test patterns and we mask this chain. Then we set the mask bits to observe the remaining chains until the encoding capability is reached. Thus we attempt to mask a single chain that has been masked the least number of times in earlier generated patterns. It is possible that the encoded mask may mask additional chains or observe other chains but at least one scan chain is masked by this procedure. In the last step based on the determined mask bits, the masking signals of all the scan chains are set by the decoder in the scan selection logic and the scan chain selection flow is complete for this test pattern.

In the proposed procedure, in order to preserve the fault coverage, for each scan pattern, several chains which provide the best fault coverage gains are observed as in the normal flow. The rest of the scan chains are set to be masked or observed based on the improvement of diagnosis capability for multiple chain failures. The diagnosis capability of multiple chain failures is measured by the frequency of a scan chain being

masked/observed. The most frequently observed scan chains are given higher priority to be masked while the least frequently observed scan chains are given higher priority to be observed. By balancing the masking and observation of the scan chains, we improve the probability that some failing scan chains are observed while other chains are masked for some scan patterns. The proposed chain selection algorithm masks a small portion of the scan chains even when there are no X (unknown) states or limited X states for the generated test pattern. So there are more partial masking patterns and fewer non-masking patterns in the generated production test pattern set. This enhances the diagnosis capability of the production test pattern set by improving the possibility of differentiating multiple faulty chains.

The diagnosis results on several industrial designs in Section 3.4 show the effectiveness of the proposed technique.

3.4 Experimental Results

We modified an existing commercial ATPG tool that accommodates test response compactors to derive production tests using the proposed strategy. We injected different types of defects in multiple scan chains. Diagnosis based on response to the production tests was done using a commercial fault diagnosis tool.

Experimental results are given for four industrial designs. All the designs used space compactors as shown in Figure 10. Different types of single faults are injected at random scan cell locations into 2 or 3 scan chains connected through a space compactor to the same output channel. As we indicated earlier, the faulty chains are identified by production tests during the chain-test phase using 1-hot patterns. Thus, the objective of diagnosis is to determine the faulty scan cells in the defective scan chains.

Table 2: The Statistics and Pattern Counts of 4 Designs

	Design 1	Design 2	Design 3	Design 4
# of chains	401	160	400	160
# output channels	8	4	8	2
# of gates	320K	496K	1.1M	1.4M
X ratio*	0.003%	0.04%	0.004%	0.98%
# of normal patterns	4755	2065	5561	2847
# of rand. patterns*	4761	2081	5639	2875
# of proposed patterns	4757	2070	5611	2866
Pattern count increase*	0.04%	0.24%	0.90%	0.67%

* X ratio: the ratio of X bits in test responses in scan cells

* # of rand. patterns: the number of the test patterns generated using the random method

In addition to the proposed method, we also implemented a random method for comparison. The two methods differ in only one step. In the proposed method as shown in Figure 14, in one step we go through all the scan chains which have not been assigned masking signals and find the chain which has been masked the fewest times by the previous test patterns and we mask this chain. In the random method, in this step we randomly select a chain from those which have not been assigned masking signals and mask this chain.

The statistics of the four designs are shown in Table 2 together with the pattern counts of the normal production tests and the production tests generated using the random method and using the proposed method. The pattern count increase from the normal

production tests to the tests generated using the proposed method is shown in the last row of Table 2.

From Table 2, we can see that the pattern count increase varies from 0.04% to 0.90%, the maximum of which is less than 1%. The fault coverage for all the designs are the same using either the normal production tests or the proposed test patterns. Also we can see that the numbers of patterns generated using the random method are larger than the numbers of patterns generated using the proposed method for all the four designs.

In Table 3, the diagnosis results for injected stuck-at faults in 2 scan chains are given. In Table 4, the diagnosis results for injected stuck-at faults in 3 scan chains are given. In Table 5, the diagnosis results for injected timing faults in 2 scan chains are given. The timing faults for each design have the same number of slow-to-rise, slow-to-fall, fast-to-fall, fast-to-rise and hold-time faults.

Table 3: Experimental Results for Stuck-at Faults on 2 Faulty Chains

Design / Result	Design 1 (64 cases)			Design 2 (40 cases)			Design 3 (40 cases)			Design 4 (40 cases)		
	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.
2 chains	4	33	45	26	35	38	0	21	22	23	26	31
1 chain	21	29	17	12	4	2	7	16	16	15	13	9
0 chain	39	2	2	2	1	0	33	3	2	2	1	0
Ave. Res.	17%	57%	68%	72%	78%	82%	16%	49%	50%	61%	65%	69%

Table 4: Experimental Results for Stuck-at Faults on 3 Faulty Chains

Design / Result	Design 1 (24 cases)			Design 2 (20 cases)			Design 3 (24 cases)			Design 4 (20 cases)		
	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.
3 chains	0	9	11	6	12	17	0	5	9	9	10	13
2 chain	1	7	10	4	5	2	2	12	7	6	8	7
1 chain	8	6	3	6	3	1	3	5	6	2	1	0
0 chain	15	2	0	4	0	0	19	2	2	3	1	0
Ave. Res.	10%	47%	56%	47%	69%	84%	10%	40%	51%	53%	58%	64%

Table 5: Experimental Results for Timing Faults on 2 Faulty Chains

Design / Result	Design 1 (40 cases)			Design 2 (40 cases)			Design 3 (40 cases)			Design 4 (30 cases)		
	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.	Nor.	Rand.	Prop.
2 chains	2	24	32	19	27	31	6	21	27	24	27	28
1 chain	13	9	3	11	8	5	4	18	12	0	2	1
0 chain	25	7	5	10	5	4	30	1	1	6	1	1
Ave. Res.	18%	69%	80%	53%	73%	81%	15%	57%	70%	75%	88%	90%

If the number of suspect scan cells reported by the diagnosis tool for a failing chain is no more than 5 and the defective scan cell is in the reported suspect scan cells, we

consider that this chain is diagnosed. Otherwise, we consider that it is not diagnosed successfully.

In each table, we list the number of scan chains that are successfully diagnosed by the normal test patterns, by the test patterns that are generated using the random method, and by the test patterns that are generated using the proposed scan chain masking selection technique. The rows with "N chains" where N is 0, 1, 2, or 3, show the number of scan chains that are successfully diagnosed by the commercial diagnosis tool used.

As can be seen from Tables 3-5, the proposed method is very effective in improving the diagnostic resolution. For example from Table 3, for design 1 with two faulty chains, using normal scan chain selection flow for only 4 out of 64 cases we have no more than 5 suspect scan cells identified in both the failing chains. However, using the tests by the proposed method, in 45 out of 64 cases the number of suspect scan cells is within 5 for both the failing chains. This higher success rate is achieved at the cost of only 0.04% increase in pattern count.

Meanwhile, we can see that the proposed method is more effective in improving the diagnostic resolution than the random method. For example from Table 4, for design 3 with three faulty chains, using the random method for only 5 out of 24 cases, we have no more than 5 suspect scan cells identified in all the three failing chains. However, using the tests by the proposed method, in 9 out of 24 cases the number of suspect scan cells is within 5 for all the three failing chains.

We also calculated the average diagnostic resolution for each design. The average diagnostic resolution is calculated as follows: The diagnostic resolution of a failing chain is the reciprocal of the number of the suspect scan cells for this chain. For example, if the

diagnosis result gives 4 suspect scan cells for a chain including the defective scan cell, then the diagnostic resolution is 25%. The average resolution, given in the last row of Tables 3-5, is the average over all the injected faults. From Table 3 we note that the average resolution for Design 1 is also improved from 17% to 68%. Similar improvements in average resolution can be observed for the cases of stuck-at faults in 3 scan chains and for timing faults given in Tables 4 and 5, respectively. As can be seen, for all the cases from Table 3 to Table 5, the random method improves diagnostic resolution but not as much as the proposed method does.

The proposed method can be readily adapted to any space compactor designs using any test flow. With minimal increase in test pattern counts and without fault coverage degradation, diagnostic resolution can be improved effectively. Thus we can potentially avoid using additional diagnostic test patterns for some diagnosis of multiple scan chain failures.

CHAPTER 4. IMPROVING DIAGNOSTIC TEST GENERATION FOR SCAN CHAIN FAILURES USING MULTI-CYCLE SCAN PATTERNS

4.1 Motivation

In software-based diagnosis, usually the production test patterns used to detect defective chips are used to collect test failures for diagnosis. Since the aim of production tests is only fault detection, they may not provide sufficient information for diagnosis purpose, leading to poor diagnosis. In [Huang 06], it is reported that for only 23% of failing dies, diagnosis based on production tests gave less than or equal to three scan cell suspects, which is regarded as the minimum desirable resolution. To improve diagnostic resolution, diagnostic test patterns can be generated either before or after initial diagnosis using production test patterns. In this work we propose an effective method to generate test patterns with a higher rate of successful diagnosis to improve the resolution of the diagnosis.

Previous diagnostic test pattern generation techniques for scan chain failures target a single scan cell at a time, which could be a time consuming process when there are hundreds of scan cell to target, especially for very large designs. In this work we propose a speed-up technique with up to 20X run time improvement.

4.2 Previous Work

Methods to generate additional test patterns to improve diagnostic resolution have been proposed. In this work we call these patterns *diagnostic test patterns*.

Several diagnostic test generation techniques for scan chain failures have been proposed. The test pattern generation method by Kundu [Kundu 94] used a sequential

ATPG but, as pointed out in [Guo 07], it did not constrain the tests appropriately to insure that sufficient conditions for diagnosis are met. Li [Li 05_1] [Li 05_2] and Guo [Guo 07] proposed test generation algorithms for scan chain stuck-at and timing faults. In [Li 05_1] [Li 05_2] scan cells are targeted one at a time while the entire faulty scan chain is masked as unobservable. In [Guo 07], an algorithm to generate diagnostic test patterns with a single capture cycle was proposed. If a test is generated successfully for each cell, a complete diagnostic test set can be created to give only one suspect scan cell which is the faulty one. However, due to the added constraints and masking of scan cells, the procedures in [Li 05_1] [Li 05_2] [Guo 07] might fail to generate a pattern that meets all the conditions for some scan cells thus affecting diagnostic resolution. In [Wang 08], a diagnostic test generation method is proposed to target scan chain defects in the presence of defects in the logic part of the DUT. It targets a scan cell using only fault propagation conditions. Specifically, test generation using functional data capture, which is used in the test generation procedure we propose, is not used by Wang.

4.3 Preliminaries

In this work we propose an effective method to generate test patterns with a higher rate of successful diagnosis. Even though the proposed method can be extended to diagnose other fault models and multiple faulty scan cells/chains, in this work we consider stuck-at faults in a single faulty chain in the experimental data given later. Additionally we assume that the combinational logic of the circuit under test is fault free. This assumption can be relaxed by, for example, adopting procedures similar to those in [Wang 08]. We also assume a faulty scan cell fails during shift and capture cycles. Even though in this work we assume that a single scan cell in a faulty scan chain is defective

and is stuck-at-0 or stuck-at-1, the proposed procedures can be extended to diagnose other faults. Also, for simplicity, we assume that all the scan cells have no inversions between a scan cell and the scan chain output.

Given a test set and a fault model, a scan cell N is differentiated from scan cell M if and only if the response of the faulty circuit with the fault at scan cell N is different from the response of the faulty circuit with the fault at scan cell M. If a test set differentiates each scan cell in the range of suspect cells from the other scan cells in the range, we call the test set a complete diagnostic test set, which produces a unique response for each faulty scan cell. We can locate the exact faulty scan cell by diagnosis based on a complete diagnostic test set.

The diagnostic test generation procedures generate a test targeting a single scan cell in an ordered list of scan cells. The list of scan cells is typically an ordered list of consecutive scan cells. In this work, by *generating a test targeting a scan cell* or simply *targeting a scan cell* we mean that if a test is successfully generated it can differentiate the targeted scan cell from all scan cells that appear in the list after the targeted scan cell. By *differentiating a scan* cell we mean that the target scan cell among the list of suspected scan cells is classified correctly as fault-free or faulty.

Flush tests are used to detect scan cell defects and to determine the fault type in a defective scan chain.

Production test patterns may be used for diagnosis prior to generating diagnostic patterns. The initial diagnosis using production test patterns can determine a range of suspect cells with an upper bound (UB) and lower bound (LB).

For some large designs, test failures for production patterns may not be collected due to tester storage limitations since scan failures fail at most all tests and many errors appear in every test response. In such cases the initial suspect candidates in a scan chain will be every scan cell in the chain. Earlier proposed scan chain diagnostic test generation flow takes impractically large run times to complete in these cases as we show in Section 4.6. We also demonstrate that the speed-up flow we propose reduces diagnostic test generation times by an order of magnitude.

4.3.1 Analysis of the test generation algorithm in [Guo 07]

The methods we are proposing to improve diagnostic resolution and run times builds on the procedures given in [Guo 07]. For this reason, next we briefly discuss the basic diagnostic procedure of [Guo 07].

We assume that the type of stuck-at fault, stuck-at-0 or stuck-at-1, is known from the response to flush tests as discussed above and the range of suspect scan cells is known from initial diagnosis, for example using production test patterns. If no initial diagnosis result is available prior to generating diagnostic test patterns, the range of suspect scan cells is the entire scan chain. Let UB and LB be the upper and lower bound, respectively, of the scan cells suspects. The procedure given below was proposed in [Guo 07] to create a complete test set to diagnose scan cells between UB and LB.

Diagnostic_Test_Gen (LB, UB, stuck-at-v)

Step 1: *Set N=UB*

Step 2: *Constrain scan load values of scan cells N, N-1, N-2, … , 0 to value ‘v’*

Step 3: *Generate a test pattern to capture value (1-v) into the data (functional) input of scan cell N-1.*

Step 4: *Set N=N-1*

Step 5: *If (N > LB), go back to Step 2. Otherwise, go to Step 6.*

Step6: *All the scan cells are targeted, end.*

To differentiate scan cell N from cell N-1 by capturing data through the combinational logic, the scan unload values on scan cell N-1 should be different for the circuit with the fault on cell N (F_N) and the circuit with the fault on scan cell N-1 (F_{N-1}). During the scan unload process, value (1-v) on cell N-1 can be correctly shifted out in the faulty circuit F_N, while in the faulty circuit F_{N-1} only value 'v' can be shifted out of scan cell N-1. So in order to differentiate F_N and F_{N-1}, value (1-v) is captured into scan cell N-1 using a scan based test.

As shown in Figure 15, the stuck-at-v fault at scan cell N may propagate to scan cell N-1, causing the captured value into scan cell N-1 to be 'v' instead of (1-v). If this happens, we cannot determine whether it is scan cell N or scan cell N-1 that caused the wrong observed value in the unload value of cell N-1. The captured value of scan cell N-1 must be valid no matter whether the fault location is scan cell N or cell N-1, as we assume that the combinational logic is fault-free. For this reason scan cell N and all the cells downstream of it are constrained to value 'v' during scan load. This is a sufficient condition to guarantee that a generated single capture cycle pattern can differentiate scan cell N from cell N-1. Similarly this procedure can also be used to differentiate scan cell N from all the cells downstream of it.

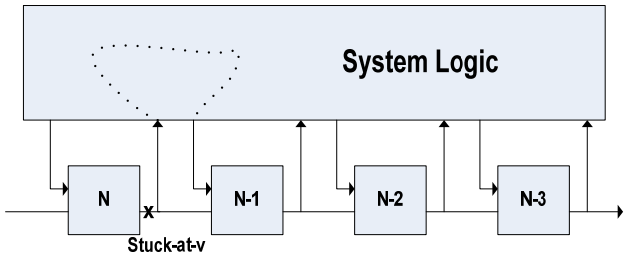


Figure 15: Illustration of Scan Cell N Impacting Cell N-1

Although a complete test set can be created if the procedure given above can target each scan cell successfully, there is an issue in this algorithm. The issue is that the above procedure only provides sufficient conditions for scan patterns with only one capture cycle.

As shown in Figure 16(a), since the scan load values of the downstream cells of the target scan cell N are constrained to value ‘v’, for some scan cells it may be very difficult, if not impossible, to capture a (1-v) value into scan cell N-1 in just one cycle and hence this procedure may fail to generate an effective pattern for some scan cells. Multiple capture cycles can be used to capture a desired value into scan cell N-1 when it is hard to capture the desired value in just one capture cycle. As shown in Figure 16(b), constraints are placed on scan cell N and all its downstream cells only during scan load. With the cycle expansion, it may be possible to capture value (1-v) into scan cell N-1 while it’s impossible with one capture cycle.

