Net Diagnosis Using Stuck-at and Transition Fault Models

by

Lixing Zhao

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
December 12, 2011

Keywords: Diagnosis, faulty nets, fault dictionary, multiple faults, stuck-at faults, transition faults

Approved by

Vishwani D.Agrawal, Chair, James J. Danaher Professor of Electrical Engineering
Adit Singh, James B.Davis Professor of Electrical and Computer Engineering
Victor P. Nelson, Professor of Electrical and Computer Engineering

Abstract

As deep sub-micron technologies are widely adopted in modern VLSI design and fabrication process, the shrinking size and increasing complexity of digital circuits make it more difficult to maintain a high yield. Diagnosis is the procedure used when circuit verification fails. Determining the cause of the failure and finding the possible defect locations are included in diagnosis. In this thesis, a procedure of diagnosing multiple net-faults is proposed.

Many previous studies on fault diagnosis mainly focused on single failures. However, considering the increasing complexity of current devices, multiple fault models may better reflect the real failures. Therefore, this thesis aims at using multiple fault models for diagnosis, though only the single fault simulation information is used. We utilize test patterns that have high diagnostic coverage for single stuck-at and single transition faults. Thus, we can expect a better distinguishablility between faults and hopefully better diagnosis can be achieved.

We propose a candidate filtering system based on probabilistic counting criteria, which is different from previous methods that simply relied on strict matching criteria and simple matching count. The proposed candidate filtering system utilizes information on passing patterns as well as that on both erroneous and error-free primary outputs of failing patterns to construct a balanced ratio-count for each fault that could potentially exist in the faulty circuit. By using such ratio-count for each fault, we hope to have better chances of discovering those faults that have been reported detected only few times. In reality this would be the case with hard to detect faults. Fault candidates are then filtered by dropping those with counts lower than an empirically determined threshold.

After candidate filtering, we use a candidate ranking system to rank the remaining fault candidates. We define a structure called erroneous primary output tree (EPO-Tree), which for a failing pattern represents both the observed failing information from the circuit under diagnosis (CUD) and the simulation information of fault candidates. Our ranking procedure is performed within each EPO-Tree. A novel feature of this ranking system is that we do not rank the fault candidates only based on the overall matching performance, but we also consider their potential within each branch of the EPO-Tree. Any fault that has a chance to be the top candidate in any branch of certain EPO-Tree will be selected as the final candidates. By using this method, we can have a better chance to discover the fault candidates that may have a worse overall ranking from first stage but are unique enough in certain branch of EPO-Tree. What's more our candidate ranking system does not require the single-location-at-a-time (SLAT) ability of test pattern to activate single fault among multiple faults as previous works do.

After the fault ranking, we expand the collapsed fault candidates by uncollapsing faults. Each fault candidate in the final ranked list represents a net that it belongs to. According to the ranks of faults on a net we rank the net candidates. All nets derived from the final fault list are put into a net pool. The final net candidate list includes two parts: the first part includes the nets that have more than two members in the net pool, and the second part includes those nets that have only one member in the net pool. The further ranking of the net depends on the top ranked fault's rank.

We use a commercial fault simulation tool for experiments on ISCAS85 benchmark circuits. When simulated faulty responses of benchmark circuit c7552 with multiple faults on a single net were diagnosed, in 90% cases a set of one to three nets could be found such that this set contained the actual faulty net. Similarly, when responses for two faulty nets were diagnosed, up to six nets were identified containing the two real ones in 90% cases. We thus assessed the diagnosis algorithms to have 90% diagnosability with a resolution of three.

Acknowledgments

I would like to express my deepest and most sincere appreciation to my supervisor, Dr. Vishwani D. Agrawal, the James J. Danaher Professor at Electrical Engineering Department. Without his kindness, patience and continuous guidance during my research and course of study, this thesis would not have been completed. I really appreciate all his encouragement and effort to help me get through those struggling days of my research.

I am deeply grateful to my committee members, Dr. Victor P. Nelson and Dr. Adit. D. Singh for their great teaching during my course of study and their patient guidance of my thesis writing.

I am very grateful to my colleague and friend, Yu Zhang, for generating the diagnostic test sets so that I can make use of them in my research. I also appreciate his help for teaching me how to run Fastscan.

I am indebted to my parents and other family members for their love and support all these years. I am very happy and proud to dedicate this thesis to my family.

Many thanks to all my dearest friends at Auburn. Without them around, my life in Auburn would not be so enjoyable.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to high logic density, incomplete characterization information and variations in modern VLSI devices, chips on the first round of tape-out often fail or suffer a relatively low yield that may be economically unacceptable. Failure analysis plays an important role in bringing the yield up. First, it collects the failing information from the failed devices. Then it uses diagnostic methods to identify the possible causes and locations of failures. Finally, it looks into the failing circuit and finds the real defects caused by manufacturing processes or flaws of design. Subsequently, these failures or flaws are corrected. However, with the increasing complexity of Integrated Circuit (IC) logic design and increasing difficulty of physical inspection in today's multi-layer deep sub-micron devices, failure analysis has become exceedingly expensive and time-consuming. Therefore, improving the efficiency and accuracy of failure analysis is necessary for the yield ramping-up. Logic-level fault simulation and analysis are critical parts of failure analysis.

In failure analysis, a logic-level process called fault diagnosis is used to find the possible faulty locations of real defects on a failing chip. A good diagnosis algorithm or method is very critical for reducing the time spent on physical inspection and lowering the total cost of product development. There are three basic requirements for a sound diagnosis framework: high diagnosability, high resolution and good time efficiency. Diagnosability represents the percentage of the true defects that can be covered in the final reported results. Resolution is the average number of candidates that is identified in place of a true defect. Time efficiency represents the time consumed in the process of diagnosis.

The objective of fault diagnosis research is to find methods to improve the criteria mentioned above. Previous works on fault diagnosis have considered several aspects. For

example, fault models have been evolved in order to perfectly represent the true defect behavior. Several attempts are focused on algorithms that can improve the resolution and accuracy of the diagnosis. Others have concentrated on reducing the time of the diagnosis. In the present research we introduce a novel probabilistic filtering method and a fault candidate ranking algorithm. The thesis aims to propose a diagnosis method that can achieve both high resolution and accuracy for single and multiple net faults.

Net fault diagnosis is an important area in fault diagnosis. A net means a metal wire in the circuit. Typically, net problems can be divided into two groups: problems on nets within the logic gates and problems on nets which connect different logic gates. Due to the large routing area of modern VLSI devices, compared to the nets within logic gates, the routing interconnection nets are more vulnerable to certain manufacturing defects. In this work, our objective is to locate one or more interconnection nets with possible defects.

1.1 Problem Statement

Given the failing responses, test patterns and a netlist of the circuit under diagnosis (CUD), we must provide an effective method to identify the possible faulty nets.

1.2 Thesis Contributions

There are several original contributions in this thesis. First, we have proposed a novel probabilistic candidate filtering system that can assign a more balanced count to each fault compared to previous works, especially for those faults with fewer detection instances compared to others. This filtering system uses a probabilistic count and a partial matching criteria instead of the strict matching criteria or simply accumulated counting criteria as used in in some previous diagnosis works.This candidate filtering system also utilizes the passing pattern information together with the information of passing outputs under failing patterns to reduce fault candidates in the first stage of diagnosis. Passing information has seldom been used in the first stage of multiple fault diagnosis methods.

A proposed data structure called erroneous primary output tree (EPO-tree) is used to represent both the failing information of CUD and the simulation information of fault candidates under certain failing pattern. Our ranking procedure is performed within each EPO-Tree. The novel part of the proposed ranking system is that we do not rank the fault candidates based on the fault performance on every EPO-Tree. The faults that have a chance to be top candidates in any EPO-Tree are selected as the final candidates. By using this method, we can have a better chance to discover the fault candidates that have fewer detection instances because either they are hard to detect faults or their effects at primary outputs are masked by other faults. After the fault ranking, we extend the diagnosis to net candidates. Because our method has good diagnosis resolution and diagnosability for multiple faults diagnosis, we have a good chance to diagnose a net fault, which often leads to a multiple fault scenario.

The diagnosis method that we propose in this thesis can be used not only for net fault diagnosis with stuck-at and transition faults but with other type of faults as well. Besides, this method can be used both in cause-effect and effect-cause diagnosis procedures. In this work, we utilize test patterns [52, 53] with relatively high diagnosibility for single stuck-at faults and for single transition faults. Because we consider that the traditional fault simulation methods can have a better accuracy of fault diagnosis compared to the critical-path-tracing method, we utilize the traditional simulation method to perform fault simulation.

1.3 Organization of Thesis

In Chapter 2, we will first talk about some basic concepts from the field of VLSI diagnosis. Previous work on fault diagnosis will be discussed in Chapter 3. A method of diagnosing multiple faults will be presented in Chapter 4 and Chapter 5. The net diagnosis work will be discussed in Chapter 6. The thesis will be concluded in Chapter 7.

Chapter 2

Review of Concepts in Fault Diagnosis

In this chapter, we briefly review some basic concepts from the field of VLSI diagnosis.

2.1 Circuit Types

Generally speaking, digital circuits are divided into two main categories: *combinational circuit* and *sequential circuit*. As showed in Figure 2.1, a combinational circuit is a type of circuit for which the outputs of the circuit are functions of present inputs only, while the values of sequential circuit outputs depend not only on the present input values but also on the stored previous values. Sequential circuit usually contain memory components (e.g., D flip-flops), which can store certain previous values of the circuit. A sequential circuit utilizing D flip-flops as memory components is shown in Figure 2.2.

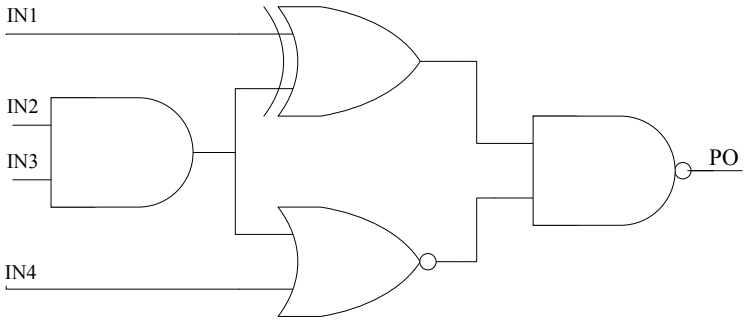

Figure 2.1: A combinational circuit example.

In this thesis, we aim at solving the diagnosis problem for combinational circuits only. Since scan-based design [6] is nowadays broadly used as a digital design-for-test (DFT) structure, the contents of the memory components can be shifted out by a scan-chain and

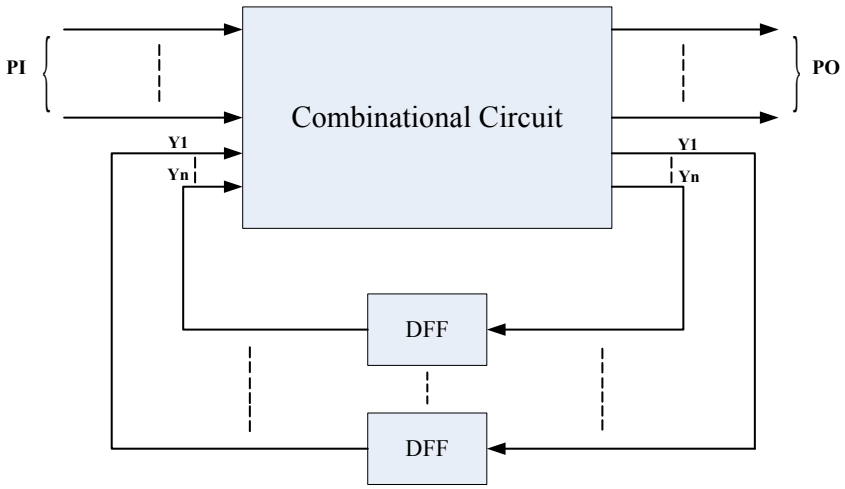

Figure 2.2: A sequential circuit example.

observed by the tester. Assuming that the DFT structure of a circuit is full-scan, which means all flip-flops in the circuit should be in the scan-chain and the contents of the flip-flops can be scanned out after each test vector application, the testing as well as diagnosis can be accomplished in a similar way as for combinational circuits. This is the reason why we choose combinational circuit as the circuit type for diagnosis in this thesis. Diagnosis for non-scan sequential circuits may be suggested for the future work.