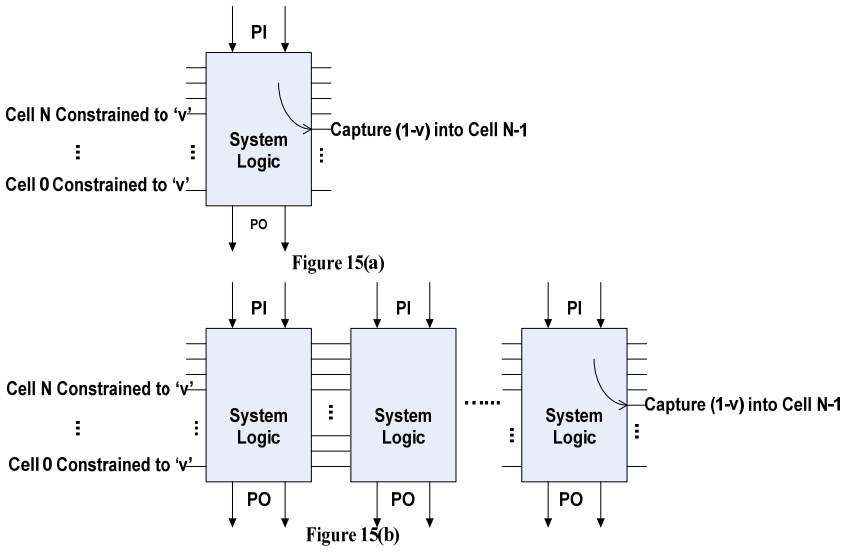


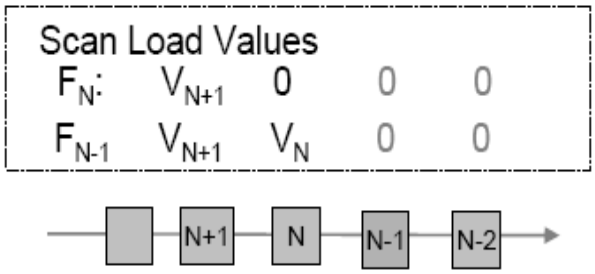
Figure 16: Illustration of Using Multi-Cycle Scan Patterns

Use of tests with multiple capture cycles is the basic idea of the proposed test generation algorithms to improve diagnostic resolution. However, as we discuss in Section 4.4, we need to add an appropriate additional constraint to generate the desired scan patterns with multiple capture cycles.

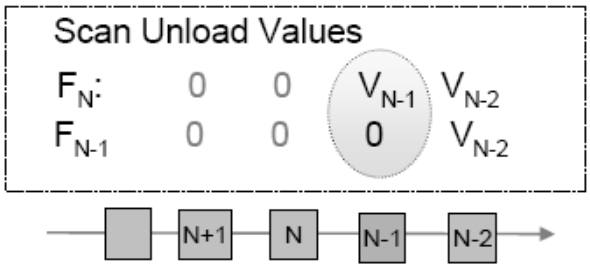
4.4 Procedures to Improve Diagnostic Resolution

In Section 4.4, a new procedure is proposed to improve diagnostic resolution of scan chain failures, assumed to be stuck-at faults. The proposed procedure preserves the sufficient condition which guarantees that a generated pattern can differentiate the target scan cell and improves the success rate of generating a test pattern to differentiate a scan cell by using tests with multiple capture cycles. We also discuss how to determine the number of capture cycles to be used and how to extend the procedure to other fault models.

4.4.1 Differentiate Scan Cells through Data Input Capture



(a): Scan load values in Faulty Circuits with a Stuck-at-0 Fault



(b): Scan Unload Values in Faulty Circuits with a Stuck-at-0 Fault

Figure 17: An Example with a Stuck-at-0 Fault

Assume that the faulty scan chain has stuck-at-v fault. We try to differentiate scan cell N from its downstream scan cells through data (functional) input capture from system logic. In Figure 17, VN is the scan load or scan unload value on scan cell N. As shown in Figure 17(b), to differentiate scan cell N from scan cell N-1, the unload values on scan cell N-1 should be different for FN and FN-1. If the fault is at scan cell N, we want to see an unload value (1-v) on scan cell N-1. If the fault is at scan cell N-1, we

want to see an unload value ‘v’ on scan cell N-1. So in order to differentiate scan cell N from scan cell N-1, we need to capture value (1-v) into scan cell N-1.

For patterns with more than one capture cycle, we propose to use extra constraints to make sure that a fault on scan cell N doesn’t propagate to scan cell N-1 during capture. As shown in Figure 15, if no constraints are placed on scan cells after scan load, a stuck-at-v fault on scan cell N may propagate to scan cell N-1, precluding capturing of (1-v) into scan cell N-1. So to prevent the fault effect of scan cell N from propagating to scan cell N-1, the test is generated assuming unknown content of cell N. This extra constraint will guarantee that the captured value (1-v) on scan cell N-1 is valid even if scan cell N is faulty and hence the patterns can be used to differentiate scan cell N from its downstream scan cells. To summarize, the proposed procedure is given below as Procedure 1.

Procedure 1: Test_Gen_for_Target_Cell_Input (N, stuck-at-v)

Step 1: Constrain scan load values of scan cells N, N-1, N-2, …, 0 to value ‘v’

Step 2: Mask scan cell N by setting it to unknown value X for all capture cycles.

Step 3: Generate a test pattern with single and when necessary multiple capture cycles, to capture value (1-v) on data input of scan cell N-1 with cell N content set to unknown value X.

Compared with the procedure analyzed in Section 2.1, we add more sequential constraints on scan cell N in Step 2. The following discussion shows that the conditions in Steps 1 and 2 are sufficient to differentiate scan cell N from its downstream scan cells.

In order to prove that the generated pattern guarantees to differentiate scan cell N from cells downstream of it, we prove that the test response of a faulty circuit with a fault

at cell N (FN) is different from the test response of a faulty circuit with a fault at scan cell N-1 (FN-1), or N-2 (FN-2), …, or 0 (F0).

If the fault is at scan cell N, the above constraints in Steps 1 and 2 make sure scan cell N-1 captures value (1-v) into it. The captured value of (1-v) on scan cell N-1 can be correctly shifted out and be observed during scan unload, i.e., there is no failure observed at scan cell N-1 since the fault is at scan cell N.

If the fault is at any scan cell downstream to scan cell N, the unload value of scan cell N-1 will be the incorrect value 'v', because scan cell N-1 is faulty or some other cell downstream of it is faulty.

As discussed above, the generated test pattern can guarantee to differentiate scan cell N from cells downstream of it. If all scan cells in the range of suspect cells are targeted successfully by this procedure, each scan cell will have unique faulty behavior for the test pattern set and a complete diagnostic test set will be obtained.

4.4.2 Sequential Depth for Procedure 1

To generate a multi-cycle scan test pattern described in Procedure 1, one needs to determine how many capture cycles are needed, also called the sequential depth of the tests, to generate the pattern. We can attempt to generate patterns by increasing the number of sequential depth one at a time. However, it is beneficial to avoid, as much as possible, sequential depths at which a desired pattern cannot be obtained. Before running the ATPG, the earlier proposed testability measure sandia controllability and observability analysis program (SCOAP) [Goldstein 79] can be applied to calculate the minimal sequential depth at which a pattern may exist. The procedure to use SCOAP [Goldstein 79] to calculate the smallest sequential depth is described below.

C0_depth of a gate denotes the sequential depth needed to control the output of the gate to value 0, and C1_depth of a gate denotes the sequential depth needed to control the output of the gate to value 1.

Step 1: Set C0_depth=M and C1_depth=M for all the gates in the design. M is the maximum sequential depth used by the sequential ATPG.

Step 2: Initialize C0_depth=1 and C1_depth=1 for all the Primary Inputs (PIs) in the design.

Step 3: Initialize C0_depth=1 and C1_depth=1 for all the scan cells that are not constrained. If the scan cell is constrained to value 1 during scan load, only set C1_depth=1 for this scan cell; If the scan cell is constrained to 0 during scan load, only set C0_depth=1 for this scan cell.

Step 4: Similar to logic simulation, the calculation of C0_depth and C1_depth based on the initialized values of PI and scan cells is performed based on SCOAP concept. For example, for an AND gate, the C0_depth is the minimum of the C0_depth among the input gates. The C1_depth is the maximum of the C1_depth among the input gates. For an OR gate, the C0_depth is the maximum of the C0_depth among the input gates. The C1_depth is the minimum of the C1_depth among the input gates. The SCOAP idea applies to other gates in the same manner.

Step 5: If there is no change of C0_depth or C1_depth in Step 4, stop. Otherwise, go back to Step 4.

In the proposed test generation procedure, in order to generate a test pattern to capture a value (1-v) on data input of scan cell N-1, C(1-v)_depth of the gate connected to the data input of cell N-1 gives the minimum sequential depth needed to generate such

a pattern. The diagnostic test generation engine can then generate a test pattern starting with this calculated sequential depth. Note that if C0_depth (C1_depth) of the gate connected to cell N-1 is M, it indicates the data input of cell N-1 cannot be set to value (1-v) within the sequential depth of M. If desired, when the sequential depth given by the procedure is too high, one can abort generation of the test to save run time.

4.4.3 Practical Considerations

As mentioned in the above Section 4.3, it is possible that the proposed test generation Procedure 1 fails to generate a test pattern for some scan cells. In [Guo 07], two additional procedures were used when the basic procedure described earlier does not generate a test pattern for differentiating a target scan cell. In this work, we propose two procedures similar to the two additional procedures in [Guo 07]. In the proposed procedures we use additional sequential constraints on scan cells to guarantee sufficient conditions for multi-cycle scan patterns.

4.4.3.1 Extension of Procedure 1 to Procedure 2

If Procedure 1 fails to generate a pattern to differentiate scan cell N from all its downstream scan cells, we try generating a pattern to differentiate cell N from cell N-x, N-x-1, …, 0, where x is a positive integer larger than 1. The proposed procedure is given below as Procedure 2:

Procedure 2: Extended_Test_Gen_for_Cell_Input (N, N-x, stuck-at-v)

Step 1: Constrain scan load values of scan cells N, N-1, N-2, …, 0 to value 'v'

Step 2: Mask scan cell N by setting it to unknown value X for all capture cycles.

Step 3: Generate a test pattern to capture value (1-v) into data (functional) input of scan cell N-x with cell N content set to unknown value X.

The constraints in Steps 1 and 2 prevent a stuck-at-v fault at scan cell N impacting captured value into cell N-x during scan load and multiple capture cycles. So if scan cell N is faulty, we will see value (1-v) on cell N-x after unloading. And if scan cell N-x or any of its downstream cells is faulty, we will see value 'v' on cell N-x after unloading. Thus, the proposed Procedure 2 can differentiate scan cell N from scan cells N-x, N-x-1, … , 0.

4.4.3.2 Test Generation Using Fault Propagation

We also propose another procedure called Procedure 3 (given below) to differentiate cell N from N-x by fault propagation. The idea is to propagate the effect of the fault at the output of scan cell N while not activating a fault at the output of scan cell N-x. Procedure 3 does not generate tests that insure a sufficient condition to differentiate scan cell N from all of its downstream scan cells other than scan cell N-x.

The proposed Procedure 3 is described below:

Procedure 3: Pair_Diff_Through_Cell_Output(N, N-x, stuck-at-v)

Step 1: Constrain scan load values of scan cells N-1, N-2, ... , 0 to value 'v' .

Step 2: Mask scan cell N-x by setting it to unknown value X for all capture cycles.

Step 3: Constrain upstream scan cells of N-x and scan cell N-x unobservable during scan unload.

Step 4: Generate a test to detect stuck-at-v fault at data output of scan cell N by capturing the fault effect in some observable scan cell or primary output with cell N-x content set to unknown value X.

If the fault location is scan cell N, the generated test pattern propagates the fault effect to a reliable observation point. So we have specific failures to differentiate scan

cell N from scan cell N-x if the fault is at scan cell N. If the fault location is scan cell N-x, due to the constraints, no faulty value will be observed during scan unload. So the conditions used are sufficient to guarantee that the generated test pattern can differentiate scan cell N from cell N-x.

4.4.4 Extension to Timing Failures

The proposed algorithm can be extended to generate diagnostic test patterns for timing faults. In Section 4.4.4, we briefly explain the extended procedure for fast-to-rise fault model. Other timing faults can be handled similarly.

When scan cell N has a fast-to-rise fault, a 0 to 1 transition will happen one cycle earlier. In order to differentiate scan cell N from cell N-1, the generated test pattern should have a different test response of the faulty circuit with the fault at scan cell N (FN) and the faulty circuit with the fault at scan cell N-1 (FN-1). The proposed Procedure 4 is given below:

Procedure 4: Extended_Test_Gen_for_Fast-to-rise (N, fast-to-rise)

Step 1: Constrain scan load values of scan cells N, N-1, N-2, …, 0 to a constant value ‘1’ or ‘0’.

Step 2: Mask scan cell N by setting it to unknown value X for all capture cycles but the last cycle.

Step 3: Generate a test pattern with single and, when necessary multiple capture cycles, to capture value ‘0’ into the data input of scan cell N-1 and value ‘1’ into the data input of scan cell N.

Similar to the analysis in Section 3.1, the constraints in Steps 1 and 2 make sure the captured values into scan cell N-1 and cell N are valid even if the fault is at scan cell

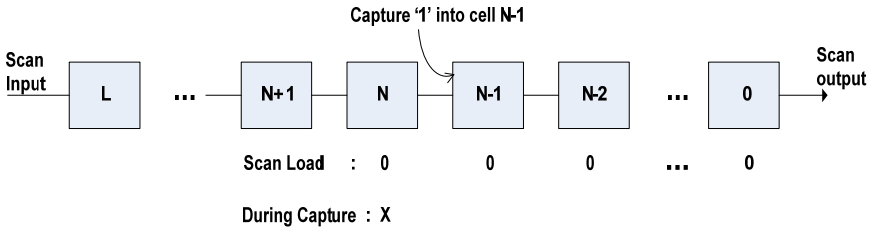
N. So if scan cell N is faulty, the value ‘0’ on scan cell N-1 can be correctly shifted out and we will see value ‘0’ on cell N-1 after unloading. If scan cell N-1 or any of its downstream cells is faulty, the value ‘0’ on scan cell N-1 will be corrupted during scan unload and we will see value ‘1’ on cell N-1 after unloading. Thus, the proposed Procedure 4 can differentiate scan cell N from all the downstream cells of it.

4.5 Improve Runtime Performance of Diagnostic Test Generation

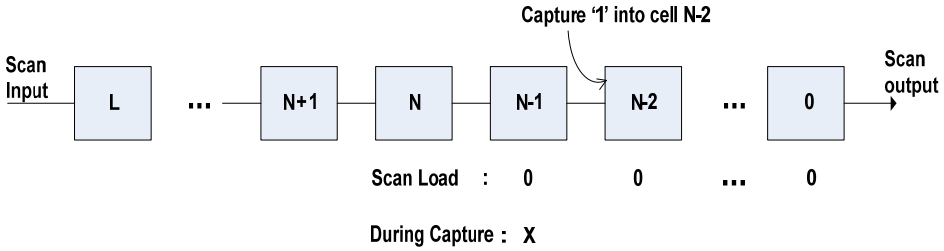
Diagnosis based on production test patterns may contain a large number of suspect scan cells. Or when no initial diagnosis results are available, for example, when diagnosis is based on production tests, the range of the suspect scan cells is the entire faulty scan chain. In such cases, if the range of the scan cell suspects contain hundreds or more scan cells, run time for test generation and the number of test patterns may become an issue, especially for very large designs. In Section 4.5, we propose a test generation flow to speed up the generation of diagnostic tests and also reduce the number of test patterns.

The test generation flow used in Procedure 1 targets scan cells one at a time with appropriate constraints on scan cell contents. If the faulty scan chain has 1000 scan cells, we need to call the test generation procedure 999 times and each time the scan cell constraints are changed appropriate for the target scan cell.

4.5.1 The Key Idea behind the Speed-up Flow



(a): Scan Cell Constraints when Targeting Scan Cell N (SA0)



(b): Scan Cell Constraints when Targeting Scan Cell N-1 (SA0)

Figure 18: Scan Cell Constraints

Suppose the faulty scan chain has a stuck-at-0 (SA0) fault, as shown in Figure 18(a). When we use Procedure 1 to target scan cell N, scan cell N and all the downstream scan cells to it are constrained to load value '0'. Meanwhile during capture cycles, the scan cell N is masked by setting its content to unknown value X. The test generation procedure tries to generate a test pattern to capture value '1' into the data input of scan cell N-1.

In Figure 18(b), we show the scan cell constraints when targeting scan cell N-1. Scan cell N-1 and all the downstream scan cells to it are constrained to load value '0'.

Meanwhile during capture cycles, the scan cell N-1 is masked by setting its content to unknown value X. The test generation procedure tries to generate a test pattern to capture value '1' into the data input of scan cell N-2. If we use the constraints shown in Figure 18(a) and generate a test pattern to capture value '1' into scan cell N-2 as shown in Figure 18(b), the pattern may still successfully targets scan cell N-1.

As we discussed in Section 4.3, for single-cycle patterns, it is not necessary to add the extra constraint that mask cell N by setting its content to unknown value X during capture cycles. So if the generated pattern has single capture cycle, the constraints for targeting scan cell N as shown in Figure 18(a) are a superset of the constraints for targeting scan cell (N-1) as shown in Figure 18(b) or any downstream scan cell of it. So if we use the scan cell constraints for targeting scan cell N and generate single-cycle test patterns to capture value '1' into any downstream scan cells to scan cell N at the same time, the generated patterns can simultaneously target many more scan cells.

If the generated pattern has multiple capture cycles, the constraints for targeting scan cell N are no longer a superset of the constraints for targeting scan cell N-1 or any downstream scan cell of it, as can be seen from Figure 18(a) and 18(b). When targeting scan cell N-1, cell N-1 needs to be masked. So if we use the scan cell constraints as shown in Figure 18(a) and generate a multi-cycle test pattern to capture value '1' into cell N-2, this pattern may not successfully target scan cell N-1 since cell N-1 is not masked. To confirm if this test pattern can target scan cell N-1, we can add the cell constraints as shown in Figure 18(b) and simulate the generated test pattern to see if we still capture value '1' into scan cell N-2. If so, we can say that the generated test pattern using constraints shown in Figure 18(a) also successfully target scan cell N-1. If not, scan cell

N-1 is not successfully targeted by this pattern and thus we need to target scan cell N-1 by generating an additional test pattern.

The basic idea of the performance improvement of the flow is to use the same cell constraints to generate test patterns to capture '1' into as many downstream scan cells as possible, followed by using fault simulation to determine which scan cells are successfully targeted by the generated test pattern.

4.5.2 The Proposed Diagnostic Flow for Speeding-up

The proposed faster diagnostic test generation flow is shown in Figure 19.

In Step 0 we add no scan cell constraints on the faulty scan chain and use an ATPG engine to generate patterns to capture value (1-v) into all the suspect scan cells. Such patterns detect stuck-at-v faults on the data (functional) inputs of the scan cells. If a scan cell data input is identified ATPG untestable (AU) for capture value (1-v) due to the design constraint, we can skip these scan cells in the following steps since it's impossible for them to capture value (1-v) after adding additional cell constraints needed for diagnostic tests.

In Step 1, we find the most upstream scan cell that is not identified as AU in Step 0 and call the scan cell N. In Step 2 we add scan cell constraints as described in Procedure 1 and generate test patterns to capture value (1-v) into scan cell N-1 and also into all the scan cells downstream to it except those that have been identified AU in Step 0.

In Step 3, we determine the cells successfully targeted by any single-cycle patterns generated in Step 2. In Step 4 we determine the cells successfully targeted in Step 2 by using fault simulation with appropriate constraints as discussed in Section 4.4.

In the following iterations, we move on to the next scan cell and set it as scan cell N. We try to generate a pattern to capture value (1-v) into cell N-1 and also all the scan cells downstream to N-1 except the scan cells which meet any one of the following 3 conditions: (1) If a scan cell is already identified as AU in Step 0. (2) If in previous iterations, a multi-cycle pattern captured value (1-v) into the scan cell during the pattern generation phase, but confirmed by simulation that the scan cell does not capture (1-v) when appropriate cell constraints are applied. (3) If a scan cell has been earlier identified as AU in Step 6 two times.

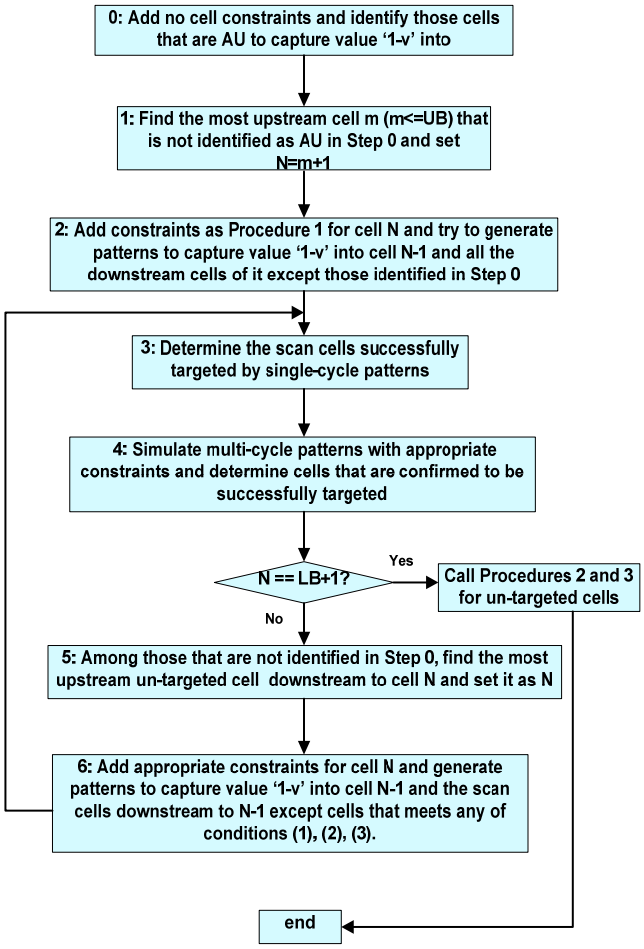


Figure 19: Diagnostic Test Generation Flow

If a scan cell meets condition (2) or (3), it is likely that the cell will be identified as AU even with appropriate constraints on scan cells or the cell can only successfully capture value (1-v) with appropriate constraints. So by excluding these scan cells in Step 6, we can avoid spending ATPG effort on these scan cells in many iterations to save run time. These scan cells will be dealt with specifically when they are set to scan cell N with appropriate constraints.

If a complete test set is desired, after exiting Procedure 1, we call backup Procedures 2 and 3 (set x=2 for Procedure 2 and set x=1 for Procedure 3) to target the scan cells that have not been successfully targeted by Procedure 1. If Procedure 1 fails to generate a test pattern to differentiate scan cell N from all the cells downstream of it, Procedure 2 with x=2 tries to generate a test to differentiate cell N from N-2, N-3, ..., 0 and Procedure 3 with x=1 tries to generate a test to differentiate cell N from cell N-1. So the two additional procedures can help to generate a complete test set when Procedure 1 fails to target some scan cells.

4.6 Experimental Results

Given an upper bound (UB) and a lower bound (LB) of candidate suspect scan cells, we implemented the test generation flow shown in Figure 19. If there are no initial diagnosis results available, the upper bound (UB) is the input cell and the lower bound (LB) is the output cell of the faulty scan chain.

We start with scan cell UB using Procedure 1 until we reach scan cell LB. Then for those scan cells that could not be differentiated by Procedure 1, we call Procedure 3 to differentiate scan cell N from scan cell N-1 and call Procedure 2 to differentiate scan cell N from the downstream cells of cell N-1.

We evaluated the proposed diagnostic test generation procedures on 5 industrial circuits. The profiles of these circuits are given in Table 6. The last column of Table 6 shows the longest length of the scan chains in each design. Please note that Design 4 is configured with test compression logic EDT [Rajski 06].

For the first 3 designs, we first perform diagnosis using production tests. We injected more than 500 single stuck-at faults into randomly selected scan cells of these designs and obtained fail logs. For most cases, the diagnosis based on production tests report no more than 3 suspects. 50 cases have diagnostic resolution of more than 3 suspect scan cells using production test patterns. Based on the upper bound and lower bound from diagnosis results using production tests, we then applied the proposed test generation flow and the flow from [Guo 07] to obtain diagnostic patterns. For about 2/3 of the 50 cases, both the diagnostic test generation algorithms were able to achieve diagnostic resolution of less than or equal to three suspect scan cells. We focus on the remaining 1/3 of the test cases where the procedures in [Guo 07] failed to achieve diagnostic resolution of less than or equal to three scan cells. Tables 7-9 show the experimental results for these cases.

For Designs 4 and 5, we generated test patterns for all cells in 4 longest scan chains for each design using the proposed test generation flow and the flow in [Guo 07]. For each scan chain, we injected 4 single stuck-at faults into randomly selected scan cells and performed diagnosis using the diagnostic pattern set generated for the faulty scan chain. The experimental results for these designs are shown in Tables 10 and 11. It should be noted that diagnosis using production tests was not done for Designs 4 and 5.

Hence the UB and the LB for the diagnostic test generation flow are the input cell and the output cell of the faulty scan chain.

Table 6: Design Profile

Circuits	#gates	#chains	chain length
Design 1	10K	2	251
Design 2	10K	3	152
Design 3	10M	83	11346
Design 4	15M	323	503
Design 5	2M	32	5200

Table 7: Diagnostic Test Generation for Design 1

Case	#OrigSus	[Guo 07]		Proposed Test Generation			
		#P	#Sus	#P	#Sus	Time	Dep
1	18	0	18	17	1	6	4
2	9	0	9	9	1	102	4
3	7	0	7	6	1	10	6
4	5	1	4	2	3	112	7
5	9	0	9	9	1	73	4
6	9	0	9	8	1	19	4
7	31	0	31	30	1	11	4

Table 8: Diagnostic Test Generation for Design 2

Case	#OrigSus	[Guo 07]		Proposed Test Generation			
		#P	#Sus	#P	#Sus	Time	Dep
1	12	0	12	11	1	7	4
2	7	0	7	6	1	1	2
3	12	0	12	11	1	8	4
4	4	0	4	3	1	1	4

Table 9: Diagnostic Test Generation for Design 3

Case	#OrigSus	[Guo 07]		Proposed Test Generation			
		#P	#Sus	#P	#Sus	Time	Dep
1	7	0	7	6	1	822	3
2	5	0	5	4	1	906	4
3	10	0	10	10	1	5878	5
4	11	0	11	10	1	3320	5
5	11	0	11	10	1	1881	5

In Tables 7-9, for each case, under #OrigSus", we list the number of suspect cells based on diagnosis using the production test patterns Next we list the number of diagnostic test patterns ("#P") and the diagnostic resolution ("#Sus") for the procedures in [Guo 07] and for the proposed procedure. The last two columns show the test generation times ("Time") in seconds by the proposed test pattern generation procedure and the maximum sequential depth ("Dep") of the generated test patterns.

As can be seen from column 3 in Tables 7-9, the procedures in [Guo 07] could generate in only one case an effective single-cycle pattern to improve the poor diagnostic

resolution based on production test patterns. All the sequential depths reported in the tables are either the same as those calculated by the SCOAP based algorithm given in Section 3.2 or are at most 2 higher than the SCOAP calculated sequential depth. From the results we can see that for all the cases where the procedures in [14] failed to improve the diagnostic resolution to be <= 3 scan cells, the proposed sequential pattern generation method was able to improve the diagnostic resolution to be <= 3 scan cells and in each case contained the actual defective cell.

Procedure 1 successfully targeted every scan cell in the suspected range for 3 out of 7 cases in design 1, for all the cases in design 2 and for 4 out of 5 cases in design 3. Thus, Procedure 1 improves the success rate of generating a test pattern to differentiate a target scan cell from all cells downstream to it. In the cases where Procedure 1 can successfully target each scan cell in the suspected range, Procedure 2 and 3 are not needed.

Procedures 2 and 3 are applied when Procedure 1 fails to target some scan cell. For example, in case 2 of design 1 the diagnosis based on the original production patterns reported 9 suspects. The procedures in [Guo 07] generated no patterns and hence the number of suspects could not be reduced. The proposed Procedure 1 successfully generated 7 patterns for 7 scan cells and failed to generate a pattern for one scan cell, which means Procedure 1 cannot generate a test to differentiate one scan cell, say N, from all the downstream cells of it. We then call Procedures 2 and 3 to generate 1 pattern each for this scan cell. The test generated by Procedure 2 can differentiate cell N from cell N-2, N-1, ..., 0 and the test generated by Procedure 3 can differentiate cell N from cell N-1. So a complete test set is still obtained. The total number of patterns is 9 and the

number of suspects was reduced to one scan cell that was the actual defective cell. The maximum sequential depth of the 9 patterns is 4 as listed in the last column.

In Tables 10 and 11, for each case, we list the time for test generation for the entire scan chain, the number of diagnostic test patterns ("#P") and the average number of suspect scan cells ("ave sus") for the procedures in [Guo 07] and for the proposed procedures. The average number of suspect scan cells is the average over all the injected faults in the faulty scan chain.

Table 10: Diagnostic Test Generation for Design 4

Scan Chain	[Guo 07]			Proposed Test Generation		
	Time	#p	ave sus	Time	#p	ave sus
1	46.0h	485	1	2.0h	21	1
2	53.7h	518	1	5.1h	46	1
3	47.8h	486	1	4.0h	32	1
4	48.6h	330	1	20.4h	20	1

Table 11: Diagnostic Test Generation for Design 5

Scan Chain	[Guo 07]			Proposed Test Generation		
	Time	#p	ave sus	Time	#p	ave sus
1	14.6h	4233	1.8	2.4h	307	1.8
2	14.2h	4105	1.5	2.6h	243	1.5
3	12.8h	4172	1	2.8h	863	1
4	10.8h	4192	1.5	1.4h	884	1.5

As can be seen from Tables 10 and 11, the proposed diagnostic test generation flow reduces the run times by up to 20X. This makes the diagnostic test generation method much more practical for very large designs. Since in the proposed flow, one diagnostic test pattern simultaneously targets many more scan cells instead of targeting only one scan cell, we can also see the numbers of test patterns reduce dramatically, which reduces the test cost as well saving test application time. The average numbers of suspect scan cells for each case are the same for the proposed method and the method of [Guo 07]. Thus the quality of test patterns generated by the proposed method is not compromised.

For example, in case 2 of design 4, the run time for generating diagnostic patterns for the entire scan chain reduces from 53.7 hours to 5.1 hours and the number of test patterns reduces from 518 to 46. The average diagnostic resolutions are 1 for both the methods.

CHAPTER 5. DIAGNOSIS OF MULTIPLE PHYSICAL DEFECTS USING LOGIC FAULT MODELS

5.1 Introduction

Successful defect isolation relies heavily on the guidance from logic fault diagnosis and will depend even more for the future technologies. In this work, physical defects include interconnect opens, open vias, bridges/shorts, or opens and bridges inside a design library cell. Each such physical defect may manifest as faults at multiple sites. For example, an interconnect open defect may present itself as multiple input pin fault of several gates. Furthermore multiple defects are likely to be present in a CUD [Aitken 97].

Diagnosis of multiple defects has been investigated by several researchers [Boppana 99 - Yu 08, Tsai 09]. Prior to these studies it was customary to perform diagnosis using what are called SLAT patterns [Huisman 04, Bartenstein 01]. SLAT patterns are tests that fail a CUD on the tester and the observed response is matched by at least one single stuck-at fault. Methods that use SLAT patterns have been both effective and efficient when single defects such as opens and bridges that manifest as multiple faults are present [Huisman 04]. However such methods tend to be less effective as defect density increases significantly.

With the exception of the work in [Yu 08, Ye 10], the diagnosis procedures compare the simulated responses of circuits with single or multiple stuck-at faults. These procedures have been successfully used for defect diagnosis in industrial designs. Such methods were also shown to permit what has been called direct diagnosis of defects in circuits using test response compactors [Cheng 04]. Direct diagnosis refers to using the compacted responses for diagnosis as opposed to using responses obtained from scan

chains bypassing the response compactor. Direct diagnosis has the advantage that it does not require additional logic on the CUD to bypass test response compactors and it also facilitates volume diagnosis since test flows identical to production test of manufactured devices can be used. The method in [Yu 08] does not use simulation of faults and hence its run time can be independent of the number of defects in contrast to that for the methods such as in [Huang 01-Lin 07] which perform single or multiple fault simulations todiagnose CUDs with multiple defects. However, the run time it takes to eliminate false sites using per-pattern X simulation can be long and the effectiveness of X simulation pruning false sites can be limited [Tsai 09]. Also the effectiveness of the diagnosis procedure in [Yu 08] for direct diagnosis of designs with test response compactors has not been studied.

Another recent work that uses simulation of single stuck-at faults but does not restrict itself to the use of SLAT patterns is by Holst and Wunderlich [Holst 07]. This method has been shown to yield good diagnostic resolution even when the test responses are highly compacted [Holst 09]. The effectiveness of this method has been demonstrated for the case of single but complex defects such as wired AND bridges, cross talk faults etc. [Holst 07, Holst 09].

One latest method proposed in [Ye 10] does not use fault simulations to identify faulty locations as most diagnosis methods. The method constructs a fault-tuple equivalence tree (FTET) for each failing pattern and assigns a score to each fault in the construction. It only performs fault simulation on fanout branches when necessary, in order to recover passing bit mismatch.