2.2 Test Patterns and Circuit Testing

Input signals that are applied to the circuit for testing are called *test patterns* or *test vectors*. In scan testing, the test patterns are not only applied at the primary inputs they are also applied directly to the scan-registers as well. A *passing pattern* is the test pattern under which that the observed response of the device-under-test (DUT) matches the expected values, while a *failing pattern* is the test pattern under which the response of a circuit does not match the expected values. The test patterns used for diagnosis can be of different types: *random patterns*, *N-detect patterns* or *fault-model based diagnostic patterns*.

Random patterns are the test patterns that are generated randomly by a computer program or pattern generator (e.g., LFSR) [6, 41]. Because random patterns are generated

without specific requirements, good diagnosability may require increasing the number of test patterns. Usually, a random test set has a large data size for fault diagnosis. This causes problems with test set storage and testing time, because the memory storage space in tester is limited and time spent on testing each single circuit is critical for the whole testing cost.

N-detect patterns are model-based test patterns ensuring that each fault is detected by at least N different test vectors. N-detect patterns can improve the defect coverage [3, 43] by increasing, N, the number of times a single fault is detected [1, 3, 10, 16, 48]. However, N-detect patterns face the same problems of large test size. Many efforts have been devoted to minimizing the N-detect test size [19, 20, 33, 21].

Fault model based diagnostic patterns focus on some specific fault models, like bridging fault [31], gate-design-error fault [44], transition fault [53, 11] and single stuck-at fault [52]. In this thesis, we utilize previously generated test patterns [52, 53], which are based on single-stuck at and transition fault models, respectively, to solve the problem of net-fault diagnosis.

Testing is a procedure of applying a test pattern to the circuit and observing the output signals. A *test set* is a collection of tests that is used to test a certain circuit. The test set is applied by a *tester* whose basic purpose is to drive the inputs and to monitor the outputs of a device-under-test. Testers are popularly known as ATE (automatic test equipment) [6]. A *test program* is a program that runs on ATE and contains a sequence of instructions that the tester must follow to conduct the testing. This sequence of instructions may include controlling the power, applying clocks and patterns to the circuit, strobing output pins and comparing output signals with the expected responses stored in the tester.

2.3 Fault Models

As we have discussed in Chapter 1, due to the complexity of VLSI device fabrication and high logic density, there will be many types of design-errors or defects during the process of chip design and manufacturing. A *fault model* is an abstraction of a real defect in a chip

and we use different fault models to handle specific types of defects in testing and fault diagnosis. In VLSI circuits, functional correctness is guaranteed by good timing and perfect logic validity, so in order to model the behavior of real defects as well as it can a fault model should focus on the logic and timing behaviors. Physical defects can be modeled as logical faults or delay faults depending on whether the defect affects timing or logical function of the circuit. Several common logic fault models are reviewed in the following sections.

2.3.1 Stuck-at Fault Model

The term *stuck-at fault* was first introduced in a paper by Galaey et al. [9] in 1961. Stuck-at fault is a gate-level model, which assumes that certain interconnection of the circuit has a fixed value of '1' or '0' (referred to as "stuck-at-1" or "stuck-at-0"). In a circuit of n interconnections, there would be totally 2n stuck-at faults, but only one fault is assumed to appear on a interconnection at one time. When some physical defects occurs during the manufacturing precess, it can be modeled by stuck-at fault, like shorting of a signal line to VDD (logic '1') or GND (logic '0'). As showed in Figure 2.3, the OR gate has a stuck-at '0' fault on input line1. The expected output value should be '1', however, because of the presence of the stuck-at fault on input line, the output has a faulty value '0'. In order to detect the stuck-at fault value in the circuit, the test set should force an opposite value on the concerned line and drive the effect to primary outputs. As showed in Figure 2.3, if we want to test whether line1 is stuck-at 0, we have to apply a '1' on line1 and at the same time apply a non-controlling signal '0' on line 2 so that the value on line1 can be propagated to the output without blocking.

2.3.2 Bridging Fault Model

As showed in Figure 2.4, there is low resistance short between two lines that are close to each other. Because of such a short defect, there is an interfering effect between the two lines and logical errors may appear on either or both of the lines. Bridging fault model

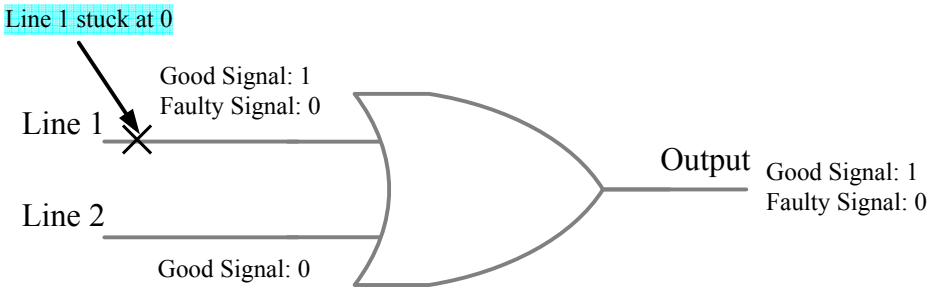

Figure 2.3: A stuck-at fault illustration.

represents such logical interference effects between shorted lines. The most common models for bridging are *wired-AND* and *wired-OR*. In the wired-AND model, when either of the expected value on two bridged lines is '0', then the value on both lines will be represented as '0'. Likewise, in the wired-OR model, if either value on the two bridged lines is '1', then the represented value on the two shorted lines will be '1'. Another bridging fault model is *dominant bridging fault*. In this model, the value on one of the bridged lines is much stronger than that on the other line and always forces the signal value on the other line as its own stronger signal value.

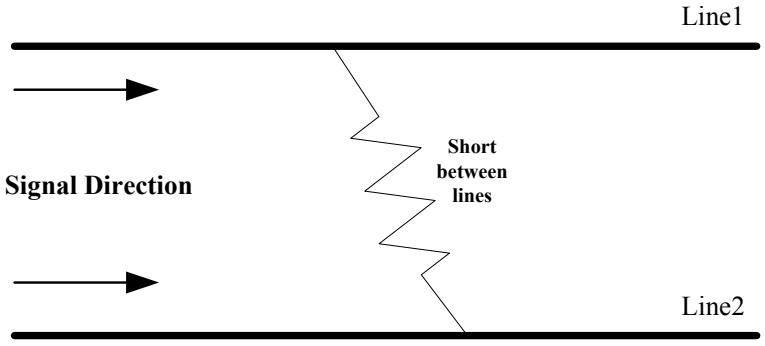

Figure 2.4: A bridge defect.

2.3.3 Transition Fault Model

Since modern VLSI devices usually work at very high clock frequencies, the requirement of right timing becomes critical to VLSI design and fabrication. Due to the defects in the fabrication process, some components like interconnection lines may add undesirable resistance to propagation paths of signals and result in delays in signal propagation. Transition delay fault model is one of the several delay fault models and it assumes that there is only one gate in the circuit that has excess delay. It is often referred to as just *transition fault*. The difference between the transition delay fault model and the stuck-at fault model is that the faulty behavior of a transition delay fault is 'slow to rise' or 'slow to fall' instead of permanently being 'stuck-at-0' or 'stuck-at-1'. In order to trigger the transition delay fault, a pair of test patterns has to be applied to the circuit inputs. The first pattern has to set the initial value of the interconnecting line that needs to be examined. Then the second pattern generates a transition on the line and also at the same time drives the effect to primary outputs. The observed faulty response is similar to the stuck-at fault response. A transition fault example is presented in Figure 2.5. There is a slow-to-rise fault on input line1 and the correct value of line1 comes too late so that the faulty value of '0' instead of the expected value '1' is observed on the output. In order to test whether there is a slow-to-rise fault on Line1, first we have to use an input patten to set the value on Line 1 to be '0', and then we apply an test pattern similar to the patterns that we have discussed for stuck-at fault testing to propagate the 'rising' signal to the output.

2.3.4 Open Fault Model

Open defects are the undesired breaks and electrical discontinuities in interconnects and also one of the most important production defects [13, 22, 23, 37]. Depending on whether the open crack is narrow or wide, open defects can be divided into two categories: weak open or resistive open as shown in Figure 2.6 and hard open or complete open as shown in Figure 2.7. A weak open only causes a small increase in the interconnection resistance and

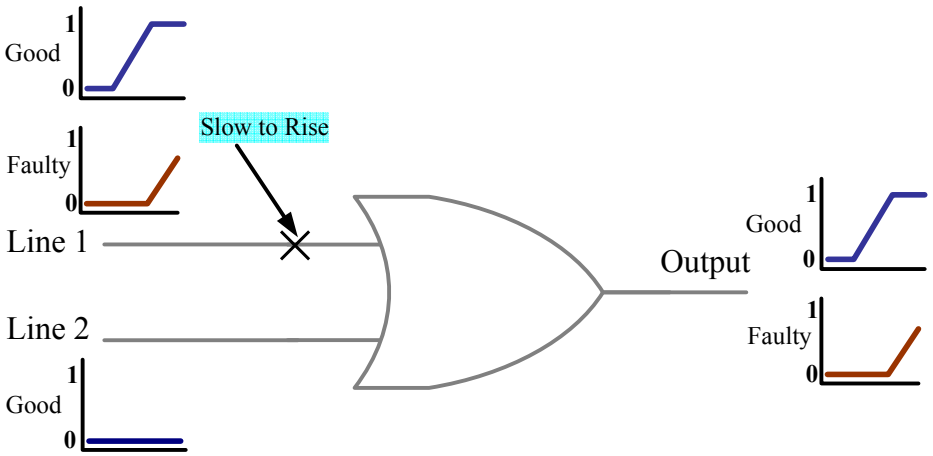

Figure 2.5: A transition delay fault example.

does not completely block the current through the line. Strong open causes a large increase in line resistance (usually larger than 1GΩ). Weak open usually causes delay problems in the circuit. The modeling of a strong open is complicated. There are three factors that may influence the voltage level of the floating node [37]:

- Coupling capacitors between the floating node and the supply and ground.
- Drain voltage of the driven gates.
- Effects from surrounding lines.

The effect of a strong open defect can be modeled in two ways [34]:

1. We assume that the gates driven by floating lines may interpret the logic state of the line differently, because of the variations of threshold voltages among the driven gates. This is referred to as Byzantine effect [25, 28]. Thus, the open lines behave as a combination of stuck-at-0 and stuck-at-1 faults at the inputs of the driven gates.

2. We assume that the open causes some lines to float. The open lines may be subject to crosstalk due to capacitive coupling to other nodes in the circuit. In other words, the states of the affected lines are pattern sensitive.

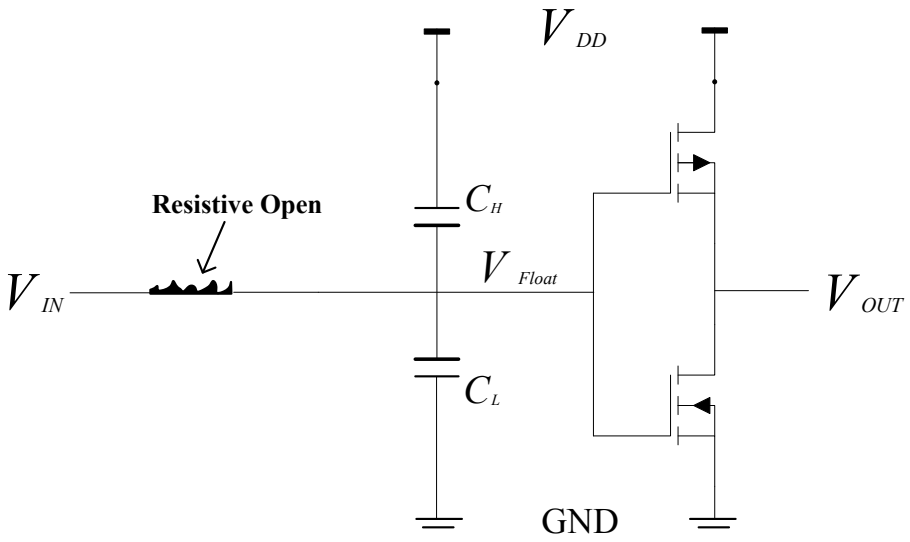

Figure 2.6: A resistive open defect example.

From the results of experimental comparison between open fault and stuck-at fault conducted by Intel [47], we found that in most cases the behavior of strong open defects can be explained by certain stuck-at faults. So, in this thesis we assume that for the second case listed above we only consider the supply and ground coupling capacitances (C_H and C_L in Figure 2.6 and Figure 2.7) and drain voltage of driven gates determine the dominant condition, then we can assume that the voltage value on the floating node is fixed and would not change dynamically with input patterns.

2.4 Fault Simulation

Fault simulation is a method that consists of simulating a circuit that has been injected with single faults, comparing the response of the faulty circuit and the good circuit under the same pattern, and determining which of the faults have been detected. A fault is found to be detectable when the responses from the good circuit and the faulty circuit are different under the same pattern. Three most important applications of fault simulation are listed below [6]:

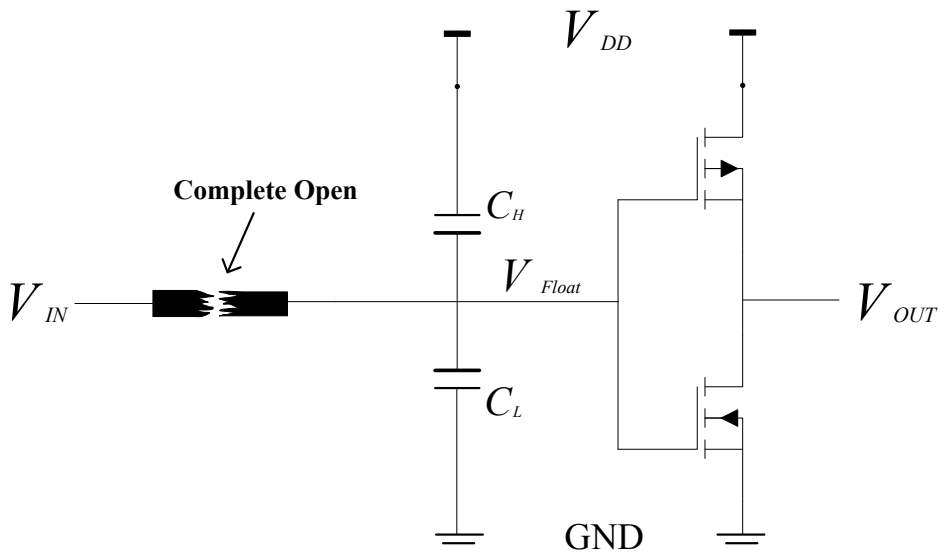

Figure 2.7: A complete open defect example.

- To evaluate the quality of a test set.
- To incorporate undetected faults into an ATPG (automatic test pattern generator) for test generation.
- To generate fault detection data for reducing fault candidates in diagnosis.

The testability of a test pattern set is usually measured by *fault coverage*. Fault coverage is the ratio of the number of faults detected by a test set and the total number of potential faults in the circuit. Because fault simulation is used to find out the faults that can be detected by certain test pattern, a list of detectable faults can be constructed after simulation and then the fault coverage of the test set can be computed. The representation of test coverage is shown below in Equation 2.1. In order to generate as few test patterns as possible to achieve highest fault coverage in test pattern generation, fault simulation can be used to determine whether the fault coverage of the generated test set is high enough. If the coverage is high, say close to 100%, then the test generation process can be stopped. Fault simulation is also very crucial in fault diagnosis, because using a fault simulator we

can compare the output responses of the an actual faulty circuit under test (CUT) and those from fault simulation. If the response from fault simulation after injecting certain fault can match the response from the real faulty CUT, then the simulated fault will be the diagnosed fault candidate.

$$\text{Fault Coverage} = \frac{\text{Number of faults detected}}{\text{Total number of faults}} \tag{2.1}$$

2.5 Fault Dictionary

Using fault simulation, we can inject all single faults into the circuit and produce the faulty response corresponding to each fault. The observed responses can be divided into two parts: information on passing or failing pattern index and information on passing and failing outputs under each failing pattern. Based on this, two kinds of fault dictionaries can be constructed: *pass-fail dictionary* and *full-response dictionary* [26]. The pass-fail dictionary only keeps the information of the failing pattern index for each fault and the full-response dictionary keeps not only the failing pattern index but also the failing outputs information for each fault. Table 2.1 shows a full-response dictionary of a single fault and Table 2.2 shows the corresponding pass-fail dictionary. The 1s in Table 2.1 represent mismatches observed in the corresponding primary outputs and the 1s in Table 2.2 show that the faulty responses are observed under the corresponding test patterns. In this thesis, we utilize a partial full-response dictionary to store the simulated failing information that includes the failing pattern index and failing output index for collapsed faults in in the circuit under diagnosis (CUD). Fault dictionary usually are used for single fault diagnosis. For multiple fault diagnosis, because of the exponential number of fault combinations, dictionary is not used. In this work, because we only use single fault simulation information, in order to save the time on fault simulation, we prefer to utilize a partial full-response dictionary.

Table 2.1: A full-response dictionary.

Pattern Index	Primary Outputs				
	PO1	PO2	PO3	PO4	PO5
P1	0	1	0	0	0
P2	0	1	1	0	0
P3	0	0	0	0	0
P4	0	0	1	0	0
P5	0	0	0	0	0
P6	0	0	0	0	0

Table 2.2: A pass-fail dictionary.

Pattern Index					
P1	P2	P3	P4	P5	P6
1	1	0	1	0	0

Chapter 3

Background Review on Previous Work on Diagnosis

In this chapter, we will review previous research work mainly focusing on stuck-at fault and open net fault diagnosis.

3.1 Stuck-at Fault Diagnosis

The Single stuck-at fault model has been used in many research works to solve the problem of fault diagnosis [35, 36, 49]. Traditional single stuck-at fault diagnosis usually involves the use of a full-response fault dictionary. Some works like [35] focus on building a diagnostic dictionary and compare the failing responses of a defective circuit and the simulated responses for faults to identify the true candidate. The identification of a candidate fault is based upon the number of times a mismatch occurs between fault simulation data and the observed circuit under diagnosis (CUD) response. The simulated fault that has the lowest mismatch count would be the candidate fault. The detailed counting method is as follows: if a value of '0' or '1' is observed on a primary output which is different from the simulation value of good circuit, then the count of this fault will be increased by 2. If the observed value is unknown, then the mismatch count will be increased by 1. In other cases, the count will not change.

In [36], instead of using a full-response dictionary covering the faulty behaviors of all faults in the circuit, the author first utilizes a test set that has a 100 percent fault coverage to test the circuit. After that, failing patterns from the first stage are used to do the fault simulation and find the common faults that these test patterns can detect. Then a larger test set is applied to construct a dynamic dictionary with relatively smaller size than the one mentioned in [35] and use the similar method as in [35] to do the fault diagnosis. This

method has the advantage of using less space for storing the simulation information, but the total simulation times would be increased with the number of faulty circuit analyzed. Single stuck-at fault model can cover many types of defects in circuit testing. However, in fault diagnosis, single stuck-at fault model probably will not work where multiple defects exist. So we need to extend the diagnosis model to multiple stuck-at faults. Because it is not realistic to construct a dictionary for multiple fault diagnosis, many other methods are proposed using ranking systems, Boolean equations, candidate elimination based on single fault, multiple fault or symbolic simulation, or region model.

In [7], the authors proposed a balanced ranking system targeting single and multiple fault diagnosis. This work considers both the failing pattern and passing pattern simulation information. The authors assign different weights on primary output mismatches on both failing and passing patterns to each fault, and then calculate the possibility that the fault may exist on the basis of the final weight. By modifying traditional ranking system with these weights, the algorithm assigned a quantitative ranking to each fault. The authors claim that the method could be used to explicitly target defects that behave similar to but not exactly like stuck-at faults. The shortcoming of this method is that although the author tried to balance all factors of each fault, the purpose of the work is still to use the simulation response of single fault to match all the erroneous responses of CUD under certain test pattern. When faults are activated at the same time, the accuracy of the diagnosis will not be guaranteed. So, this work may have a good diagnosis accuracy for single stuck-at fault diagnosis. The results for multiple stuck-at fault diagnosis are, however, pessimistic. We can only say that this work is an attempt, with many limitations, to extend the single stuck-at fault diagnosis to multiple-stuck at fault or arbitrary fault diagnosis.

In [12], Huang proposed a concept called partially curable vector and a grading system using this concept to handle multiple fault diagnosis problems. An input vector is partially curable by a fault injection if primary outputs reachable by that fault can be corrected by re-synthesis [5]. Then this vector will be a partially curable vector for the fault. The

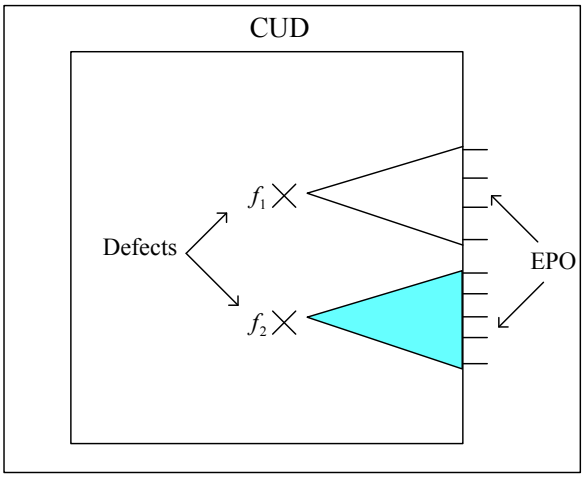

Figure 3.1: Multiple defects without mutual interference. EPO: erroneous primary output.

grading system in this work is based on this concept. The possibility of existence for each fault depends on how many partially curable vectors the fault has. So more failing vectors certain fault can fully explain, the higher possibility it has of being the true candidate. The algorithm also records the total number of erroneous primary outputs that each fault can correct. When there is more than one top candidate with the same number of curable vectors, then the total number of corrected primary outputs will be used as the tie breaker. The reason why the author abandoned the traditional curable output measure and used such a grading system is because some faults may have larger number of curable outputs than others and it depends on the circuit structure. Using the curable output measure the diagnosis can be misleading in some cases. Besides, it is quite possible that there are more than two defects in the faulty circuit and certain failing input vectors only activate just one defect. Thus, the faults will not affect each other. This kind of vectors should have a higher weight when grading the faults and faults detected by these vectors should be in a top candidate list.