In this work, in the first phase, we proposed a diagnosis method based on SLAT patterns and bit-union technique. This method achieves better diagnosis compared with previous diagnosis methods. The method is described in Sections 5.2. Then in the second phase we proposed a new diagnosis method to further improve the diagnosis quality using fault-tuple equivalence tree. It is described in Section 5.3.

While most previous works on multiple fault diagnosis focus on identifying the fault locations, we not only identify the fault locations, but also identify the physical fault types. The physical fault types considered in this work include interconnect open faults, bridge faults, and cell internal faults. The physical fault type identification is important in three aspects: (1) for failure analysis purpose, accurate physical fault type identification provides better guidance for FA engineers to locate a defect and (2) identifications of physical fault types can also help locate the faulty locations more accurately. In [Venkataraman 00] it was shown that using the interconnect models can avoid limitations due to the stuck-at fault equivalences at gates on resolution of interconnect opens defects. However, other physical defect types and the presence of multiple physical defects were not included in [Venkataraman 00]; (3) for yield improvement based on large volume diagnosis, accurate physical fault type identification provides better source data for root-cause analysis based on statistical algorithms.

5.2 Diagnosis Method Based On SLAT

5.2.1 Preliminaries

5.2.1.1Failing Pattern Types

When multiple faults exist in a circuit, failing patterns can be classified into three types. For the simplicity of explanation, let's assume that only two faults are present in a circuit as shown in Figure 20[Holst 09].

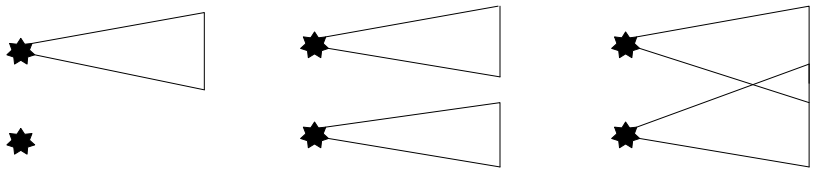


Figure 20: Failing Patterns Types

Type 1: Type 1 failing patterns only activate one fault and propagate its faulty effect to observation points (primary outputs or scan cells). Type 1 patterns are also known as Single-Location-At-a-Time (SLAT) patterns [Huisman 04, Bartenstein 01].

Type 2: Type 2 failing patterns activate multiple faults and propagate their faulty effects to different observation points. Their effects do not interact.

Type 3: Type 3 failing patterns activate multiple faults and propagate their effects to observation points. There are interactions between the propagation paths of the activated faults.

5.2.1.2 Diagnostic Metrics

The fault simulation signature is a set of failing bits caused by the simulated fault. In this paper we use the following diagnosis metrics [Holst 07, Holst 09, Venkataraman 00] to evaluate how good a suspect fault signature matches the test failures:

SFTF: The number of failing observation points (failing bits) that appear in both the fault simulation (FM) signature and the failures observed on device under diagnosis (DUD).

SFTP: The number of observation points that fail in the fault simulation signature but are not observed in tester failures.

SPTF: The number of observation points that appear in the test failures but not in the fault simulation signature.

For a physical fault, its diagnosis metric calculation is based on the stuck-at faults that constitute the physical fault. The details on how a physical fault is composed and how their score is calculated are explained in Section 5.2.2.2.

5.2.2 Multiple Physical Defect Diagnosis

Figure 21 shows the overall flow of the proposed diagnosis procedure.

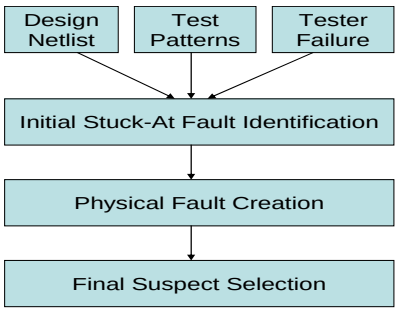


Figure 21: Diagnosis Flow Overview

5.2.2.1 Initial Stuck-At Fault Identification

First the diagnosis procedure processes each failing pattern to find any single logic location (stuck-at fault) that can explain this failing pattern. For each failing pattern, the path-tracing procedure [Venkataraman 97] starts from each failing observation point and traces back in the circuit towards primary inputs and memory element output. Then each single location in the intersection of all the cones is simulated to compare with the observed failures for the failing pattern. If any location's fault simulation signature matches the failures of this pattern, the location is put on the initial fault list and this failing pattern is called a SLAT pattern.

After the procedure processes all the failing patterns, the initial fault list, which includes all the faults that explain all the SLAT patterns, is constructed. The failing patterns that cannot be explained by a single location are non-SLAT patterns and in the worst case there may be no SLAT patterns, especially when defect density is large in the faulty circuit. In the proposed diagnosis procedure, for each non-SLAT pattern we obtain all the faults in all the traced-back cones of the failing observation points. These faults are then put on the initial fault list. Thus the initial fault list includes faults obtained by using SLAT and non-SLAT patterns.

Next we attempt to identify Type 2 (cf. Figure 20 (b)) failing patterns among the non-SLAT patterns by a procedure called *bit union*. In the bit union process, for each non-SLAT pattern we try to find a set of faults from the initial fault list, the union of whose faulty outputs can explain this failing pattern. For example, in Figure 22, for a non-SLAT failing pattern, say *t*, the bit union procedure finds two faults *f1* and *f2*. If the faulty machine simulation of single faults *f1* and *f2* finds that each fault explains a subset

of the failing observation points and the union of the fault effects of the two faults matches the observed failure of this pattern, then we say that faults *f1* and *f2* partially explain the failing pattern *t*. As we can see, on failing pattern *t*, *f1* and *f2* both have (SFTF>0, SFTP=0).

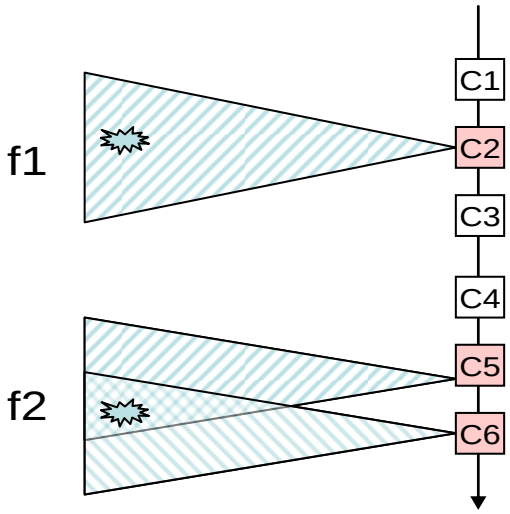


Figure 22: Example of Bit Union

After this step, the diagnostic metrics, i.e. SFTF, SFTP and SPTF, have been calculated for each fault on this list.

5.2.2.2 Physical Defect Creation

In this subsection, we explain how a group of logic faults are selected to build a single physical fault. A physical fault is composed of several single stuck-at faults based on the relation of their locations and the logic simulation values of the stuck-at signals. The diagnostic metric of a physical fault is calculated based on the diagnostic metrics of the constituent stuck-at faults.

We use the SFTF and SFTP metrics calculated using failing patterns to derive a single physical fault from a group of logic faults. For example, when an open defect affects an interconnect, the faulty behavior of the defect could be complex. Figure 23 shows a net-open physical defect. The net has a stem Z and four branches A, B, C and D. As shown in Figure 23, faulty location A explains failing patterns 1, 3 and 7; location C explains failing patterns 2 and 6; location D explains failing patterns 4 and 5; Locations B and Z do not explain any failing patterns. In [Venkataraman 00], the authors proposed to take the union of all the failing patterns that can be explained by the stem or any branch connected to the stem to create a superset of possible faulty behaviors of the open defect on the net. That is, the derived net-open fault at Z will explain the union of all the above failing patterns. That means net-open fault at Z explains failing patterns 1 through 7. The idea behind this method [Venkataraman 00] is to roughly predict the faulty behavior of a real open defect by taking the union of all the explained failures. However, it could be too aggressive, thus wrong net-open faults could be constructed and reported using this method in some cases. In this paper we propose to derive a physical fault from a group of logic locations when multiple defects are present in the circuit. It is worth noting here that for diagnosis methods that only identify logic locations without trying to identify the physical fault types, location A could be ranked higher than location Z in the example above if stuck-at fault at A explains more failures than stuck-at fault at Z. It still pinpoints the faulty location, however all the gate equivalence faults of logic location A will also be reported. If we can identify the fault types correctly and build a physical defect at Z, we will only report a net-open fault. Identifying the physical defect type and predicting the faulty behavior of the physical defect based on failures explained by constituent logic

locations can break the ambiguity introduced due to equivalence of gate faults, thus helping identifying the faulty locations more accurately.

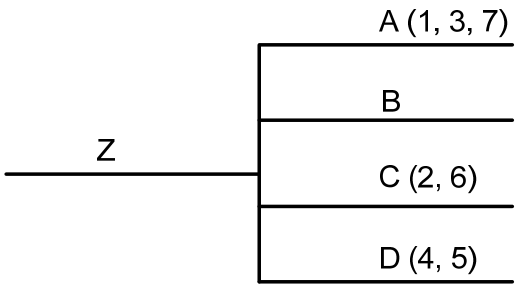


Figure 23: Net-open Defects

Figure 24 shows an example of multiple interconnect open defects. Assume interconnect Z is one of the open defects in the faulty circuit and Y doesn't have an interconnect open defect. The open defect on Z causes a SA0 fault on D and due to the equivalence of faults at a gate, SA1 on G is also a faulty location on the initial fault list. If H and K are on the propagation paths of another defect and are on the initial candidate list, the union of stem Y and all its branches may explain more observed failures than the union of stem Z and all its branches. Since location D and G explain the same failures due to fault equivalence, and if location H explains exactly the same number of failing patterns as location B, then vector-wise prediction (pattern-level fail match) [Venkataraman 00] is not able to differentiate composite fault Z and Y, and the union of the failures explained by logic location Y, G, H and K could be more than the union of failures explained by logic location Z, A, B, C and D. This will lead to the wrong callout of net Y instead of calling out the real fault Z. Actually when multiple defects are present,

especially when a large number of locations fail, the chance of vector-wise match decreases and there may be no vector-wise match. Then in the example above it could also lead to wrong callout.

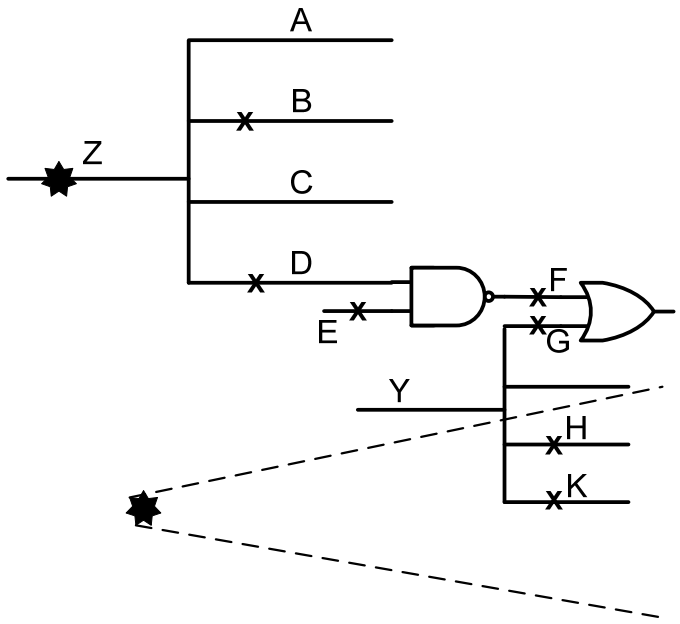


Figure 24: Illustration of Diagnosis Challenge

From the above example, we can see that while building a physical fault from a group of logic locations, simple union of all the faults may lead to wrong diagnosis. From our observation, careful selection of component stuck-at faults and careful analysis of the diagnostic metrics of the composed physical fault improves diagnosis quality by removing false locations. As we can see in the example shown in Figure 24, if both or either one of locations H and K is excluded when deriving the net-open defect Y from stuck-at faults, the real defect Z could be selected instead of Y.

In the following, we will explain how to analyze the relations among logic locations and diagnostic metrics of each component stuck-at fault during physical fault creation phase. The selection of component stuck-at fault is motivated by the work done by Holst and Wunderlich [Holst 07]. In [Holst 07], the authors proposed a metric σ_T that uses *min* (sftf, sftp) for each failing pattern to rank stuck-at faults. It has been shown to be very effective to rank stuck-at fault suspects. In this work, we use a modified metric σ'_T to compute a score for each derived physical fault.

To derive a physical fault, we check the list of initial faults and try to find faults that can be grouped together to create physical faults. The faults on the same interconnect net may compose a net-open fault as shown in Figure 23. The faults on the input and output nodes of the same gate may compose a logic cell internal fault [Sharma 07]. Bridge faults are derived from two or more logic locations that meet certain excitation and propagation criteria [Zou 06].

After a physical fault is created, the next step is to calculate its diagnostic metric values. In [Holst 07] the metric $\sigma_T = \sum_{t \in FP} \min(sftf, sftp)$ was proposed to be used as a ranking criteria.

We propose to use the metric $\sigma'_T = \sum_{t \in FP-1,2} \min(sftf, sftp)$ to calculate diagnostic metric values for the derived physical fault. To get σ'_T , *min*(sftf, sftp) is added over all the non-Type-3 (Type-1 and Type-2) failing patterns, which can be explained by a single location or by bit union of several locations. If σ'_T of a fault is 0, we call such a fault *seed fault*. While the previous works consider all the failing patterns explained by each component stuck-at fault as also explained by a composed physical fault, we only consider the failing patterns explained by the seed faults as explained by a composed physical fault. That is,

if a stuck-at fault, say *f*, has both SFTF and SFTP on any non-Type-3 failing pattern, thus leading to $\sigma'_T>0$, the SFTF of this fault is not counted in the SFTF of the derived physical faults composed from *f* and other faults. The reason for using this heuristics is that a fault with SFTF>0 and SFTP>0 for one failing pattern cannot explain this failing pattern unless this pattern is a Type-3 pattern. As this pattern can be explained by a single location (Type-1 pattern) or by bit union procedure (Type-2 pattern), most likely, this fault is not a real fault. Including this fault usually leads to a false suspect in many of the cases that we have studied. σ'_T doesn't include *min*(sftf, sftp) over a failing pattern if this pattern cannot be classified into Type-1 or Type-2 patterns. When multiple faults are present, a real fault could have both SFTF and SFTP over some failing patterns if the fault has interactions with other faults on the observation points as shown in Figure 20(c). The chance of interactions increases as fault density increases. To add *min*(sftf, sftp) over all the failing patterns can be too pessimistic for a real fault when multiple faults are present, and hence we add *min*(sftf, sftp) over only Type-1 and Type-2 failing patterns.

We use the example shown in Figure 23 to explain how to calculate the diagnostic metrics for a physical fault. Suppose location D (a stuck-at fault on D) has both SFTF and SFTP on a failing pattern *p*.

If the failing pattern *p* can be explained by a single logic location or by bit union of a set of locations, then D is not a seed fault ($\sigma'_T>0$) and we do not include the failures explained by location D when computing the SFTF of the derived net-open fault. So the number of failing patterns that net-open fault on Z explains is 5: failing patterns 1, 3, 7, 2 and 6.

If the failing pattern *p* can neither be explained by a single logic location nor by bit union, then the pattern is classified as a Type 3 pattern and D is still a seed fault (σ'_T=0). The SFTF of location D will be counted in the SFTF of the net open fault. So the number of failing patterns that the net-open fault on Z explains is 7: failing patterns 1-7.

Similarly, for the example shown in Figure 24, if false location H has both SFTF and SFTP on the same failing pattern, which can be explained by a single location or a set of locations, then for the derived open fault on interconnect Y, the failures that H explains won't be counted in the SFTF of the open fault Y. Then it will not lead to the wrong call out of open fault Y. This concept applies to other physical faults that are derived from a group of logic locations, such as bridge faults and cell internal faults in a similar manner.

5.2.2.3 Final Suspects Selection

In the earlier steps, we have constructed a list of faults that include all the stuck-at faults and the derived physical faults. We have also determined which failing patterns can be explained by each of the faults on the list. We define the number of patterns explained by a fault "#EFP". Now we choose a subset of these faults to explain the observed failures of the CUD. We use a greedy set covering algorithm to select the final suspect(s) iteratively. In order to use the passing pattern information to prune some false suspects that happen to explain many failing patterns, we calculate the SFTP values for passing patterns at this step. When a fault has SFTP on a passing pattern, we say that this pattern is a passing mismatch pattern for this fault. We define the number of passing mismatch patterns for a fault "#PMP". Previous works also use passing pattern information to rank suspects. However, passing mismatch information is only considered when failing match

(#EFP or SFTF) is exactly the same. In this procedure, a diagnosis score is calculated and assigned to each stuck-at and physical fault. The diagnosis score is based on the number of explained failing patterns (#EFP) and the number of passing mismatch patterns (#PMP):

$$\text{Diagnosis Score} = \#EFP - \alpha \times \#PMP \quad (1)$$

where α, $0 \leq \alpha \leq 1$, is a parameter.