Although this work may find some faults which have the chance to be activated alone among multiple faults, there is a shortcoming. The diagnosis depends heavily on the test pattern's ability to activate a single fault among multiple faults and assumes that there is

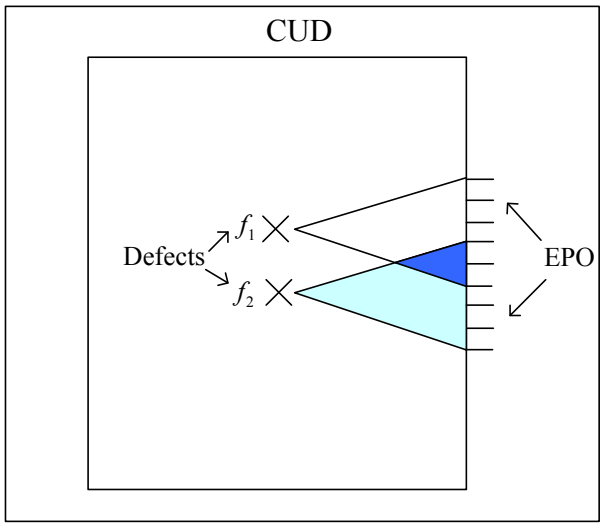

Figure 3.2: Multiple defects with mutual interference. EPO: erroneous primary output.

no interfering effect between multiple faults as shown in Figure 3.1. The condition is too restrictive and much information of fault simulation is ignored as a result. As showed in Figure 3.2, when multiple faults affect each other, this method will not work. Experiment results have shown that this method cannot handle multiple fault diagnosis very well. Similar to the work mentioned above, other authors [2, 18] have proposed a diagnosis method using SLAT (single location at-a-time) patterns. A SLAT pattern is a test pattern that produces a failure on the tester and the circuit response exactly matches the simulation response of some single stuck-at fault. In other words, under a certain test pattern, no matter how many faults exists in the faulty circuit, a faulty response observed on the primary outputs can be explained by at least one single stuck-at fault. The pattern which can activate such a fault is a SLAT pattern. In this work, the author used all failing patterns and traditional fault simulation to construct a table which contains all the SLAT pattern/fault pairs. After that, the algorithm uses the constructed table to find the minimal set of faults called multiplet that can explain all the SLAT patterns. The multiplet may have many combinations which can be further analyzed by other methods like multiple fault simulation. Many works have been proposed based on this SLAT idea [17, 27, 47, 51]. These works try to find single

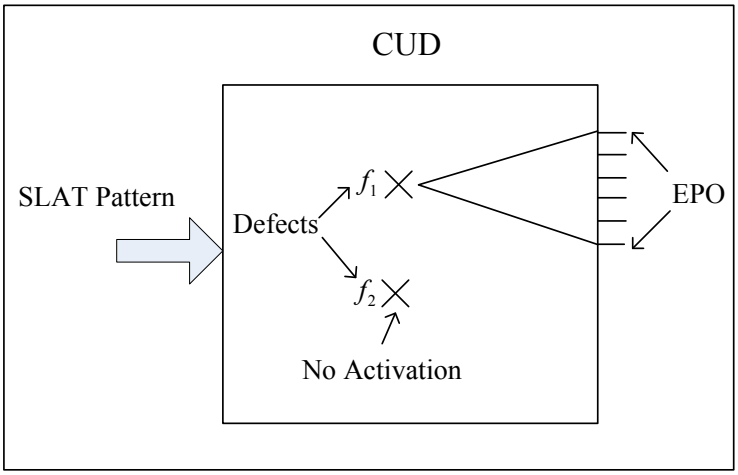

Figure 3.3: A SLAT pattern. EPO: erroneous primary output.

faults that can explain certain failing pattern responses. However, any method that relies on the existence of SLAT patterns will fail if there is no such pattern or the number of such patterns is not sufficient to locate all faults. What's more, for existing ATPG tools, it is hard to generate diagnostic test vectors to assure that there is only one fault activated and the faulty response can be observed on the primary outputs. Existing ATPG algorithm only guarantee that certain fault is detected without considering that other faults will not be activated at the same time. As shown in [2], although many test patterns have the SLAT ability, not all the faults can be properly activated when multiple faults exist. In [50], the authors propose a multiple fault diagnosis algorithm that can use some types of non-SLAT failing patterns together with SLAT patterns. However, when fault-canceling or fault reinforcement exist in the failing patterns, the result will be pessimistic.

Instead of using traditional simulation methods, in [32], a diagnosis method based on a backtrace simulation technique and a corresponding ranking system is proposed. The method used in this work first backtraces from every erroneous primary output to identify the possible fault candidate and, at the same time, the algorithm records how many times the fault has been counted. This record is used to levelize the fault candidates, which means if a fault has a higher count then the rank of the fault will be higher. However, this ranking

system runs into problems when multiple faults exist in the faulty circuit. Multiple faults may share the same propagating path and the faults in this shared path will have a high count and can partially explain the faulty behavior observed on the primary outputs. In such cases, the faults with the highest count in the ranking system may not be the right candidates.

All the works mentioned above focus on using single fault simulation information to deal with multiple fault diagnosis. Sometimes these methods will run into trouble when the faults interfere with each other. When the fault density increases, multiple-fault behaviors like fault enhancement or fault canceling will become common.

In [42], information of multiple fault injection and simulation is used to diagnose multiple stuck-at faults. The authors present a fault diagnosis algorithm which involves both single and multiple fault simulation. In this work, all faults that could exist in the circuit are first added to fault candidate set. After that, the algorithm uses the passing pattern simulation to remove the candidates that can be detected by these patterns. Then failing patterns are used to add the faults which share the common failing primary outputs between single fault simulation and the real faulty response of the CUD. Next, all the suspected faults are injected into the circuit and multiple fault simulation is performed under passing patterns. If the response observed on some primary outputs are different from correct value, then these outputs are recorded and single fault simulation is performed again on all the suspected faults under the same test patterns. Any fault that can cause the error on the recorded primary output will be removed from the suspected fault list. These procedures are recursively performed and the faults are added and removed from the suspected fault set. Finally a set of faults is found that can explain all the faulty behaviors or the number of candidate cannot be shrunk any more.

The flaw of this method is that it injects all the suspected faults into the circuit and performs a multiple fault simulation. As a result, unnecessary interfering effect among these faults can seriously affect the judgment. What's more, because of the recursive procedure, in

which fault simulation requires large processing time making the algorithm inefficient. The experimental results show that the diagnosis resolution is not very good.

Instead of injecting all the suspected faults into the circuit and performing the multiple fault simulation, in [15], the authors proposed a diagnosis method using extensive multiple fault enumeration and simulation. However, this method has an obvious shortcoming that if the size of initial fault candidate set is too large, then the time spent on enumeration and fault simulation would be tremendous because of the exponential candidates search space. So there is a need to guide that enumeration instead of randomly selecting candidates to construct the multiplets.

In [24], the authors presented a multiple stuck-at fault diagnosis method using a neural networks method called *particle swarm optimization*. In this work, first the critical-path tracing technique is used to identify all the suspected candidates that may cause a failure, then a "particle" which is a series of binary numbers representing the existence or non-existence of a fault is constructed. The main idea in this method is to change the combination of the particle and perform the multiple faults simulation. By comparing the simulation results and the real circuit response, the rank of the multiplet will be generated. The ranking is based on the number of explained failing patterns. The selection of the next particle is guided by the multiplet that currently has the best ranking. The diagnosis result of this work shows that it is much more time efficient than the methods based on random fault enumeration. There is one point of this work that needs improvement. That is the size of the initial set of candidates is too large, which means the length of the particle will be very long. More effort should be used to shrink the size of initial candidate set.

Besides conventional simulation methods, X-simulation is also used for multiple fault diagnosis [4, 8, 40]. These methods are targeted at solving arbitrary faults including multiple stuck-at faults that occur in a small connected area. The authors in [8] present a concept called Xlists to represent a set of nodes whose actual values are replaced by an uncertain value of X, and the signal propagation of these nodes will use 3-value (0, 1 and X) logic

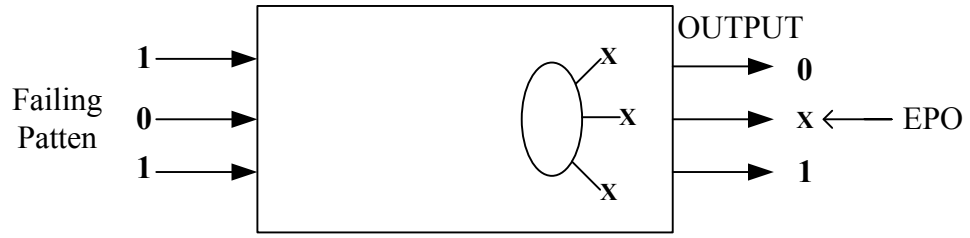

Figure 3.4: X region may include the real defect. EPO: erroneous primary output.

simulation. By using this concept, enumeration of combinations of the output of the small area is avoided. After the 3-value logic simulation and a comparison of the response with that of the real faulty circuit, if the X value can be propagated to the faulty primary outputs, then the nodes with X injection could be the real fault sites. As shown in Figure 3.4, under the failing pattern, if the output X symbol of the region can propagate to a faulty output, then it is possible that this region includes the real fault. Otherwise, as in Figure 3.5, the X symbol cannot propagate to the primary output with error, so it could not be the region that includes the real fault. In [4], the authors first partition the circuit into small areas with just one or two gates. Then inject the X value on the output boundary of a region and perform the 3-value logic simulation mentioned above. Each region is ranked by the matching situation after the simulation:

- If the X value can propagate to primary outputs with erroneous response, then a weight of matching will be added to the count of the region.
- If a contradictory value of '0' or '1' propagates to a primary output with no error observation in the circuit test, then the region is dropped from the candidate list.
- In other cases, the count of the region remain the same.

In [40], based on the work of [4], the authors proposed two techniques for improving the diagnostic resolution. One is called *flipping-fanout-bits*. In this method, for each region, a flipping operation is performed on every output signal, all except one outputs of the region

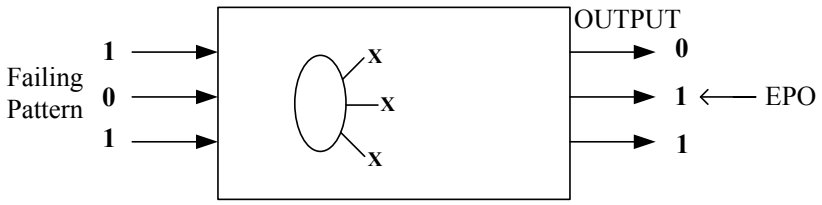

Figure 3.5: X region cannot include the real defect. EPO: erroneous primary output.

are forced to an 'X' value, the left output is flipped to its opposite value, and 3-value logic simulation is performed. By observing the simulated values on the primary outputs, if in all flip cases, at least one erroneous output remains or an additional error is observed, then the region should be abandoned. The other method is called *distinguishing* Xs. In this method, every X is injected in the circuit with a unique sign. This sign is used to record the times that the X is inverted in the signal propagation and determine the output value of the gate that the X value generated from.

All of the three works mentioned above on X simulation have the same limitation that the diagnosis resolution is low. This is because X simulation is a conservative method. What's more, all of these works are limited to a small area, usually these works only assume that the region contains only one or two gates which is not practical in real diagnosis.

3.2 Open Fault Diagnosis

Open fault model usually is used for interconnection defect diagnosis. Much area of modern ICs is taken by the metal interconnects compared to routing within the cell, so a break is more likely to happen on the interconnect wiring metal than in the cell [46].

In [46], the authors proposed an open fault diagnostic model which is used to model the faulty behavior in the presence of an open defect on an interconnection line. This diagnostic model is a superset of all possible behaviors of a net under single stuck-at fault assumption. As shown in Figure 3.6, the model treats each interconnection net as a diagnostic unit. All the single fault simulation information of stuck-at-1 or stuck-at-0 faults of the stem and its

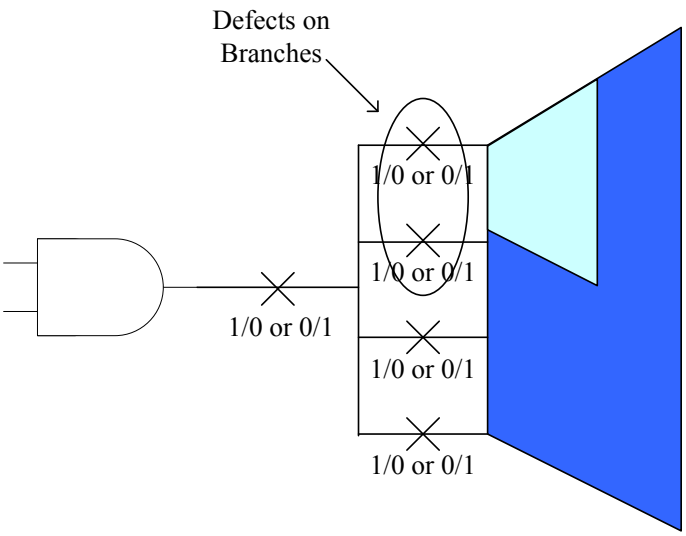

Figure 3.6: A net diagnosis model.

branches are included in the net influence. If such superset of a net influence can include all or most of the faulty behaviors, then this net will be selected as a candidate. Then influence caused by the real defects is a subset of the net influences. After the first level selection, a path-tracing technique is used to further reduce the number of candidates. This technique can solve the single open fault very well, but is hard to apply to multiple open faults on different nets, because the matching system of the method is based on the single net fault assumption. When faults on multiple nets can be activated on the same test pattern the matching system will not work.

In [14], a symbolic simulation method is proposed for open segment fault diagnosis. This work aims to identify the faulty segments of a net with open defect. In this work, a series of symbols are injected on each branch of a suspicious net and symbolic simulation is performed as shown in Figure 3.7. Each influenced primary output will have an algebraic expression constructed by these symbols to represent the real value of '1' or '0'. The stuck-at value of each faulty branch can be determined by solving the set of equations by relating the faulty responses and the algebraic expressions together. Figure 3.8 shows an example of how

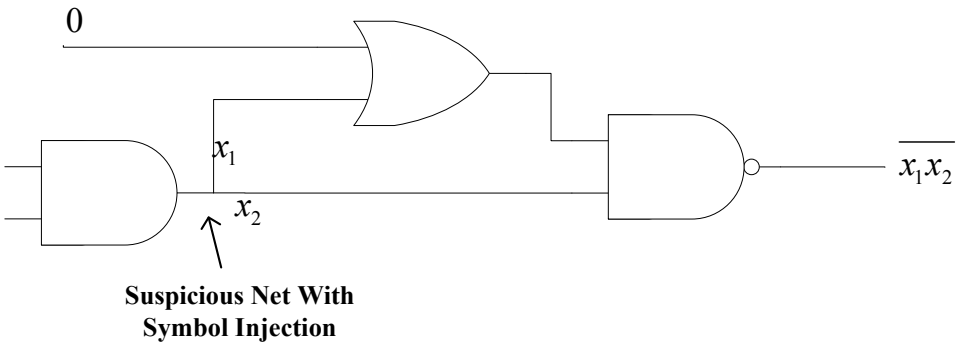

Figure 3.7: Symbol injection and propagation.

this comparison works. From the comparison, we can determine that the values X_2, X_3 and X_4, are '0', '1' and '0', respectively. Thus, we have the information on the stuck-at values of the faulty branches. The limitation of this work is that it can only solve the problem for a circuit of limited size because symbolic propagation requires a large memory space to store the algebraic expression for each node, especially when more open segment faults exist in the circuit.

In [22, 23], the authors propose a diagnosis method based on integer linear programming (ILP) technique. The main idea of this work is to find smallest possible number of candidates to explain the observed faulty behavior. The authors propose a three-stage method to diagnose multiple open defects in combinational circuits. First, the method starts with a path-tracing procedure to identify potential defect sites that can cause errors on the faulty outputs. Each net is weighted to indicate its potential to explain the faulty behavior. In the second stage, several constrains are formed with variables corresponding to the occurrence of defects at each potential defect site. By solving the problem within these constraints, the results of ILP are treated as possible fault combinations for the multiple defect diagnosis. In the final stage, fault combinations that cannot explain all the failing output responses are removed and physical information is taken into account to accurately determine metal segments that can explain the faulty behavior. Since ILP is an NP-complete method and it

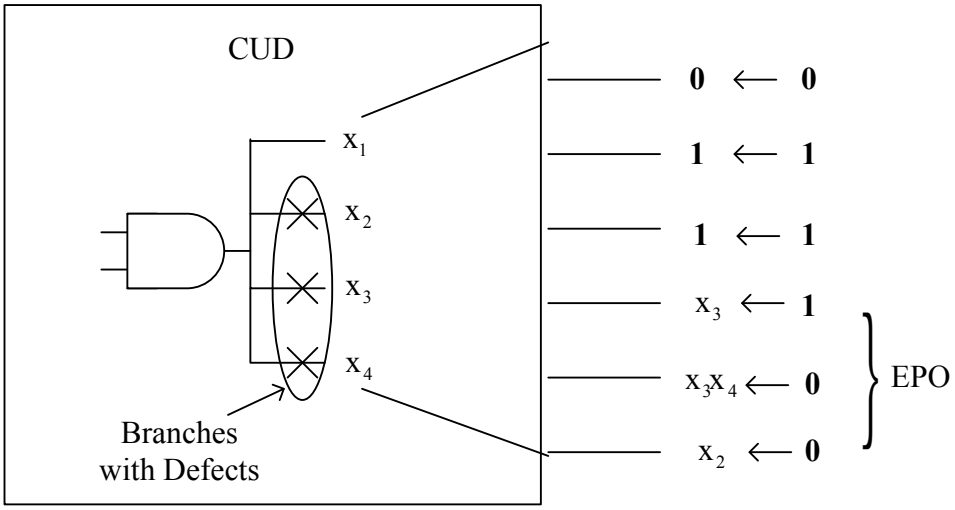

Figure 3.8: Comparison between faulty response and symbolic simulation results. EPO: erroneous primary output.

involves exhaustive searching of solutions, when the size of the test circuit or the number of defects increase, time spent on computing will be huge and the method will not be practical.

Chapter 4

A Fault Candidate Filtering System

The first stage of the proposed diagnosis algorithm is a fault candidate filtering system. In this chapter, we will first discuss the motivation of the proposed system and then give a detailed description of this system.

4.1 Motivation

As we mentioned in Chapter 3, the diagnosis of multiple faults is much more difficult than that of single faults. This is because fault dropping can not be performed during multiple fault diagnosis and their interfering effects make the output matching harder. Even though in recent works in which multiple fault injection and simulation techniques are used, the exponential fault searching space is still the most difficult part of multiple fault enumeration. So a critical part of multiple faults diagnosis is how to shrink the size of the initial fault candidate set. Then other techniques can work effectively on the reduced initial candidate set. Since we try to avoid multiple enumerations in the first stage candidate reduction process, single fault simulation information is the main reference that we are concerned about.

In Chapter 3, we have introduced many previous works which utilize single fault simulation information to reduce the number of fault candidates. Multiple fault diagnosis methods based on X simulations and region-model are not specific-model-based methods, so the results of such methods are conservative and still not accurate enough to be used. SLAT methods try to find patterns that can activate a single fault at a time [2, 3, 12, 17, 27, 51]. However, these methods heavily rely on the single-fault activation ability of the test patterns. Other works are mainly based on certain fault candidate ranking systems. No matter what

kinds of fault simulation techniques these systems use, the ranking or filtering criteria are almost similar and have their limitations.

Most previous multiple fault diagnosis methods that utilize ranking and filtering systems, rely on the simple or weighted accumulated count based on the matching situation of primary outputs [7, 32, 47]. This means the more matching outputs that a fault has, more possible it is that this fault really exists in the circuit. However, these methods ignored two important scenarios in fault diagnosis. First, during the process of test pattern generation, ATPG tools usually tend to select the same propagation path for a certain fault. This may not have any influence on single fault diagnosis, but when there are multiple faults existing in the circuit, and these faults share the same propagation path, then the problem may appear. Faults on these propagation paths may have higher number of detection instances than the real faults. This can incorrectly rank the suspected faults. Second, because some faults are hard to activate by certain test sets, their detection times are much lower than other faults. Although an N-detection test set can increase the overall detectability of all the faults, it still cannot change this non-uniformity of the detection times of faults. It is not fair to just use the simple or weighed sum of matching times to rank the candidates.

A simple case is illustrated in Figure 4.1. In this case, we assume there are three faults f_1, f_2 and f_3 whose fault effects propagate through the same path. When they arrive at the AND gate inputs, they all perform as stuck-at-0 faults and we assume they can be detected separately by the test set, which means they will not be activated together under the same pattern. The total detection times of these three faults are shown in the figure, respectively. Fault f_4 is a stuck-at-0 fault on the output of the AND gate, so the total activation times of f_4 would be 20. The single fault simulation information of these four faults is listed in Table 4.1. Now we assume that f_1 and f_2 are the real faults that exist in the circuit and patterns P_1 through P_{17} fail in the CUD test. If we do not consider the passing pattern detection penalty, which means a fault is activated by certain pattern in the single fault simulation while this pattern passed in the CUD test, then a negative count will be given

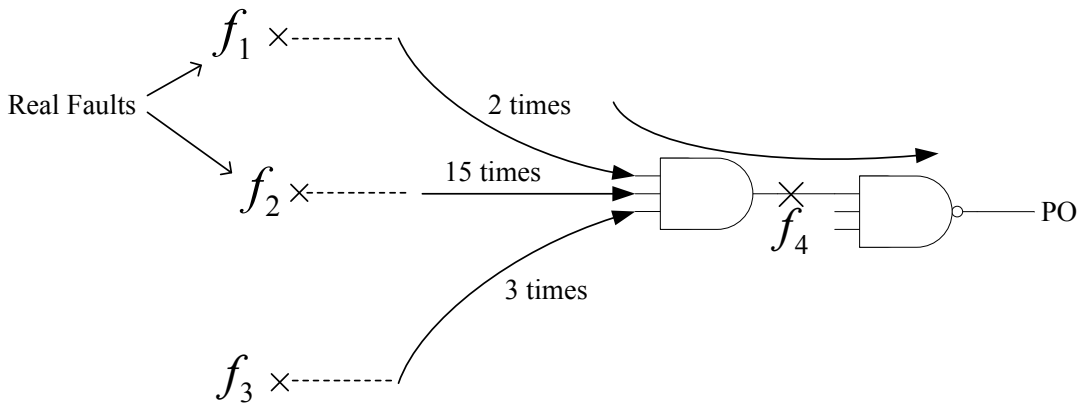

Figure 4.1: Multiple defects sharing the same propagation path.

for the fault. We do not consider such penalty weight, so the counts for f_1, f_2, f_3 and f_4 will be 2, 15, 0 and 17, respectively. From this calculation we see that because of this kind of circuit structure, instead of ranking f_1 and f_2 at the top, f_4 becomes the top candidate in the diagnosis. If we consider the passing pattern penalty and because we have to consider the interfering effect, like fault canceling, between the multiple faults, we cannot set the penalty weight too high. So we set the negative count weight the same as the positive count weight and the count for these faults will be 2, 15, 0 and 14. From the calculation we can see that f_4 still has a much higher count than f_1. In a real CUD case, there could be many more defects than in this simple case, so previous ranking methods will introduce many false faults, like f_4, in fault diagnosis and ignore faults like f_1 that have much lower detection count than other faults after applying some selected threshold value. What makes it worse is that these faults may lose the chance to be the fault candidates for further analysis after the candidate filtering because of the low rank.

So in this thesis, we want to first propose a novel candidate filtering system which can reduce such effects caused by the circuit structure and the non-uniformity of the test set with respect to multiplicity of detecting faults. The single fault simulation information that we utilize includes both failing pattern index information and passing-failing outputs information

Table 4.1: Single fault simulation information.

Fault	Failed Pattern Index
f_1	P_1 - P_2
f_2	P_3 - P_{17}
f_3	P_{18} - P_{20}
f_4	P_1 - P_{20}

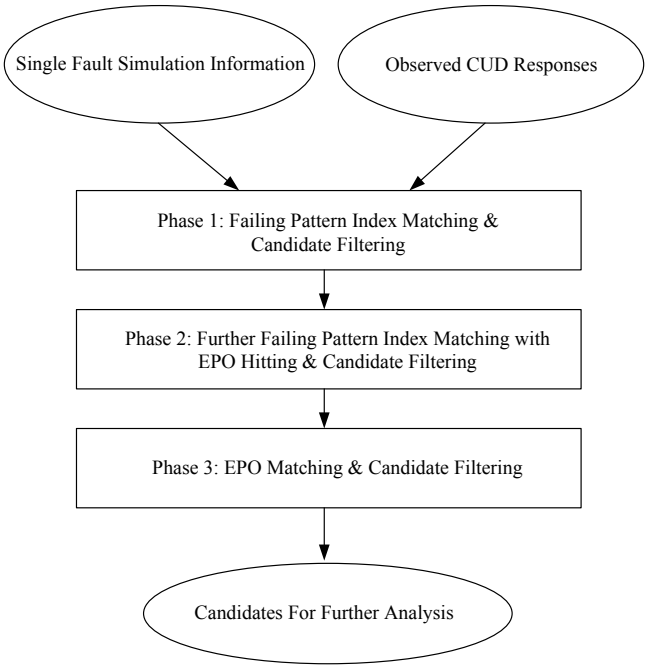

Figure 4.2: Candidate filtering system.

under failing patterns. This system contains three phases: *Failing-pattern-index-matching (FPIM) and candidate filtering*, *Further FPIM with erroneous primary output (EPO) hitting and candidate filtering* and *EPO-matching and candidate filtering*. The overall candidate filtering system is shown in Figure 4.2 .