So the passing mismatch information is also considered even if two suspects have different failing matches. During the set covering process, in each iteration we choose the suspect with the highest diagnosis score. Once a suspect is chosen, the failing patterns that are explained by the suspect are eliminated from the tester failure set. Table 12 shows an example of the set covering procedure. There are 10 test patterns and 4 candidate suspects. The first 6 patterns are failing patterns and the last 4 patterns are passing patterns. If a suspect explains a failing pattern, we mark the corresponding entry with "√" and if a suspect has mismatches on a passing pattern, we mark the entry with "×".

Table 12: An example of suspect selection (α=0.4)

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P 10
S1	√		√	√	√			×	×	×
S2	√		√	√						
S3		√			√	√				×
S4		√			√					

In the first iteration, the diagnosis score for suspect S1 is 2.8 as computed using equation given above; the diagnosis score for suspect S2 is 3; and the diagnosis scores for

suspects S3 and S4 are 2.6 and 2, respectively. So suspect S2 is selected in this iteration. Since P1, P3 and P4 are explained by S2, these three patterns are dropped from consideration in future iterations and the table is updated as shown in Table 13.

Table 13: Updated table

	P2	P5	P6	P7	P8	P9	P 10
S1		√			×	×	×
S3	√	√	√				×
S4	√	√					

Using similar calculation, in the second iteration, suspect S3 is selected. Now all the failing patterns have been explained by set {S2, S3}.

5.2.3 Experimental Results for Multiple Physical Defect

To evaluate the effectiveness of the proposed diagnosis technique, we conducted experiments on simulated test cases of several industrial designs and also on 4 available silicon test cases. In each simulated test case, we inject two or three physical faults simultaneously. We then perform fault simulation of the injected physical faults to create a failure file. This failure file is fed to the diagnosis tool and check whether the injected fault can be identified.

The faulty behavior of a cell internal and bridge defect is created based on SPICE simulation of a defect injected at transistor level net list. The faulty behavior is captured into a truth table. A truth table contains the faulty outputs' values for all the input value combinations. We then use logic simulation of the circuit by injecting the truth table behavior of single or multiple defects to derive the failures. The faulty behavior of a net-

open defect is created by injecting stuck-at faults on a subset or all of the branches connected to the defective stem. For some nets, the number of failing branches can be up to 10. So for a test case which has three net-open faults, the actual number of faulty logic locations are up to 30. The characteristics of the four industrial designs are shown in Table 14. The diagnosis results on the four designs are reported in Table 4 and Table 5. Designs 1, 3, 4 use on-chip EDT [Rajski 04] test data compression with input compression and output compaction logic.

Table 14: Design Characteristics

Circuits	#gates	#chains	chain length	#output channel	compaction ratio
Desig	9.3K	24	19	3	8X
Desig	2M	32	439	N/A	N/A
Desig	2M	32	439	2	16X
Desig	9.8	6012	152	65	92X

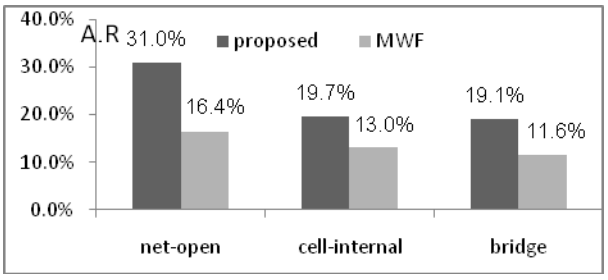


Figure 25: Comparison for Two Defects

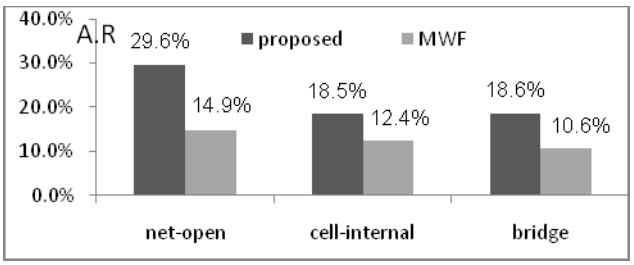


Figure 26: Comparison for Three Defects

Diagnosis accuracy is a measure of how many injected defects are correctly identified. In tables 15 and 16, the total numbers of test cases for each fault type are shown under defect types. In Table 15, the row heading of "Both" show the number of cases for which both the injected defects are reported while the rows headed with "One" or "None" show the number of cases that one or none of the injected defects are reported. Similarly in Table 16, the row heading of "All" show the number of cases for which all the three injected defects are reported while the rows headed with "Two", "One" or "None" show the number of cases that two, one or none of the injected defects are reported. For example, in Table 15, for net-open defects, out of 60 cases for Design 2, we can report both open defects for 58 cases. The missed diagnosis may be due to the following. When multiple defects are present, defective global signals such as defects on scan enable or clock tree can explain the failures with small passing mismatch in few cases and injected defects are not reported in such case. Also the diagnosis accuracy is bounded by the production test set used to do diagnosis. For example, under some test set, one fault could be dominated by other faults and cannot be reported.

Table 15: Experimental Results for Two Physical Defects

Design / Result	Design 1			Design 2			Design 3			Design 4		
	Open (55)	Cell (124)	Bridge (63)	Open (60)	Cell (100)	Bridge (60)	Open (60)	Cell (100)	Bridge (60)	Open (40)	Cell (60)	Bridge (50)
Both	48	116	33	58	95	40	58	93	39	36	49	36
One	5	5	29	1	4	19	1	5	20	4	7	12
None	2	3	1	1	1	1	1	2	1	0	4	2
A.R	30.6%	19.0%	15.4%	32.1%	20.7%	20.3%	31.9%	20.1%	20.0%	28.7%	19.0%	21.1%

Table 16: Experimental Results for Three Physical Defects

Design / Result	Design 1			Design 2			Design 3			Design 4		
	Open (50)	Cell (50)	Bridge (50)	Open (50)	Cell (50)	Bridge (50)	Open (50)	Cell (50)	Bridge (50)	Open (50)	Cell (50)	Bridge (50)
All	43	40	29	45	41	34	44	41	30	41	41	33
Two	3	5	12	3	3	9	4	2	12	6	4	9
One	3	3	7	2	4	4	2	5	5	3	3	6
None	1	2	2	0	2	3	0	2	3	0	2	2
A. R.	28.7%	17.5%	15.1%	30.6%	19.8%	20.0%	30.9%	18.8%	19.7%	28.0%	17.9%	19.7%

Diagnosis resolution shows how many faults are reported by the diagnosis tool. The diagnosis resolution for an injected fault is defined as the reciprocal of the number of the reported faults, whose scores (rankings) are larger (higher) than or equal to the injected fault, except other injected faults. For example, in a test case injected with two net-open defects, say F1, F2, the proposed method reported five suspects with scores as {F1 (100), F3 (100), F4 (95), F2 (95), F5(90)}, then the resolution for F1 will be 50% and the resolution for F2 will be 33%. Please note that when we calculate diagnosis resolution, we include all the un-collapsed stuck-at faults. When a logic location is reported with two fault types, we consider them two suspects. For example the same location can be identified net-open and cell-internal defects at the same time under a test

set. In such cases, we consider them two suspects when calculating diagnosis resolution. In Tables 15 and 16 we list the average resolution for all types of defects for each design under columns “AR”.

For the purpose of comparison, we implemented the location based diagnosis method of [Holst] and we call this method MWF. MWF method ranks all the stuck-at faults in the circuit and MWF method yields a list of suspects with rankings. In the example shown above, in a test case injected with two net-open defects, say F1, F2, the MWF method ranked F1 first with 2 other logic locations, then the resolution for F1 will be 1/3= 33% and F2 is ranked 26 with 3 other logic locations, the resolution for F2 will be 1/(28-1)=4%. Figures 25 and 26 give the average diagnosis resolution for both methods. We can see that for all types of physical defects, the proposed method has better resolution than the MWF method. This shows that carefully deriving physical faults and calculating their diagnosis metrics can improve the diagnosis resolution for multiple physical defect diagnosis.

We also conducted experiments on four available silicon test cases of Design 4. Physical failure analysis (PFA) has identified one defect for each of the 4 test cases. For 3 cases, the defect type is net-open and for 1 case, the defect type is cell-internal. The proposed diagnosis method reported that there are multiple faults in each of the four cases. And the diagnosis results include the identified real defect for all the 4 silicon cases. The real defects are reported top suspects for all the 4 cases and their defect types are also correctly identified.

5.3 Diagnosis Method Based on FTET

In Section 5.3, we will investigate a method based on fault-tuple equivalence tree (FTET). When multiple faults are present in faulty circuits, the interactions between them on observation points complicate the diagnosis. A new method of diagnosis of multiple arbitrary faults was recently proposed [Ye 10]. This method constructs a fault-tuple equivalence tree (FETE) for each failing test pattern to store the relation among potential fault locations to consider the mask and reinforcement effect. If the faulty effect can be propagated to at least one observation point in the single-fault situation, but be blocked in the multiple-fault situation, then it is called mask effect. If the fault effect cannot be propagated to an observation point in the single-fault situation, but can be observed in the multiple-fault situation, then it is called reinforcement effect. Instead of doing fault simulations on all potential fault locations, the method only perform fault simulation on fan-out branches to check passing bit mismatches. The experimental results showed good diagnosability and resolution compared with the latest work[Yu 08]. During the tracing back from the failing observation points and the build-up of the FTET, the method tries to consider masking and reinforcement effect, which is very important for diagnosis of interacting multiple faults. In Section 5.3, we will investigate a method to diagnose multiple faults using fault-tuple equivalence tree. We propose to use conflict between different faults to further improve diagnosis quality and also we do not only identify fault locations, but also identify fault types as the method proposed above. The overview of the diagnosis method is shown in Figure 21 above in Section 5.2.2. However, in this method, we propose a new approach in Step 2 to identify initial stuck-at faults.

5.3.1 Analysis of Diagnosis Method in [Ye 10]

In Section 5.3.1, we review the method proposed in [Ye 10] and also give the terminologies used in this work.

For each failing pattern, the method traces from all failing observation point towards the primary input to construct a FTET. The tracing for each failing patterns starts from the first fault-tuple, which are all the failing observation points. For example, in Figure 27, the circuit under pattern abc(101) has two failing observation points *g* and *h*, so the first fault-tuple is (*g/1, h/1*), where *g/1* represent a fault which causes the line g to get a faulty value 1. The tracing starts from the first tuple. A score is assigned to each node in the tree. After a FTET is constructed for each failing pattern, the sites are selected based on the score until all the FTETs are pruned.

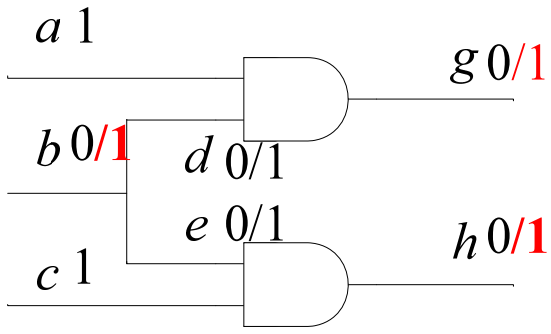


Figure 27: A Faulty Circuit under Pattern abc(101)

In Figure 28, it shows the overview of the diagnosis method.

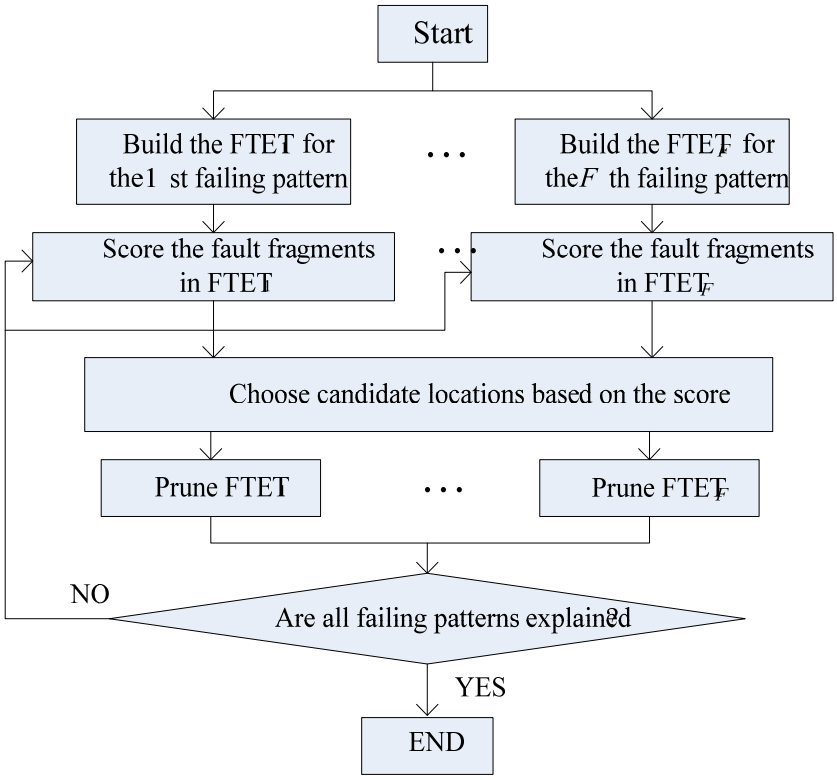
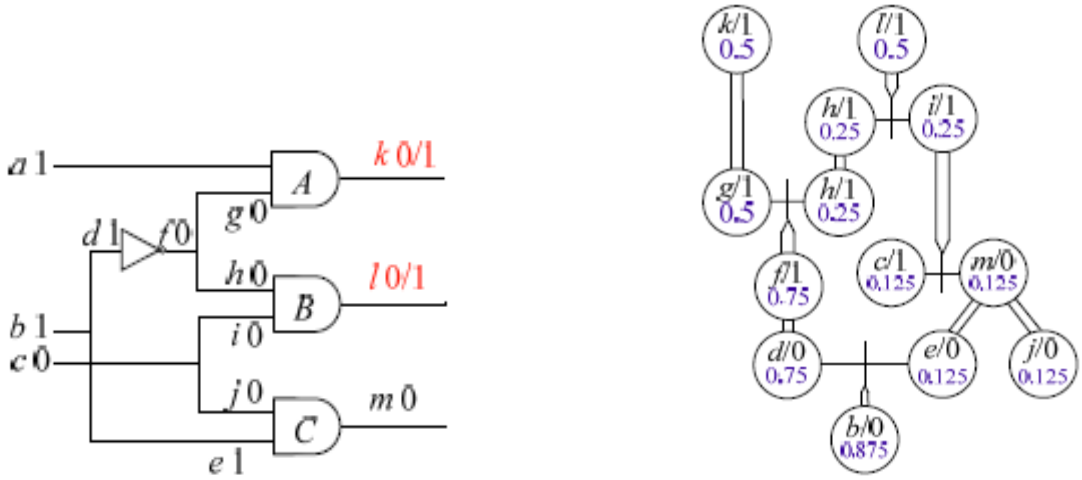


Figure 28: Diagnosis methodology overview [Ye 10]

A simple example in Figure 29 is given next to explain how to construct a fault-tuple equivalence tree (FTET) for a test pattern abc(110). The first fault-tuple is all the failing observation points, which are (*k/1, l/1*). If there are n failing observation points, each node will be assigned the same 1/n score. In this example, n equals 2. So both *k/1* and *l/1* has score 0.5. During the process of path-tracing, there are two situations:

(1) The line under trace is an output of a gate and not a fanout branch of a stem. In this case, they build the equivalent relation between the fault at the output of the gate and the faults at the inputs of the gate. For example, in Figure 29(1), because faulty-free-value (FFV) of line a is 1 and FFV_b=0, *k/1* is equivalent to *g/1*. The two nodes will have the same score 0.5. For *l/1*, because FFV_h=0 and FFV_i=0, *l/1*is equivalent to the fault

tuple (*h/1, l/1*). *h/1* and *l/1* will both have score 0.25. Similarly, a fault of an output could be equivalent to anyone among several faults of its inputs. For example, when a two-input AND gate has a faulty value 0 at its output, say *c*, if FFVs of its inputs, say *a* and *b*, are both 1, *c/0* will be equivalent to either *a/0* or *b/0*. *a/0* and *b/0* will both have the same score as *c/0*.