4.2 Failing-Pattern-Index-Matching and Candidate Filtering

For introducing this diagnostic phase, we first require some definitions.

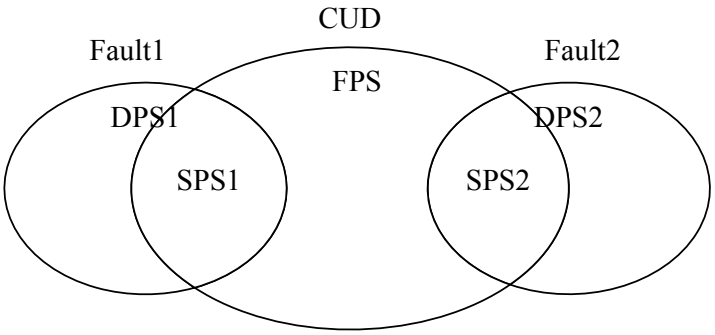

Figure 4.3: Relationships of DPS, SPS and FPS.

Definition 4.1 *The union of all the failing pattern indexes from the single fault simulation of a fault candidate is defined as the* detectable pattern set (DPS) *of this fault under test.*

Definition 4.2 *The union of all the failing pattern indexes of the observed CUD response is defined as the* failing pattern set (FPS) *of the test.*

Definition 4.3 *The shared part between the DPS of a fault candidate and the FPS of CUD is called the* shared pattern set (SPS) *of this candidate.*

Theorem 4.1 *If the CUD is a circuit with multiple faults and we assume that all the multiple faults in the circuit will not totally cancel (mask) each other at primary outputs, then the* DPS *of any one of the multiple faults in the circuit should be a subset of* FPS *of the test. In other words, the* SPS *of the fault candidate equals its DPS.*

The relationships of sets according to Definitions 4.1, 4.2 and 4.3 are shown in Figure 4.3. Based on Theorem 4.1, we can conclude that generally the more SPS covers the DPS of a fault candidate, greater is the possibility that this fault is the real one in CUD, so we can calculate a count for each fault according to how much SPS overlaps onto the DPS of the candidate, Thus, in general, higher the ratio of these two values, more likely it is that the candidate exist in the circuit. The count for each fault is calculated as follows:

Table 4.2: An example of pass (p)-fail (f) information of CUD and faults.

Pattern Index	CUD	Fault 1	Fault 2	Fault 3
P1	p	p	p	f
P2	p	p	p	f
P3	p	p	p	f
P4	f	f	p	f
P5	f	f	p	p
P6	f	f	p	p
P7	f	f	f	p
P8	f	p	f	p

Table 4.3: Information from Table 4.2.

FPS	DPS		
	Fault 1	Fault 2	Fault 3
P4-P8	P4-P7	P7-P8	P1-P4

$$\text{Count} = \frac{\text{Number of patterns in SPS}}{\text{Number of patterns in DPS}} \tag{4.1}$$

To better illustrate the count calculation, we use a simple example shown in Table 4.2. In Table 4.2, 'f' indicates the CUD (on actual test) or the fault (on simulation) shows 'failure' under the corresponding pattern and 'p' indicates the CUD or fault 'passes' the pattern. So from this data we can get the information shown in Table 4.3. Using this information, we can calculate the count for each fault as shown in Table 4.4. From this table we can see that because DPS of Fault 1 and Fault 2 are both subsets of FPS of CUD, they have higher counts than Fault 3.

Thus the Theorem 4.1 tells us that, at least in an ideal sense, the detectable pattern set of a real fault in the circuit must be a subset of total failing pattern set of CUD. Then the count for the fault would be 1, like Fault 1 and Fault 2 in the preceding example. However, in a practical situation, multiple faults in the circuit may totally cancel (mask) each other

Table 4.4: Calculation of count in first phase.

	cardinality of SPS	cardinality of DPS	Count for Fault
Fault 1	4	4	$4/4 = 1$
Fault 2	2	2	$2/2 = 1$
Fault 3	1	4	$1/4 = 0.25$

under certain patterns, then the count of the real faults may be lower than 1 and we have to set a margin to handle this case so that the real faults that have such masking will not be pruned out because of the over-constraint. In this work, we will set a threshold value to filter fault candidates. In this first phase of fault diagnosis, all collapsed faults in the circuit will be initial candidates and the calculation mentioned above will be applied to them. It should be noted that we use a ratio value for the count instead of simply using the number of matching patterns. This is because we want to reduce the non-uniform number of detections across faults, as mentioned in the motivation part, which is caused by the circuit structure and by relatively hard to detect faults. We want to give an equal opportunity to each fault no matter how many times the fault can be detected by the test set. Therefore, in this work we always use ratio values as the criteria to weigh and filter out fault candidates.

4.3 Further Failing-Pattern-Index-Matching with EPO Hitting and Candidate Filtering

In the first diagnosis phase, we only considered how much that detection pattern set of a fault candidate overlaps with the failing pattern set of CUD. However, only considering those patterns is not enough. Because through observation from experiments we found that some fault candidates may show detection upon simulation under a failing pattern of CUD, but they do not affect the actually observed failing primary outputs of CUD. In other words, their erroneous primary outputs (EPOs) have no shared part with the ones observed from

CUD. So in the second phase we consider for each pattern in SPS, whether the candidate simulation response and the test response from CUD have at least one shared EPO.

This phase of candidates filtering is called *further failing-pattern-index-matching with EPO hitting*. To better understand the procedure of this phase, we first give a definition for 'EPO hitting':

Definition 4.4 *Under certain test pattern, if the affected primary outputs of a candidate fault upon simulation share at least one common erroneous output with the faulty response of CUD, then we say that this fault candidate can 'hit' the EPO under this pattern and the pattern is called a* hit-pattern *of the candidate fault.*

Theorem 4.2 *If the CUD is a circuit with multiple faults and we assume that there is no canceling (masking) effect among those faults, then the number of* hit-patterns *for any one of these faults in the circuit should be the same as the number of patterns in the respective* SPS.

The procedure of this phase is based on the Theorem 4.2, from which we conclude that for a candidate fault if more hit-patterns are in SPS, greater will be the possibility that the candidate fault exists in the circuit. So we utilize another ratio measure as a count for each fault. It is defined below:

$$\text{Count} = \frac{\text{Number of Hit-Patterns in SPS}}{\text{Number of patterns in DPS}} \tag{4.2}$$

This count is similar to the one in the previous phase with one difference, that is, we only count the hit-patterns in SPS instead of all patterns in SPS. For the same reason as discussed in motivation and phase 1, we use a ratio value instead of the total number of hit-patterns for the candidate. Similarly, because we want to consider the canceling (masking) effect among faults, we set a rather relaxed threshold value to avoid over-constrained filtering of fault candidates.

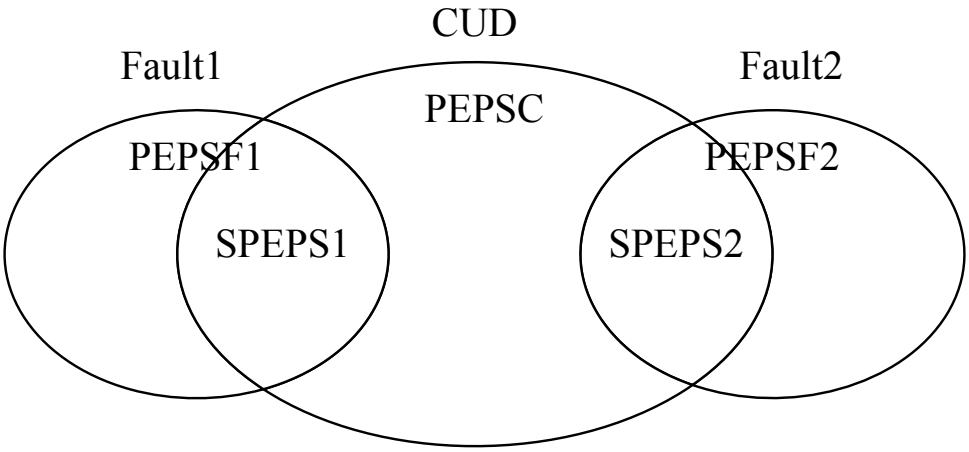

Figure 4.4: Relationships of PEPSC, PEPSF and SPEPS.

4.4 EPO-Matching and Candidate Filtering

For each failing case of the faulty circuit under a given pattern, there can be some faulty outputs and other passing outputs. Until now we only used the faulty output information. However, passing outputs provide additional information, which can improve the diagnosis. Hence the second phase of our diagnosis procedures. Several definitions are relevant to this phase.

Definition 4.5 *A* pattern-EPO-pair (PEP) *is a pair of failing pattern number and an EPO associated with it. For example, [P2, PO1] is a PEP indicating that under pattern P2, we observed an error on primary output PO1. A PEP could be used to represent either an observed faulty response of CUD or the fault simulation results of a fault candidate.*

Definition 4.6 *The union of all the PEPs from CUD test is called* PEP-set-of-CUD (PEPSC) *and the union of PEPs under all the patterns in SPS of a fault candidate is called the* PEP-set-of-fault (PEPSF). *The shared part of PEPSC with a fault candidate PEPSF is called* shared-PEP-set (SPEPS) *of that fault candidate.*

The sets of Definition 4.6 are shown in Figure 4.4. The procedure of this phase is based on the following theorem:

Theorem 4.3 *If a CUD has multiple faults and we assume that there is no canceling (masking) effect among these faults, then the* PEPSF *of a fault candidate in single fault simulation should be the subset of* PEPSC *of CUD and the* SPEPS *of the fault candidate should equal its* PEPSF*.*

From Theorem 4.3 we can conclude that, in general, the more SPEPS overlaps the PEPSF of a fault candidate, more are the chances that this fault is included among the multiple faults of the circuit. Therefore, we can calculate a count for each fault based on this theorem, using the following equation:

$$\text{Count} = \frac{\text{Number of PEPs in SPEPS}}{\text{Number of PEPs in PEPSF}} \tag{4.3}$$

To illustrate how the count is calculated for each fault, we take a simple example shown in Table 4.5. The data in the table indicates that PEPs in PEPSF of Fault 1 are [P1, PO1], [P2, PO1], [P2, PO3] and PEPs in PEPSE of Fault 2 are [P1, PO2], [P1,PO3], [P2, PO2], [P2, PO3]. Also, we find that PEPs in SPEPS of Fault 1 are [P1, PO1], [P2, PO1], [P2, PO3] and PEPs in SPEPS of Fault 2 are [P1, PO2], [P2, PO2], [P2, PO3]. So we can get the count of Fault 1 as $3/3 = 1.0$ and count of Fault 2 as $3/4 = 0.75$.

4.5 Threshold Value Determination

We will use three-step filtering to narrow down the diagnostic resolution. For each step, we need to set a threshold (TH) value for the computed count value to filter unnecessary fault candidates out. A higher threshold (close to 1.0) tends to reduce the suspected fault group thus improving the diagnostic resolution. However, this has the potential for filtering the real fault away and produce a misleading diagnosis. On the other hand, too low a threshold

Table 4.5: A full-response example.

	Pattern Index	Failing Outputs
CUD	P1	PO1, PO2
	P2	PO1, PO2, PO3
	P3	None
Fault1	P1	PO1
	P2	PO1, PO3
	P3	None
Fault2	P1	PO2, PO3
	P2	PO2, PO3
	P3	PO2

might preserve the real fault candidate but group it with a large number of other suspects thus lowering the diagnostic resolution.

The TH values depend on the our assumption on fault density of the circuit. Usually we set these TH values with conservative assumption on fault density. In our work, we use a training system to set the TH value for each candidate filtering step. In the proposed training system, we determine the TH values for each step one at a time. Because the constraint in phase one is the easiest and that in phase three is the hardest to satisfy, we first start from phase one and finally end at phase three. When we try to set one TH value, other two TH values must be fixed and when a TH value is determined, it will not change for the rest of the training procedure. For each TH value, we run several rounds of experiments to determine its value. For each round of experiments, we constructed 20 faulty circuits from each benchmark circuit and for each faulty circuit we randomly injected four stuck-at faults. By running the candidate filtering system, we can get the list of remaining faults and count how many injected fault can survive from the filtering procedure. The percentage of total surviving faults out of the total injected faults is defined as survival rate. For each phase we have different survival rate requirements as shown in Figure 4.5. The starting value of each TH is 1.0. After a round of experiment, if the survival rate requirement is satisfied, then the TH value of this phase is determined as the current value. If the survival rate requirement is

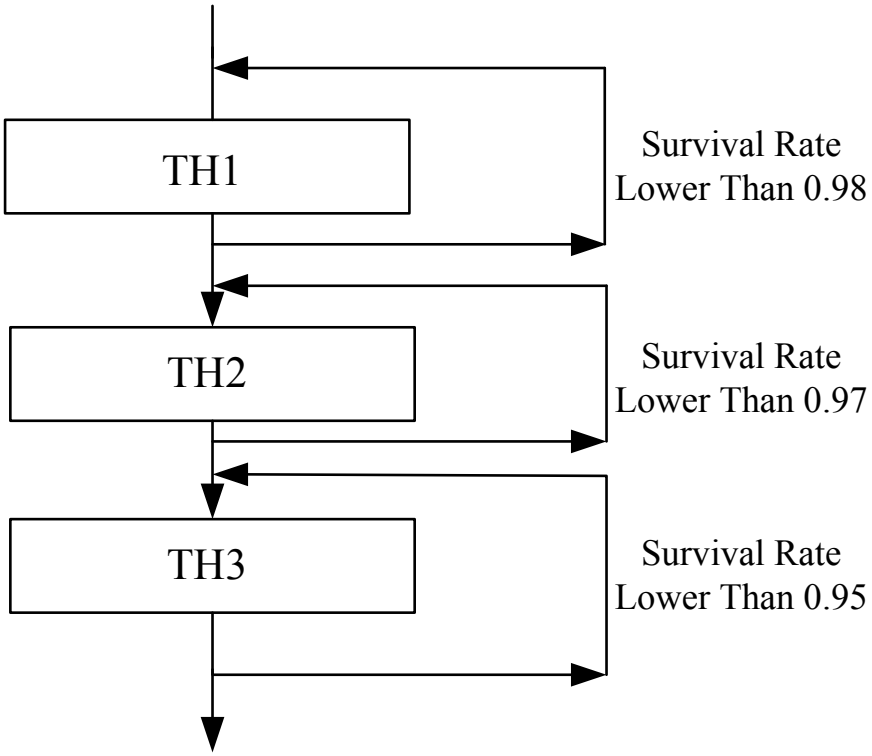

Figure 4.5: Training procedure for determining filter thresholds (TH) on counts.

not satisfied, then the current TH value is lowered by 0.1 and another round of experiments is performed. This procedure continues until the survival rate is satisfied.

4.6 Experimental Results on Candidate Filtering

In this work, all the experiments are conducted on ISCAS85 benchmark circuits. Because it is not possible to have too many faults in the CUD, in this work we assume that in each benchmark circuit, there are probably four multiple faults in the circuit. According to this assumption, we perform the training procedure to determine the TH values for each benchmark circuit. In order to see the effectiveness of the reduction stage, we randomly constructed 50 faulty circuits for each benchmark circuits, and for each faulty circuit we randomly injected 4 stuck-at faults. The reduction results is shown in Table 4.6. The reduction rate is defined as the number of abandoned faults over total number of faults. The

Table 4.6: Results of candidate filtering system.

Circuit	# of total faults	Reduction Rate	Survival Rate
C432	524	0.77	0.975
C880	942	0.97	0.96
C1355	1574	0.75	0.97
C1908	1879	0.85	0.95
C2670	2747	0.925	0.97
C3540	3428	0.95	0.965
C6288	7744	0.993	0.96
C7552	7419	0.992	0.96

survival rate is defined as the ratio of the number of remaining injected faults after reduction process over the number of injected faults. From the experimental results we can see that large partial of the fault candidates can be reduced by our method.

Chapter 5

Candidate Ranking System

The previous chapter described the first stage of the diagnosis procedure, namely, fault filtering. In this chapter we will continue to discuss the fault ranking method that we use in our diagnosis procedure.

5.1 Motivation and Structure for Ranking System

After the filtering stage of diagnosis, a relatively small list of candidates can be obtained. However, the fault filtering stage gives us a group of candidates that is likely to include the real faults, which means the result is conservative. In order to have a better resolution in diagnosis, we need to rank the fault candidates and put the most suspicious faults at the top of the candidate list. The proposed ranking system is actually a probabilistic ranking system. The final rank of each fault indicates the possibility that the fault will exist in the circuit compared to other fault candidates. Higher this fault rank, more likely is the fault to exist in the CUD. The structure of the ranking system is shown in Figure 5.1.

We start with the filtered candidate list. At this stage, we want to reduce the influence of any lack of diagnostic capabilities in the test set. We group the undistinguished fault candidates and let one fault in each group represent all members of that group. Because the test sets we use in this work [52, 53] have high diagnosability, after we get the reduced (filtered) list of candidates from the previous level, we directly process that list and group undistinguished faults by comparing the simulation results, as shown in Figure 5.1. For each group, we assume that its members are functionally equivalent; to be exact, they are *conditionally equivalent* [38, 39] with respect to the test set. In net fault diagnosis, we will recover the candidate list. Detailed recovering procedure will be discussed in a later chapter.

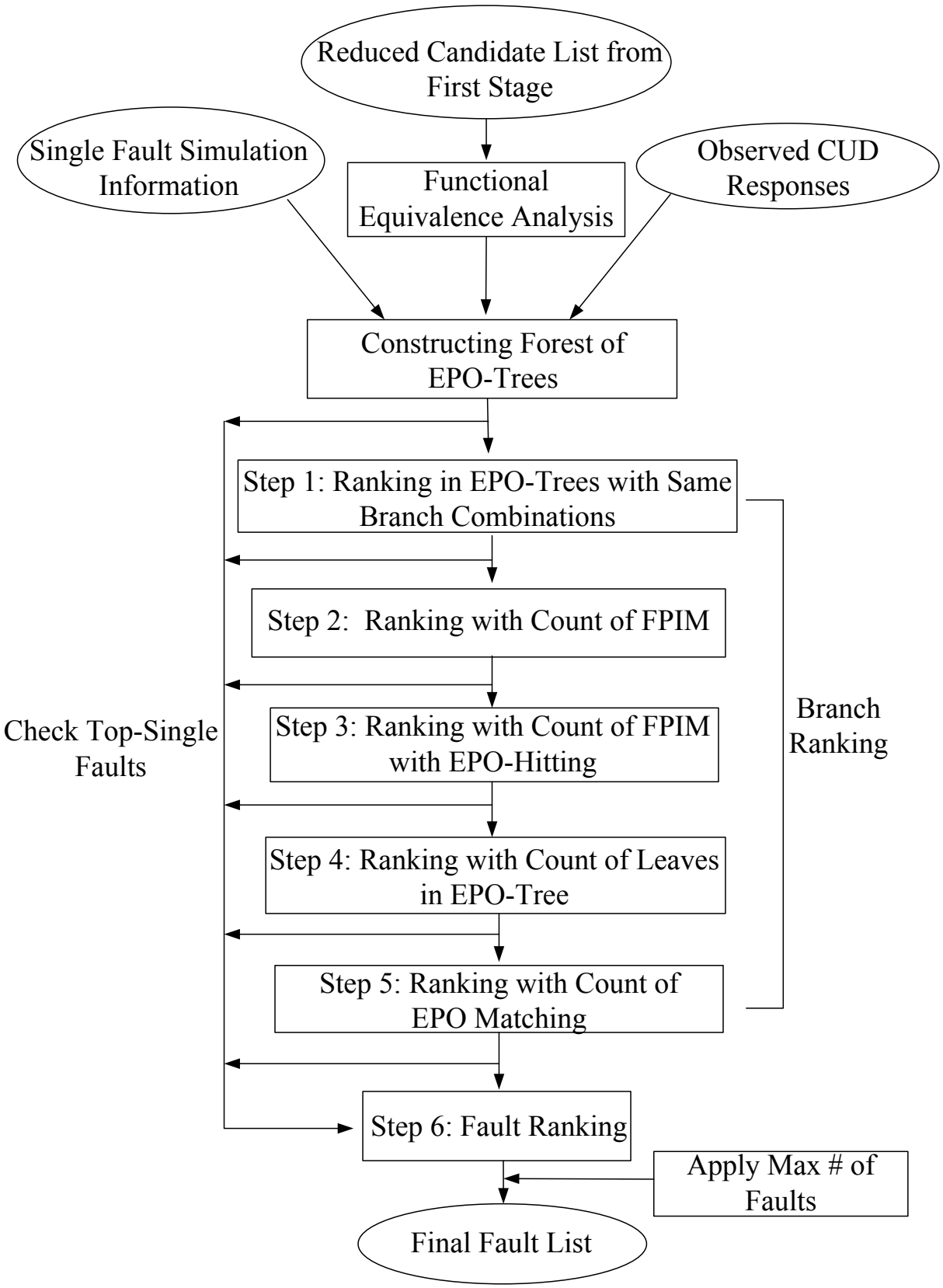

Figure 5.1: Fault candidate ranking system.

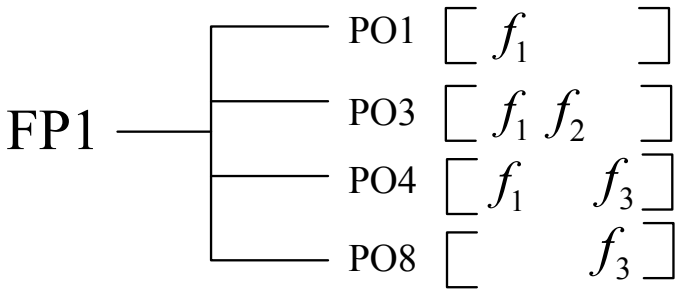

Figure 5.2: An example of EPO-tree.

The proposed ranking system uses a tree structure, called *EPO-Tree*, that relates the fault simulation information to the circuit response. A sample EPO-Tree is shown in Figure 5.2. In this figure, the root of the EPO-tree is a failing pattern from the test set of CUD, shown as FP1. The erroneous primary outputs (EPOs) for this pattern are the branches of the tree, e.g., PO1 and PO3 in the figure. The leaves on each branch are the fault candidates that can cause an error on the corresponding EPO according to their single fault simulations. From the figure we can see that, all fault candidates that can cause at least one EPO to fail are listed as the leaves of the EPO-Tree. We can thus analyze the effects caused by these potential fault candidates in each step of the ranking procedure. A collection of EPO-Trees that include all the information of CUD test and the fault simulations is called a *forest* of EPO-Trees.

Having introduced the basic analysis structure, we can now describe the ranking system. As shown in Figure 5.1, the proposed ranking system includes six steps: *ranking in EPO-trees with same branch combinations*, *ranking with count of FPIM*, *ranking with count of FPIM with EPO-hitting*, *ranking with leaf count in EPO-tree*,*ranking with EPO matching* and *fault ranking*.