(1) (2)

Figure 29: Example of FTET Construction [Ye 10]

(2) The line under trace is a fanout branch of a stem. In this case, there are two situations. If all the other branches of the same stem are traced, a relation can be built at the stem. For example, in Figure 29(1), when the trace reaches line *g*, it is found that all the fanout branches of the same stem are traced. Then (*g/1, h/1*) is equivalent to *f/1*. *f/1* will have the union of the scores of (*g/1, h/1*). If not all the other branches of the same steam are traced, fault simulation will be performed. For example, in Figure 29(1), when the trace reaches *i/1*, since line *j* is not traced, *j/1* will be injected to do fault simulation. If there is mismatch on any passing observation points, *i/1* will be equivalent to the fault

tuple, which include the fault at the stem and the fault at the mismatching passing observation points as shown in Figure 29(2).

After construction of FTETs for all the failing patterns, the method will select sites iteratively to prune all the trees based on the scores. In each iteration, after a site or several sites are selected, the FTETs will be pruned and the tree will be updated. The site selection ends when all the FTETs are pruned.

Next we will define conflicts and propose how to use conflicts to further improve the diagnosis quality for multiple faults.

5.3.2 Method to Identify Faulty Locations Based On FTET

5.3.2.1 Conflicts in FTET

We will first introduce the concept of conflicts and explain the method to use conflicts to diagnose multiple clustered faults.

Conflict: a node, say k/v_1, is said to have a conflict with a node, say h/v_2 if and only if k/v_1' is on the path from h/v_2 to the first fault tuple and h/v_2 cannot get the same score without going through k/v_1'. (v_1' is the compliment of v_1)

For example, in Figure 30 showing an FTET, the compliment of the blue nodes are said to have conflicts with the red node. Take GG15/B/1 as an example, it is said to have a conflict with GG14/0 because GG15/B/0 is on the path from GG14/0 to the first fault tuple and GG14/0 cannot get score 1 without going through GG15/B/0. GG16/B/0 is also on the path from GG14/0 to the first fault tuple, but GG14/0 can still get score 1 without going through GG16/B/0, so GG16/B/1 doesn't have a conflict with GG14/0. If a

node A is said to have a conflict with another node B, then node A somehow blocks the propagation path from node B to the faulty effects on failing observation points.

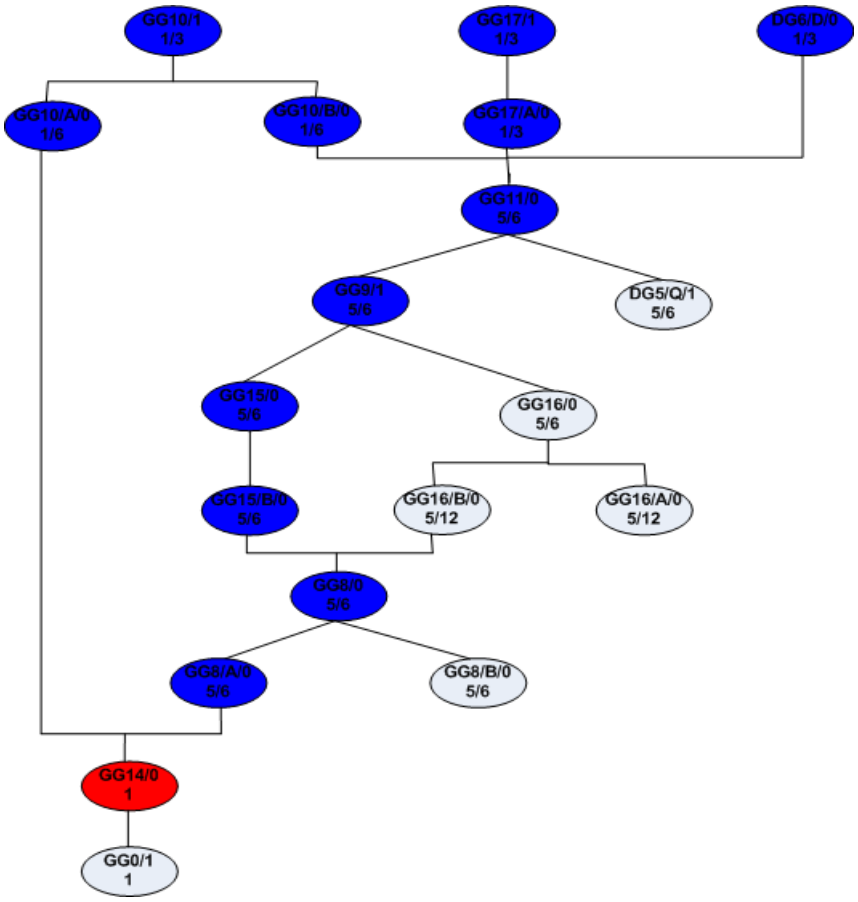


Figure 30: An example of conflict

5.3.2.2 Site Selection

We propose a method to choose candidate faulty site iteratively after construction of FTETs. In [Ye 10], the sites are also selected iteratively. In each iteration, the faults with the highest score in each FTET will be first chosen. Then among these faults, the line which appears most is chosen as one of the final suspect. The FTETs will be pruned and updated using the chosen final suspect. The later a suspect is chosen, the lower rank

it will get. So the suspect chosen in the first iteration will be the top ranked suspect. In this work, we identify conflicts after each iteration, then use conflicts to guide the site selection.

Let's take an example shown in Figure 31. The example shows a faulty S27 (ISCAS89 benchmark circuit) with 5 stuck-at faults: GG14/0, GG8/B/1, GG15/B/0, GG16/A/0, GG16/1. 4 out of 8 test patterns failed. So 4 FTETs will be constructed for all the 4 failing patterns. In Figure 31, it shows the faulty S27 under test pattern P1.

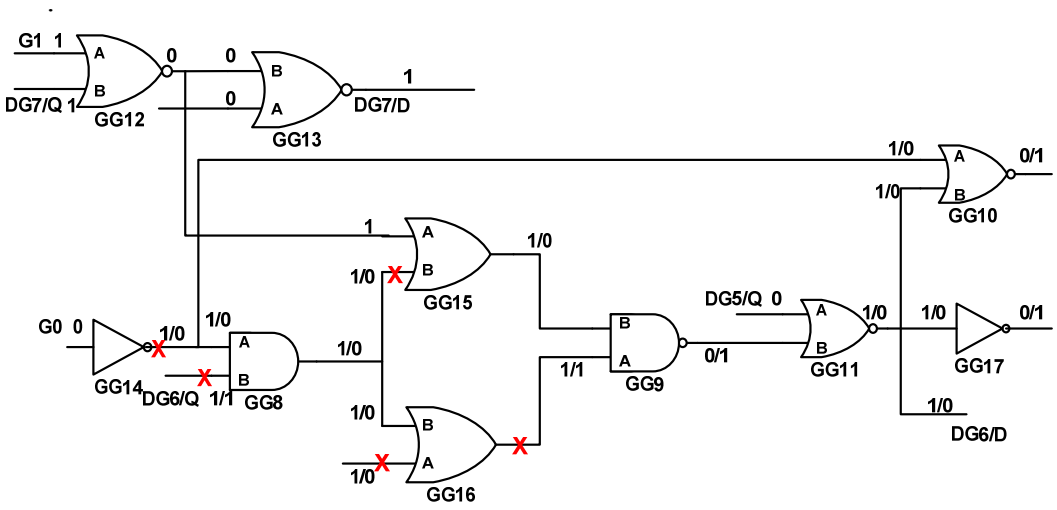


Figure 31: Faulty S27 Circuit under Pattern P1

The FTET for P1 is shown in Figure 31. We can see that GG14/0 (GG0/1) has the highest score in this tree. GG14/0 (GG0/1) has the highest score in 3 FTETs for P1, P2, P3. According to the site selection method in [Ye 10], they will be selected as one of the final suspect in the first iteration, i.e. the top suspect. GG14/0 (GG0/1) can explain the 3 FTETs completely so the 3 trees will be pruned completely by GG14/0 (GG0/1). Let's take a look the last FTET for failing pattern P4 shown in Figure 33.

In the second iteration of site selection, there is only one FTET left, as shown in Figure 33. 8 faults have the same score and are undistinguishable using the method in [Ye 10]. If we consider the nodes that have conflicts with previously selected final suspect GG14/0 (GG0/1), we can see that GG11/1, GG9/0 and GG8/1 have conflicts with GG14/0 (GG0/1). Considering conflicts in this example can distinguish real injected faults with some other candidate. Next we summarize the site selection method we propose. After FTETs are constructed for all the failing patterns, the diagnosis method enters into site selection flow as follows:

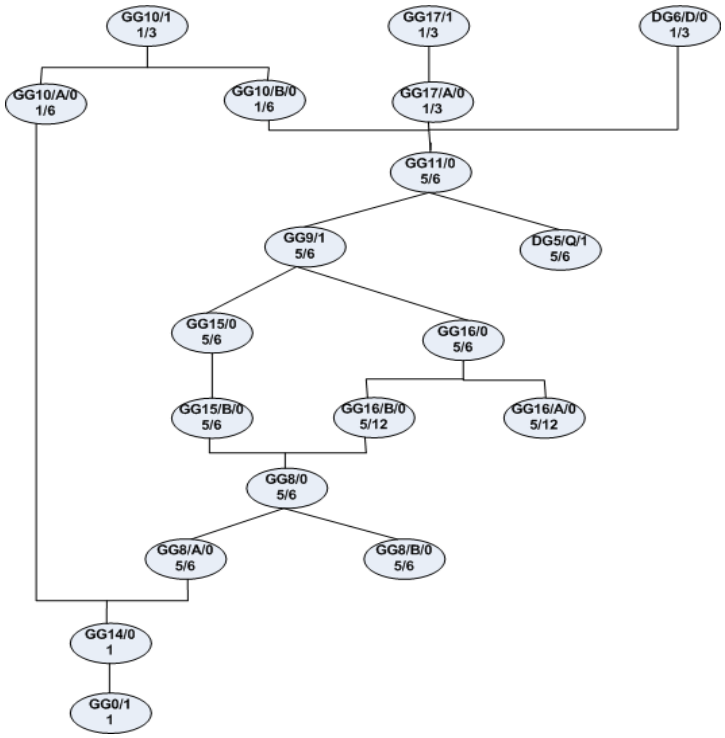


Figure 32: FTET of Pattern P1 for Faulty S27 Circuit

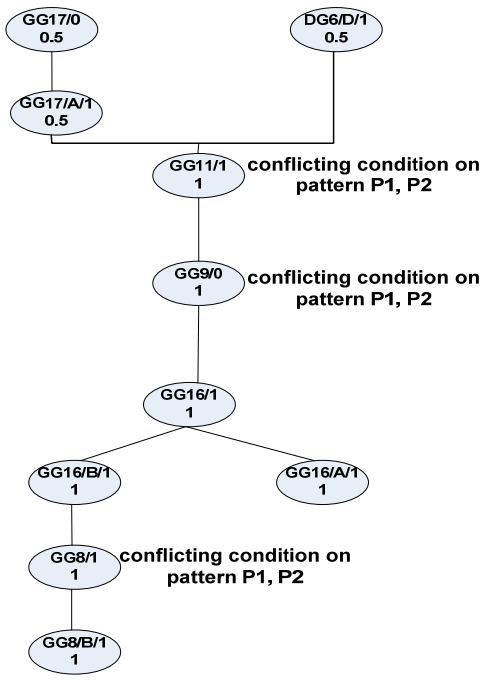


Figure 33: FTET of Pattern P4 for Faulty S27 Circuit

Step 1: *For each fault, sum up the scores over all the FTETs.*

Step 2: *Select the faults as final suspects with the highest score.*

Step 3: *Identify the nodes that have conflicts with the selected faults. Then subtract the scores in the FTETs, in which they have conflicts with the selected faults.*

Step 4: *Prune FTETs with the selected faults and update FTETs and scores.*

Step 5: *If all FTETs are pruned, site selection flow ends. Otherwise, go to Step 2.*

In the above example, GG14/0, GG8/B/1, GG16/1 will be included in the final suspect report. We know that GG14/0, GG8/B/1, GG15/B/0, GG16/A/0, GG16/1 are actually injected in the circuits. The diagnosability (S_D/S_I) for this case would be 60%, where S_D is the number of actual fault locations identified by diagnosis and S_I is the number of injected faults. However, if we inject GG14/0, GG8/B/1, GG16/1 into the circuits, the responses will be exactly the same as the faulty circuit with the 5 stuck-at

faults. This is called multiple fault equivalence. Diagnosability will be bounded by such equivalence.

Now we get a list of final suspects. There are the initial stuck-at faults. Next physical defects will be created in the same way as explained in Section 5.2.2.2. To derive a physical fault, we check the list of initial faults and try to find faults that can be grouped together to create physical faults. The faults on the same interconnect net may compose a net-open fault as shown in Figure 23. The faults on the input and output nodes of the same gate may compose a logic cell internal fault [Sharma 07]. Bridge faults are derived from two or more logic locations that meet certain excitation and propagation criteria [Zou 06].

5.3.3 Experimental Results for Multiple Physical Defects

We conducted experiments on full-scan combinational version of public benchmarks including ISCAS'89 circuits and ITC'99 circuits.

In Table 17, we first compare the proposed method with the method in [Ye 10]. As the method in [Ye 10] does not identify fault types and only report faulty locations in the final suspect list, in Table 17 we only report the suspect list we get after Step 2 in Figure 21, using the proposed method in Section 5.3. Conflicts happen when multiple faults are logic neighbors. So we inject multiple faults in a locally bounded region instead of inject random multiple faults. When a large number of faults are present in a locally bounded region, the diagnosis becomes more challenging as the chance of interactions between different faults increases. We inject multiple stuck-at, transition, dominant bridge and net open faults. For each number of injected faults, each fault type and each

circuit, we have 50 test cases. The number of failing patterns is 100 for all test cases in the experiments.

To evaluate the diagnosis quality, we define diagnosability as S_D/S_I and we also report the average resolution (A.R.) as defined in Section 5.2.3. S_D is the number of actual fault locations identified by diagnosis and S_I is the number of injected faults. The diagnosis resolution for an injected fault is defined as the reciprocal of the number of the reported faults, whose scores (rankings) are larger (higher) than or equal to the injected fault, except other injected faults. For example, in a test case injected with two net-open defects, say F1, F2, the proposed method reported five suspects with scores as {F1 (100), F3 (100), F4 (95), F2 (95), F5(90)}, then the resolution for F1 will be 50% and the resolution for F2 will be 33%. If a fault is not reported by the method, the resolution for this fault will be 0. For the total number of final suspects, as the diagnosis only reports faulty locations without fault type, we do not consider gate equivalence faults because they are undistinguishable. For example, a stuck-at-0 fault at an input of a AND gate will be equivalent to the stuck-at-0 fault at its output. They will be considered one fault in Table 17.

In Table 17, the first column is the circuit name, and the number of gates is listed under the circuit names. The second column is the type of injected faults (S: Stuck-at; T: transition; B: dominant bridge; O: net open), and the rest columns are the average diagnosability, the average resolution (A.R.). In Table 17, we can see that both diagnosability and the resolution get improved for multiple faults. For example, when there are 19 stuck-at faults in S38584 circuit, the average diagnosability improves from 60% to 62% while the average resolution increases from 14% to 15%. Please note that as

shown in the example in Section 5.3.2, the test response of a faulty circuit with 5 faults could be the same as a faulty circuit with 3 faults. So diagnosability we defined here is bounded by such multiple-fault equivalence. Also some faults could be undistinguishable under a test pattern set, this will reduce the resolution.

In Table 18, we report the diagnosis results using the whole diagnosis flow proposed in Section 5.3. It is under heading "New" in Table 18. The whole flow is shown in Figure 21 and it identifies fault types after we identify faulty location after Step 2. We also compare the diagnosis results with the earlier proposed method described in Section 5.2. It is under heading "[Tang 10]" in Table 18.

In each simulated test case, we inject two or three physical faults simultaneously. We then perform fault simulation of the injected physical faults to create a failure file. This failure file is fed to the diagnosis tool and check whether the injected fault can be identified. Please note that one physical defect can cause multiple faulty locations. For example an open net defect can cause several of its fanout branches to be faulty. So for 2 or 3 physical defects, the number of faulty locations can be more than 10.

In table 18, for each design, each fault type, each number of physical faults, there are 50 test cases. In the columns of the 2nd row, the heading "3" shows the number of cases for which 3 injected defects are reported while the headings "2", "1" or "0" show the number of cases that 2, 1 or none of the injected defects are reported. In the 2nd column under "type" it show the defect types: O represent net open defect; C represents cell internal defect; B represents dominant bridge defects. For example, we can see that in Table 18, for S38584 circuit, the new proposed method was able to identify all 3 physical

defect correctly for 45 cases, and identify 2 out of 3 correctly for 4 cases and identify 1 out of 3 correctly for 1 case.

Diagnosis resolution is defined the same as in Table 17. When a logic location is reported with two fault types, we consider them two suspects. For example the same location can be identified net-open and cell-internal defects at the same time under a test set. In such cases, we consider them two suspects when calculating diagnosis resolution. In Table 18 we list the average resolution for all types of defects for each design under columns "AR". As we can see in Table 18, the new method yielded better accuracy and resolution.

As we discussed in Section 5.2, the missed diagnosis could be due to the following. When multiple defects are present, defective global signals such as defects on scan enable or clock tree can explain the failures with small passing mismatch in few cases and injected defects are not reported in such case. Also both the diagnosis accuracy and resolution is bounded by the production test set used to do diagnosis. For example, under some test set, one fault could be dominated by other faults and cannot be reported. Or two faults could be undistinguishable under a test pattern set.

Table 17: Faulty Location Identification Using Two Methods

design	type	7 flts				11 flts				16 flts				19 flts			
		new		[Ye 10]		new		[Ye 10]		new		[Ye 10]		new		[Ye 10]	
		Diag	A.R.	Diag	A.R.	Diag	A.R.	Diag	A.R.	Diag	A.R.	Diag	A.R.	Diag	A.R.	Diag	A.R.
s13207 (2573)	S.	70%	32%	68%	28%	67%	24%	66%	22%	62%	20%	61%	18%	60%	17%	58%	15%
	T.	64%	28%	64%	26%	61%	23%	60%	22%	58%	18%	57%	17%	57%	16%	56%	14%
	B.	65%	28%	64%	27%	60%	22%	59%	21%	57%	17%	57%	16%	54%	15%	54%	14%
	O.	68%	35%	66%	29%	64%	26%	62%	23%	61%	21%	59%	20%	57%	17%	55%	15%
s15850 (3448)	S.	69%	32%	68%	29%	68%	24%	66%	22%	60%	21%	60%	19%	60%	16%	59%	15%
	T.	62%	30%	61%	28%	59%	21%	58%	20%	57%	18%	56%	18%	55%	16%	54%	15%
	B.	62%	27%	62%	25%	59%	20%	59%	19%	55%	17%	55%	16%	54%	15%	53%	14%
	O.	66%	38%	66%	34%	62%	26%	62%	24%	61%	22%	60%	20%	56%	16%	55%	15%
s35932 (12204)	S.	69%	27%	67%	24%	66%	23%	63%	21%	58%	20%	56%	18%	57%	15%	56%	14%
	T.	63%	25%	61%	24%	56%	21%	56%	20%	55%	18%	54%	17%	55%	15%	53%	14%
	B.	62%	25%	62%	24%	59%	20%	58%	18%	54%	16%	54%	16%	51%	14%	50%	13%
	O.	67%	31%	66%	30%	63%	24%	62%	22%	60%	21%	58%	20%	54%	15%	52%	14%
s38417 (8709)	S.	66%	28%	65%	25%	65%	22%	65%	20%	59%	20%	57%	18%	58%	15%	56%	14%
	T.	62%	25%	61%	23%	57%	21%	56%	19%	55%	17%	55%	17%	54%	14%	53%	13%
	B.	62%	24%	61%	23%	60%	20%	59%	19%	54%	17%	53%	16%	51%	14%	49%	13%
	O.	68%	31%	66%	29%	64%	24%	62%	22%	60%	21%	59%	20%	55%	15%	54%	14%
s38584 (11448)	S.	72%	29%	71%	26%	68%	21%	67%	19%	66%	19%	66%	17%	62%	15%	60%	14%
	T.	67%	26%	67%	25%	58%	19%	58%	18%	56%	16%	55%	16%	55%	14%	54%	13%
	B.	65%	24%	63%	23%	61%	19%	60%	18%	56%	16%	56%	16%	54%	14%	53%	13%
	O.	70%	30%	71%	27%	60%	23%	59%	21%	67%	20%	67%	20%	58%	16%	56%	15%
b17 (22645)	S.	61%	25%	59%	22%	59%	21%	57%	19%	55%	18%	54%	17%	52%	14%	50%	13%
	T.	58%	24%	57%	22%	56%	19%	55%	18%	52%	16%	50%	15%	50%	13%	49%	13%
	B.	57%	22%	57%	21%	54%	19%	53%	18%	52%	16%	51%	15%	50%	13%	48%	13%
	O.	62%	28%	61%	25%	60%	22%	60%	21%	57%	20%	55%	19%	53%	15%	51%	14%
b22 (14282)	S.	62%	24%	60%	23%	57%	21%	56%	19%	54%	18%	52%	17%	52%	14%	50%	13%
	T.	57%	22%	57%	21%	55%	20%	55%	19%	51%	16%	50%	16%	50%	13%	49%	13%
	B.	57%	21%	56%	21%	55%	19%	53%	18%	52%	16%	51%	16%	49%	13%	48%	13%
	O.	61%	27%	60%	25%	59%	23%	58%	21%	56%	20%	56%	18%	53%	15%	52%	14%

Table 18: Diagnosis Results Using Three Methods

Circuit	Type	2 physical defects											
		New				[Ye 10]+FTI				[Tang 10]			
		2	1	0	A.R.	2	1	0	A.R.	2	1	0	A.R.
s15850	O	48	2	0	33.5%	48	2	0	33.2%	45	4	1	32.2%
	C	46	4	0	23.0%	45	4	1	22.6%	43	6	1	21.7%
	B	38	11	1	22.3%	35	14	1	21.8%	34	14	2	20.5%
s35932	O	47	3	0	32.3%	46	4	0	31.7%	46	4	0	30.6%
	C	47	2	1	25.2%	45	5	0	24.9%	45	4	1	23.8%
	B	41	8	1	22.8%	41	7	2	22.5%	39	9	2	21.2%
s38417	O	48	1	1	35.1%	47	2	1	34.7%	46	3	1	33.7%
	C	47	3	0	23.5%	45	6	0	23.2%	44	5	1	22.6%
	B	40	9	1	21.7%	38	11	1	21.4%	37	10	3	21.3%
s38584	O	47	1	1	32.0%	47	2	1	31.6%	47	2	1	30.7%
	C	46	4	0	22.5%	46	4	0	22.3%	45	5	0	21.9%
	B	39	10	1	21.0%	37	13	0	20.4%	37	11	2	19.8%
b17	O	47	3	0	34.4%	47	3	0	34.2%	45	5	0	32.3%
	C	46	4	0	22.1%	45	5	0	21.8%	45	4	1	21.7%
	B	42	7	1	22.0%	40	9	1	21.4%	39	9	2	21.0%

Table 19: Diagnosis Results Using Three Methods

Circuit	Type	3 physical defects														
		New					[Ye 10]+FTI					[Tang 10]				
		3	2	1	0	A.R.	3	2	1	0	A.R.	3	2	1	0	A.R.
s15850	O	45	4	1	0	28.5%	44	4	2	0	28.2%	41	6	3	0	27.5%
	C	44	3	2	1	20.9%	44	2	3	1	20.7%	42	3	4	1	19.3%
	B	35	11	4	0	19.1%	33	12	5	0	18.6%	30	12	7	1	17.5%
s35932	O	46	3	1	0	30.2%	46	2	2	0	30.0%	44	3	2	1	29.4%
	C	44	5	0	1	19.2%	43	5	1	1	18.8%	41	4	3	2	18.3%
	B	36	8	5	1	18.0%	35	7	7	1	17.6%	32	9	7	2	16.6%
s38417	O	46	4	0	0	32.4%	45	3	2	0	31.3%	45	2	3	0	30.7%
	C	45	4	1	0	20.3%	43	5	1	1	19.9%	42	5	2	1	19.5%
	B	39	9	2	0	17.9%	37	11	2	0	17.7%	34	8	7	1	17.2%
s38584	O	45	4	1	0	31.8%	43	6	1	0	31.4%	43	3	4	0	30.5%
	C	46	3	1	0	21.2%	45	3	2	0	21.0%	43	2	4	1	20.4%
	B	40	5	4	1	18.0%	38	6	5	1	17.9%	35	8	4	3	17.7%
b17	O	46	3	1	0	31.4%	44	4	2	0	30.9%	42	5	2	1	29.8%
	C	45	3	2	0	19.3%	45	2	3	0	19.1%	40	6	4	0	18.0%
	B	38	10	1	1	18.4%	35	12	2	1	18.0%	32	11	5	2	17.4%

CHAPTER 6. CONCLUSIONS

6.1 Conclusion of Completed Research

Diagnosis becomes very crucial in nowadays yield learning process. Effectively and accurately locating the source of physical defects in failed chips and identifying the cause of such failures provide guide for expensive physical failure analysis. It is an important step toward effective silicon debug and yield improvement. This thesis describes several methods to improve the accuracy, resolution and also speed of scan chain and logic diagnosis.

In Chapter 2, we reviewed previous research work on scan chain and logic diagnosis. We gave a review of fault models: stuck-at fault model, bridge fault model, open fault model etc. Methods in different categories to diagnose scan chain failures are reviewed with a focus on software-based method. Then multiple fault diagnosis methods are also reviewed, including diagnosis method based on SLAT patterns, multiple fault diagnosis based on Xlists; curable vectors and curable outputs; incremental multiple fault diagnosis and diagnostic pattern generation techniques for multiple faults.

In Chapter 3, we proposed a method to improve diagnosability of production test patterns for multiple scan chain failures under the test compaction methodology. As test compaction becomes more and more popular in industry, diagnosis of scan chain failures become more challenging as the loss of direct observation. We proposed a simple and effective method to improve the diagnostic resolution of multiple chain failures with minimal increase in pattern counts. This method can be easily adopted into any test flow environment as it does not require any changes to test flows.

In Chapter 4, we proposed new diagnostic test generation procedures to enhance an earlier diagnostic test generation technique for scan chain failures. A diagnostic test generation flow to reduce run times for diagnostic test generation was also proposed. The proposed test generation procedures use multi-cycle scan test patterns with additional constraints to improve the success rate of generating a pattern to differentiate a scan cell from all cells downstream to it. To save test generation effort for sequential test pattern generation with multiple capture cycles, we applied SCOAP concept to calculate the minimum sequential depth of test patterns for each target scan cell. The proposed speed-up flow tries to simultaneously target differentiating as many scan cells as possible by a single test. Experimental results on several industrial designs have demonstrated the effectiveness of the proposed procedures in improving the diagnostic resolution, reducing the runtime and the number of test patterns.

In Chapter 5, we first propose a method based on SLAT patterns and bit union technique to improve diagnosis results when multiple physical defects are presents in circuits under diagnosis. The method first identifies initial faulty locations and then identifies physical fault types. To improve diagnosis results when multiple defects are present in a circuit under diagnosis, the proposed method includes (i) analyzing relations among locations of logic faults and their diagnostic metrics to carefully derive physical faults, (ii) a new set covering procedure and (iii) a method to assign scores to faults to derive candidate sets of faults. Experimental results on several industrial designs and several cases of silicon defects show the effectiveness of the proposed diagnosis method. Then we propose a new method based on fault-tuple equivalence tree to identify initial faulty locations. Then fault types are identified the same as in the first method.

Finally in Chapter 6, we concluded this thesis and propose future research directions.

6.2 Future Research

All of the techniques described in this thesis shares the objective of producing merrier diagnosis results for different types of defects in VLSI circuit. Here we propose several future research directions as continuation.

One future research topic is advanced diagnosis method for scan cell internal defects. As typically 30%-50% of the logic gates in a scan-based design impact the operation of scan chains, scan chain failures are the cause of substantial portion of failing chips. With the advancement of manufacturing processes from 90nm to 65nm and beyond, a significant number of manufacturing defects lie inside design library cells [Cheng 07]. The faulty behavior of these defects that lie inside design library cells may not be modeled using classic fault models. Recently some research has been published to address the testing issue of such scan cell internal faults. [Guo 08] also studied the diagnosis of static scan cell internal defects. The diagnosis of scan cell internal defects is still very challenging mainly due to the complexity of the faulty behavior.

In Chapter 4, we investigated a method to generate a complete test pattern set for diagnosis of scan chain failures. When production test pattern set doesn't have good enough diagnosability, diagnostic patterns can be generated to further improve diagnosis results for PFA guidance. The complete test pattern set we proposed is under the assumption that a single faulty scan cell is present in the scan chain. It is important to do the research on how to generate diagnostic patterns with good diagnosability when there are multiple faulty scan cells in the faulty scan chain.

Another research direction is on statistical learning of a large number of diagnosis data to identify the root cause of system defects. Volume diagnosis becomes very important to find the systematic defects and ramp up yield. In volume diagnosis, diagnosis is performed on a large number of failing chips to find yield-limiting systematic defects and design issues. Research on statistical methods of processing a large number of diagnosis results is necessary, in order to accurately identify the root cause of system defects.

REFERENCES

[Kundu 94] S. Kundu, “Diagnosing Scan Chain Faults," IEEE Trans. On VLSI Systems, Vol. 2, No. 4, December 1994, pp. 512-516.

[Chuang 08] W.-S. Chuang, W.-C. Liu and J. C-M. Li, “Diagnosis of Multiple Scan Chain Timing Faults”, IEEE trans. on CAD of Integrated Circuits and Systems, Vol. 27, No. 6, June 2008, pp1104-1116.

[De 95] K. De and A. Gunda, “Failure Analysis for Full-Scan Circuits,”, Proc. ITC 95, IEEE Press, 1995, pp. 636-645.

[Song 04] P. Song et al., “A Novel Scan Chain Diagnostics Technique Based on Light Emission from Leakage Current,” Proc. ITC 04, IEEE CS Press, 2004, pp. 140-147.

[Motika 03] F. Motika et al., AC Scan Diagnostic Method, US PATENT 6516432, Patent and Trademark Office, 2003.

[Motika 06] F. Motika, P.J. Nigh, and P. Song, Stuck-At Fault Scan Chain Diagnostic Method, US Patent 7010735, Patent and Trademark Office, 2006.

[Schafer 92] J.L. Schafer, F.A. Policastri, and R.J. Mcnulty, “Partner SRLs for Improved Shift Register Diagnostics,” Proc. 10th IEEE VLSI Test Symp. (VTS 92), IEEE Press, 1992, pp. 198-201.

[Edirisooriya 95] S. Edirisooriya and G. Edirisooriya, “Diagnosis of Scan Path Failures,” Proc. 13th IEEE VLSI Test Symp. (VTS 95), IEEE Press, 1995, pp.250-255.

[Narayanan 97] S. Narayanan and A. Das, “An Efficient Scheme to Diagnose Scan Chains,” Proc. Int’l Test Conf. (ITC 97), IEEE CS Press, 1997, pp. 704-713.

[Narayanan 99] S. Narayanan and A. Das, Flip-Flop Design and Technique for Scan Chain Diagnosis, US patent 5881067, Patent and Trademark Office, 1999.

[Wu 98] Y. Wu, “Diagnosis of Scan Chain Failures,” Proc. Int’l Symp. Defect and Fault Tolerance in VLSI Systems (DFT 98), IEEE Press, 1998, pp.217-222.

[Song 00] P. Song, “A New Scan Structure for Improvinig Scan Chain Diagnosis and Delay Fault Coverage,” Proc. 9th IEEE North Atlantic Test Workshop (NATW 00), 2000, pp. 14-18.

[Tekumulla 07] R.C. Tekumulla and D. Lee, “On Identifying and Bypassing Faulty Scan Segments,” Proc. 16th IEEE North Atlantic Test Workshop (NATW 07), 2007 pp. 134-143.

[Stanley 01] K. Stanley, "High Accuracy Flush-and-Scan Software Diagnostic," IEEE Design & Test, vol. 18, no.6, Nov.-Dec. 2001, pp.56-62.

[Guo 01] R. Guo, S. Venkataraman, "A Technique For Fault Diagnosis of Defects in Scan Chains", ITC, 2001, pp. 268-277.

[Kao 06] Y.-L. Kao, W.-S. Chuang, and J.C.-M. Li, "Jump Simulation: A Technique for Fast and Precise Scan Chain Fault Diagnosis," Proc. Int'l Test Conf. (ITC 06), IEEE CS Press, 2006, paper 297659.

[Huang 07] Y. Huang, "Dynamic Learning Based Scan Chain Diagnosis," Proc. Design, Automation and Test in Europe Conf. (DATE 07), IEEE CS Press, 2007, pp. 510-515.

[Huang 03] Y. Huang et al., "Efficient Diagnosis for Multiple Intermittent Scan Chain Hold-Time Faults," Proc. 12th Asian Test Symp. (ATS 03), 2003, pp.44-49.

[Huang 05_1] Y. Huang, W.-T. Cheng, and G. Crowell, "Using Fault Model Relaxation to Diagnose Real Scan Chain Defects," Proc. Asia and South Pacific Design Automation Conf., IEEE Press, 2005, pp. 1176-1179.

[Huang 05_2] Y. Huang, W.-T. Cheng, and J. Rajski, "Compressed Pattern Diagnosis for Scan Chain Failures," Proc. Int'l Test Conf., IEEE CS Press 2005, pp. 751-759.

[Huang 06] Y. Huang et al., "Diagnosis with Limited Failure Information," Proc. Int'l Test Conf., IEEE CS Press, 2006, paper 297660.

[Guo 07] R. Guo, Y. Huang, and W.-T. Cheng, "Fault Dictionary Based Scan Chain Failure Diagnosis," Proc. 16th Asian Test Symp., IEEE CS Press, 2007, pp. 45-50.

[Kundu 93] S. Kundu, "On Diagnosis of Faults in a Scan-Chain," Proc. 11th Ann. Ieee vlsi Test Symp., IEEE Press, 1993, pp. 303-308.

[Kundu 94] S. Kundu, "Diagnosing Scan Chain Faults," IEEE Trans. Very Large Scale Integration Systems, vol. 2, no. 4, Dec. 1994, 512-516.

[Yang 05] J.-S. Yang and S.-Y. Huang, "Quick Scan Chain Diagnosis Using Signal Profiling," Proc. Int'l Computer Design, IEEE CS Press, 2005, pp. 157-160.

[Hsu 06] E. Hsu, S.-Y. Huang, and C.-W. Tzeng, "A New Robust Paradigm for Diagnosing Hold-Time Faults in Scan Chains," Proc. IEEE Int'l Symp. VLSI Design, Automation and Test, IEEE Press, 2006, pp. 171-174.

[Tzeng 07_1] C.-W. Tzeng and S.-Y. Huang, “Diagnosis by Image Recovery: Finding Mixed Multiple Timing Faults in a Scan Chain,” IEEE Trans. Circuits and Systems II, vol. 54, no. 8, Aug. 2007, pp. 690-694.

[Tzeng 07_1] C.-W. Tzeng and S.-Y. Huang, “A Robust Paradigm for Diagnosing Hold-Time Faults in Scan Chains,” IET Proc. Computers and Digital Techniques, vol. 1, no 6, 2007, pp. 706-715.

[Li 05_1] J.C.-M. Li, “Diagnosis of Single Stuck-At Faults and Multiple Timing Faults in Scan Chains,” IEEE Trans. Very Large Scale Integration Systems, vol. 13, no. 6, June 2005, pp. 708-718.

[Li 05_2] J.C.-M. Li, “Diagnosis of Multiple Hold-Time and Setup-Time Faults in Scan Chains,” IEEE Trans. Computers, vol. 54, no. 11, Nov. 2005, pp. 1467-1472.

[Couch 05] A. Crouch, “Debugging and Diagnosing Scan Chains,” Electronic Device Failure Analysis, vol. 7, no. 1, Feb. 2005, pp. 16-24.

[Sinanoglu 07] O. Sinanoglu and P. Schremmer, “Diagnosis, Modeling and Tolerance of Scan Chain Hold-Time Violations,” Proc. Design, Automation and Test in Europe Conf., IEEE CS Press, 2007, pp. 516-521.

[Guo 07] R. Guo, Y. Huang, W.-T. Cheng, “A Complete Test Set to Diagnose Scan Chain Failures”, ITC 2007, paper 7.2.

[Abramovici 90] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital System Testing and Testable Design, Computer Science Press, New York, 1990.

[Waicukauski 89] J. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured Circuits", IEEE Design and Test of Computer, vol. 6, no. 4, 1989, pp.49-60.

[Veneris 99] A. Veneris, S. Venkataraman, I. N. Hajj, and W. K. Fuchs, “Multiple design error diagnosis and correction in digital VLSI circuits,” VTS 1999, pp. 58-63.

[Bartestein 01] T. Bartenstein, D. Heaberlin, L. Huisman, and D. Slinwinski, “Diagnosing combinational logic designs using the single location-at-a-time (SLAT) paradigm,” Proc. of ITC, 2001, pp. 287-296.

[Boppana 99] B. Boppana, R. Mukherjee, J. Jain, and M. Fujita, “Multiple error diagnosis based on Xlists”, in Proc. 36th DAC, 1999, pp. 660-665.

[Huang 01] S.-Y. Huang, “On improving the accuracy of multiple defect diagnosis” in Proc. 19th IEEE VLSI Test Symposium, 2001, pp.34-39.

[Kuehlmann 94] A. Kuehlmann, D. I. Cheng, A. Srinivasan, and D. P. Lapotin, “Error Diagnosis for Transistor-level Verification”, Proc. of DAC 94, pp.218-223.

[Venkatraman 97] S. Venkatraman and W. K. Fuchs, "A Deductive Technique for Diagnosis of Bridge Faults", 1997, ICCAD, pp. 562-567.

[Huang 99] S.-Y. Huang, and K.-T. Cheng, “ErrorTracer: A Fault-Simulation-Based Approach to Design Error Diagnosis”, IEEE Trans on CAD-ICS, pp. 1341-1352, Sept. 1999.

[Takahashi 02] H. Takahashi, K. O. Boateng, K. K. Saluja, and Y. Takamatsu, “On diagnosing multiple stuck-at faults using multiple and single fault simulation in combination circuits,” IEEE Trans. CADICS, vol.21, no.3, pp362-368, Mar. 2002.

[Wang 03_1] Z. Wang, K.-H. Tsai, M. Marek-Sadowska, and J. Rajski, “An efficient and effective methodology on the multiple fault diagnosis,” ITC 2003, pp. 329-338.

[Wang 03_2] Z. Wang, M Marek-Sadowska, K.-H. Tsai, J. Rajski, “Multiple Fault Diagnosis Using n-Detection Tests,” Proc. Of the 21st ICCCD 03, pp. 198-201.

[Wang 05] Z. Wang, M. Marek-Sadowska, K. H. Tsai, and J. Rajski, “Delay-Fault Diagnosis Using Timing Information”, IEEE transactions on computer-aided design of integrated circuits and systems, vol.24, no. 9, September 2005, pp. 1315 – 1325.

[Wang 06] Z. Wang, M. Marek-Sadowska, K. H. Tsai, and J. Rajski, “Analysis and Methodology for Multiple-Fault Diagnosis,” IEEE CADICS, vol. 25, no. 3, March 2006.

[Lin 07] Y.-C. Lin, F. Lu, and K.-T. Cheng, “Multiple-Fault Diagnosis Based On Adaptive Diagnostic Test Pattern Generation,” IEEE Transactions on CAD of integrated circuits and systems, vol. 26, no. 5, May 2007.

[Yu 08_1] X. Yu and R. D. Blanton, “Effective and Flexibble Multiple Defect Diagnosis Methodology Using Error Propagation Analysis,” ITC 2008, paper 17.1.

[Yu 08_2] X. Yu and R. D. Blanton, “Multiple Defect Diagnosis Using No Assumptions On Failing Pattern Characteristics,” Proc. DAC 2008, pp. 361-366.

[Schafer 92] J. Schafer, F. Policastri and R. Mcnulty, “Partner SRLs for Improved Shift Register Diagnostics", Proc. VSLI Test Symposium, 1992, pp. 198-201.

[Edirisooriya 95] S. Edirisooriya, G. Edirisooriya, “Diagnosis of Scan Path Failures," Proc. VLSI Test Symposium 1995, pp. 250-255.

[Narayanan 97] S. Narayanan, A. Das, “An Efficient Scheme to Diagnose Scan Chains," Proc. Int'l Test Conference, 1997, pp. 704-713.

[Wu 98] Y. Wu, "Diagnosis of Scan Chain Failures," Proc. Int'l Symp. on Defect and Fault Tolerance in VLSI Systems, 1998, pp. 217-222.

[Gunda 95] K. De, A. Gunda, “Failure Analysis for Full-Scan Circuits”, ITC, 1995, pp. 636-645.

[Song 99] P. Song, F. Motika, D. Knebel, R. Rizzolo, M. Kusko, J. Lee and M. McManus, “Diagnostic techniques for the IBM S/390 600MHz G5 Microprocessor”, Proc. International Test Conference, 1999, pp. 1073-1082.

[Hirase 99] J. Hirase, N. Shindou and K. Akahori, “Scan Chain Diagnosis using IDDQ Current Measurement", Proc. Asian Test Symposium, 1999, pp. 153-157.

[Hwang 08] J. Hwang, D. Kim, N. Seo, E. Lee, W. Choi, Y. Jeong, J. Orbon, S. Cannon, “Deterministic Localization and Analysis of Scan Hold-Time Faults”, International Symp. On Test and Failure Analysis, Nov. 2008, paper 13.3.

[Crouch 05] A. Crouch, "Debugging and Diagnosing Scan Chains," EDFAS, Vol. 7, Feb., 2005, pp 16-24.

[Yang 04] J. S. Yang, S. Huang, “Quick Scan Chain Diagnosis Using Signal Profiling”, ICCD, 2004.

[Wang 08] F. Wang, Y. Hu, H. Li, X. Li, Y. Jing, Y. Huang, “Diagnotic Pattern Generation for Compound Defects”, ITC 2008, paper 14.1.

[Stanley 00] K. Stanley, “High Accuracy Flush and Scan Software Diagnostic", Proc. IEEE YOT 2000, Oct. 2000.

[Huang 03] Y. Huang, W.-T. Cheng, S.M. Reddy, C.-J. Hsieh, Y.-T.Hung, “Statistical Diagnosis for Intermittent Scan Chain Hold-Time Fault”, ITC, 2003, pp.319-328.

[Huang 05] Y. Huang, W.-T. Cheng and G. Crowell “Using Fault Model Relaxation to Diagnose Real Scan Chain Defects”, ASP-DAC, 2005, pp. 1176-1179.

[Kong 05] C.L. Kong, M.R. Islam “Diagnosis of Multiple Scan Chain Faults”, ISTFA, Nov. 2005, pp. 510-516.

[Guo 06] R. Guo, S. Venkataraman, “An algorithmic technique for diagnosis of faulty scan chains”, IEEE Trans. on CAD, Sept. 2006, pp. 1861-1868.

[Huang 06] Y. Huang, W.-T. Cheng, N. Tamarapalli, J. Rajski, R. Klimgerberg, W. Hsu and Y.-S. Chen, “Diagnosis with Limited Failure Information”, ITC 2006, paper 22.2.

[Goldstein 79] L. H. Goldstein, “Controllability/observability analysis of digital circuits”, IEEE Transaction of Circuits and systems, 26:685-693, 1979.

[Chuang 08] W.-S. Chuang, W.-C. Liu and J. C-M. Li, “Diagnosis of Multiple Scan Chain Timing Faults”, IEEE trans. on CAD of Integrated Circuits and Systems, Vol. 27, No. 6, June 2008, pp1104-1116.

[Rajski 04] J. Rajski, J. Tyszer, M. Kassab and N. Mukherjee, “Embedded Deterministic Test”, IEEE trans. on CAD, VOL. 23, NO. 5, May 2004, pp 776-792.

[Holst 07] S. Holst and H.-J. Wunderlich, “Adaptive Debug and Diagnosis without Fault Dictionaries,” in Proceedings European Test Symposium, 2007, pp.7–12.

[Holst 09] S. Holst and H.-J. Wunderlich, “A Diagnosis Algorithm for Extreme Space Compaction,” DATE, 2009, pp.1355–1360.

[Aitken 97]. R. C. Aitken, “Modeling the unmodelable: Algorithmic fault diagnosis”, IEEE Design and Test of Computers, Jul.-Sep.1997, pp. 98-103.

[Boppana 99] B. Boppana, R. Mukherjee, J. Jain, and M. Fujita, “Multiple error diagnosis based on Xlists,” Proc. of DAC 1999, pp. 660-665.

[Huang 01] S.-Y. Huang, “On improving the accuracy of multiple defect diagnosis,” Proc. VTS 2001, pp. 34-39.

[Takahashi 02] H. Takahashi, K. O. Boateng, K. K. Saluja, and Y. Takamatsu, “On diagnosing multiple stuck-at faults using multiple and single fault simulation in combinational circuits,” IEEE TCAD 2002, pp. 362-368.

[Wang 06] Z. Wang, M. Marek-Sadowska, and J. Rajski, “Analysis and methodology for multiple-fault diagnosis,” IEEE TCAD Mar. 2006, pp. 558-576.

[Lin 07] Y.-C. Lin, F. Lu, and K.-T. Cheng, “Multiple-fault diagnosis based on adaptive diagnostic test pattern generation,” IEEE TCAD, May 2007, pp. 932-942.

[Yu 08] X. Yu and R.D. Blanton, “An effective and flexible multiple defect diagnosis methodology using error propagation analysis,” Proc. ITC 2008, paper 17.1.

[Huisman 04] L. M. Huisman, "Diagnosing arbitrary defects in logic designs using single location at a time (SLAT)", IEEE TCAD, Jan. 2004, pp. 91-101.

[Cheng 04] W.-T. Cheng, K.-H. Tsai, Y. Huang, N. Tamarapalli and J. Rajski, "Compactor independent direct diagnosis", Proc. ATS 2004, pp. 204-209.

[Zou 06] W. Zou, W.-T. Cheng, S.M. Reddy and H. Tang, "On methods to improve location based logic diagnosis", Proc. Inter. Conf. VLSI Design 2006, pp. 181-187.

[Venkataraman 00] S. Venkataraman and S. B. Drummonds, " Poirot: a logic fault diagnosis tool and its applications", Proc. ITC 2000, pp. 253-262.

[Sharma 07] M. Sharma, W.-T. Cheng, T.-P. Tai, Y.S. Cheng, W. Hsu, C. Liu, S. M. Reddy and A. Mann, "Fast defect localization in nanometer technology based on defective cell diagnosis", Proc. ITC, 2007, paper 15.3.

[Bartenstein 01] T. Bartenstein, D. Heaberlin, L. Huisman and D. Sliwinski, "Diagnosing Combinational Logic Designs Using the Single Location At-a-Time (SLAT) Paradigm", Proc. ITC, 2001, paper 10.4

[Venkataraman 97] S. Venkataraman and W. Fuchs, "A deductive technique for diagnosis of bridging faults," Proc. of ICCAD, pp. 562-567, 1997.

[Rajski 04] J. Rajski, J. Tyszer, M. Kassab and N. Mukherjee, "Embedded Deterministic Test", IEEE trans. on CAD, VOL. 23, NO. 5, May 2004, pp 776-792.

[Tsai 09] M.-J. Tsai, M. C.-T. Chao, J.-Y. Jou and M.-C. Wu, "Multiple-Fault Diagnosis Using Faulty-Region Identification", Proc. of IEEE VTS, 2009, pp. 123-128.

[Menon 91] P. R. Menon, Y. Levendel, and M. Abramovici, "SCRIPT: A Critical Path Tracing Algorithm for Synchronous Sequential Circuits," IEEE Tran. on Computer Aided Design, vol. 10, pp. 738–747, June 1991.

[Akers 90] S. B. Akers, B. Krishnamurthy, S. Park, and A. Swaminathan, "Why is Less Information From Logic Simulation More Useful in Fault Simulation?," in Proc. of the IEEE Intl. Test Conf., pp. 786–800, Sept. 1990.

[Ye 10] J. Ye, Y. Hu and X. Li, "Diagnosis of Multiple Arbitrary Faults with Mask and Reinforcement Effect," Proc. of IEEE DATE, 2010, pp. 885-890.

[Tang 09_1] X. Tang, R. Guo, W.-T. Cheng, and S. M. Reddy, “Improving Compressed Test Pattern Generation for Multiple Scan Chain Failure Diagnosis,” DATE 2009, pp. 1000-1005.

[Tang 09_2] X. Tang, R. Guo, W.-T. Cheng, S. M. Reddy, and Y. Huang, “Improving Diagnostic Test Generation for Scan Chain Failures Using Multi-Cycle Scan Patterns,” Workshop paper, ETS 2009.

[Tang 09_3] X. Tang, R. Guo, W.-T. Cheng, S. M. Reddy, and Y. Huang, “On Improving Diagnostic Test Generation for Scan Chain Failures,” ATS 2009. (Accepted by ATS).

[Tang 10_2] X. Tang, W.-T. Cheng, R. Guo and S.M. Reddy, "Diagnosis of Multiple Defects Using Logic Fault Models", Workshop paper, European Test Symposium, 2010.

[Tang 10] X. Tang, W.-T. Cheng, R. Guo and S.M. Reddy, "Diagnosis of Multiple Physical Defects Using Logic Fault Models", ATS2010.