In this ranking system, there are two types of 'ranking' procedures. The first is branch-ranking, which is performed in each of the first five steps. The other is the final candidate ranking, which is performed after the branch ranking. The first five analysis steps rank fault

candidates within each branch of every EPO-tree according to different measurements. The branch ranking is a step-by-step process, the constraint that we add in each step becomes tighter and tighter gradually. In other words, the constrain in the first step is most relaxed and that in the last step is the tightest. The reason why we follow this kind of sequence is that the ranking procedure in each step is based on the results from the previous step. If we set the constraint too tight in an initial step, we may eliminate a candidate before it is examined in detail. As shown in Figure 5.1, in steps one and four, we make some intuitive assumptions based on the observations in the experiments and the rankings of candidates are based on those assumptions. In steps two, three and five, we use the three count values that we got from candidate reduction stage as the measurements to rank the candidates. After collecting the ranking information of all EPO-trees, the final candidate ranking procedure is carried out and gives out a ranked fault list. After applying a maximum value on number of candidates, we finally get a list of fault candidates to be used in the net diagnosis (next chapter).

5.2 Branch Ranking Procedure

As mentioned before the branch procedure includes five steps. For each of these five steps, we will use either an intuitive assumption or certain count value we have obtained in previous stage to perform the branch ranking. A novel part of our analysis procedure is that a fault candidate may have different ranks in different branches. We perform the analysis only within each branch, which means even in the same EPO-tree but in different branches a fault candidate may have different ranks.

5.2.1 Branch Ranking in EPO-Trees with Same Branch Combination

In the first step of analysis procedure, we rank the involved fault candidates based on following observation from experiment:

Observation 5.1 *The activation situations are sometimes similar under certain test patterns, which means these patterns can trigger same set of injected faults in the CUD and the observed EPO combinations from the CUD are the same. Such a situation is usually hard to be repeated by other set of injected faults.*

Based on this observation, we make an intuitive assumption on fault behaviors:

Assumption 5.1 *Assuming we have a circuit with large enough number of primary outputs, when the failing outputs combinations are the same under different test patterns, because it is not very easy to repeat the same combinations for different injected faults in CUD, it is possible that the causes of these failures are identical. If we can find shared set of faults between the EPO-trees under these failing outputs, then these shared faults are more likely than other faults in branch to be the real faults.*

Based on this assumption, we developed a ranking method that we can use to rank the faults in the branches of this kind of situation. In reality, we use a simple method which just records the total existing count of each fault in the corresponding branches, and then rank the faults in each branch by comparing their counts. The example in Figure 5.3 illustrates how we rank candidates in each branch. In this case, failing patterns FP1 and FP3 have the same EPO combinations: PO1, PO3 and PO4. Now we assume that it may be possible that the cause of these EPOs is the same set of faults. First, we divide the fault candidates into different groups according to the branches that these faults belong to as shown in the figure. By counting the number for each fault candidate, we can rank faults in the group. When two candidates have the same count in the same group, we will give them the same rank, like f_4 and f_2 in group PO1 and f_2 and f_7 in group PO3. After the ranking, we see that f_4 and f_2 are ranked at the top of group PO1, f_4 is ranked at the top of group PO3 and f_5 is ranked at the top of group PO4.

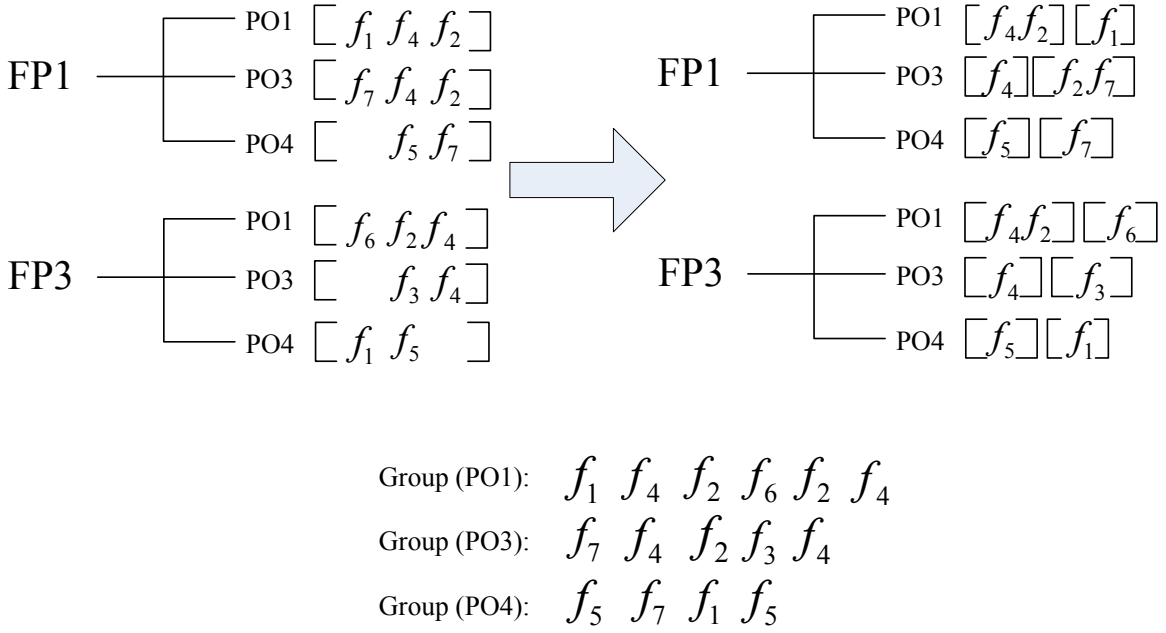

Figure 5.3: An example of ranking procedure of step one.

It should be noted that in this step of analysis, only the EPO-trees with the same branch arrangements as others will be considered. EPO-Tree with unique branch arrangements will not be considered in this step.

5.2.2 Branch Ranking with Counts from Reduction Stage

We now discuss the branch ranking procedure used in steps two, three and five. The reason why we jump across step four is that the procedures of the three steps are similar: they all use the counts of each fault from the reduction stage to rank the candidates in each branch.

As we have discussed in Chapter 4, we use three counts for each fault to reduce candidates in the candidate reduction stage, now we want to use these counts again to rank the candidates in each branch. In step two, we use the count of failing pattern index matching (FPIM). In step three and step five we will use the count of FPIM with EPO-hitting and EPO matching, respectively. Let us take an example of step two to illustrate the method.

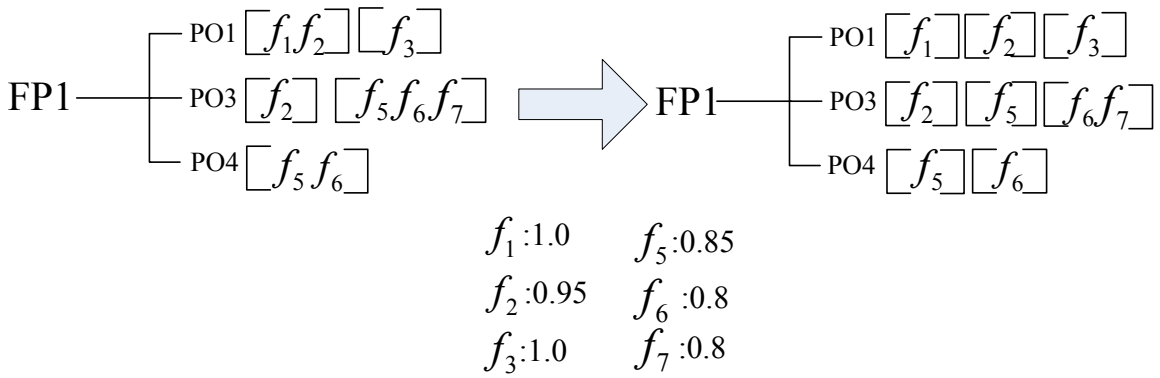

Figure 5.4: An example of ranking procedure of step two.

In Figure 5.4 we can see that after the first step, the candidates in each branch already have an initial rank. Next, we utilize the FPIM count of each fault obtained from reduction stage ranking of candidates within each already ranked group. The FPIM count of each fault is listed in the figure. Based on this count, each fault group with more than two candidates is analyzed and ranked as shown in the figure. Here the 'group' we mentioned refers to the candidates that have the same rank in previous ranking steps in certain branch, like f_1 and f_2 in branch PO1 in this case. Each group may have one or more candidates. In branch PO1, f_1 and f_2 are in the same group. Because the count of f_1 is higher than f_2, f_1 is put into a group with higher rank and f_2 is ranked below f_1. It should be noted that even though f_3 has a count higher than f_2, it still cannot be ranked higher than f_2 because no candidate in our ranking system would get a rank better than the existing rank before the present ranking began. Otherwise, the previous ranking steps will become meaningless. In steps three and five, the procedures are similar to the one just described and we will not discuss them anymore.

5.2.3 Branch Ranking with Leaf Count in EPO-Tree

In step four, we will consider the number of leaves that each candidate has in the EPO-tree. The ranking procedure in this step is based on the following assumption:

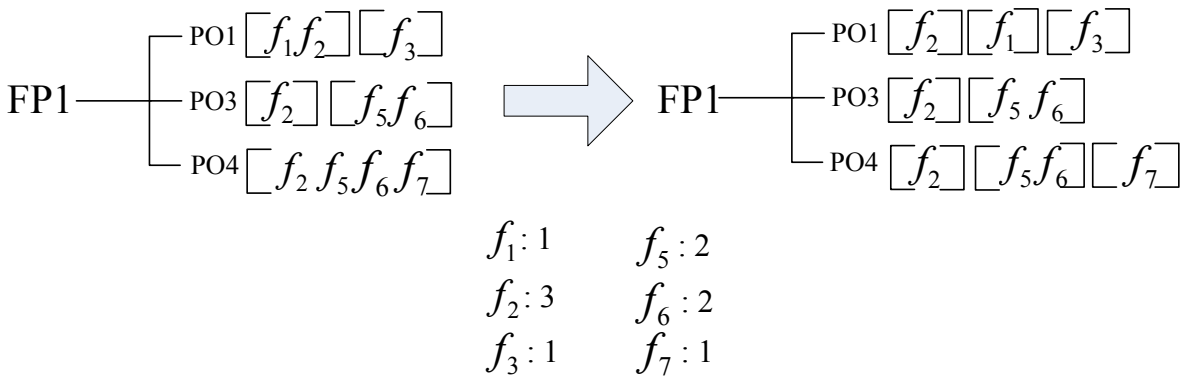

Figure 5.5: An example of ranking procedure of step four.

Assumption 5.2 *If several fault candidates still have the same rank after some previous ranking steps, then we assume that those with more leaves in the EPO-tree have greater chance to be real faults in CUD, because it is much easier to activate just one or two faults than many faults together to cause the same effect.*

An example in Figure 5.5 illustrates how the procedure works in this step. In this figure we can see that the number of branches in the EPO-tree of each fault is listed. The ranking procedure is kind of similar as before, which is performed in each group.

5.3 Final Fault Ranking

In previous sections, we have described how we rank each fault candidate in each branch of EPO-tree. In this section, we will discuss how we collectively handle the results from the analysis procedures. Fault ranking will finally give out a list of faults that are most suspicious among all fault candidates and those faults will be used as clues to locate faulty nets in next stage.

At this point, we need to introduce a very crucial concept in our ranking system, namely, *top-single-fault (TSF)*. A TSF is a single fault that has top rank in the branch. This kind of faults are considered top suspects. This is because we have already applied many constrains

in the branch ranking procedure and these faults have retained with highest potential to be the real ones compared to other faults in the branch. Second, a TSF is the only fault candidate in the top group of the branch, so there is no other fault has the same potential. Through experiments, we found that the TSFs have much higher chance to be the real faults in CUD than other faults.

Based on the reasons mentioned above, we first use the TSFs as the top candidates in our final fault list. It should be noted that, because we have applied many constraints in branch ranking, the earlier a TSF comes out, more suspicious it should seem. From Figure 5.1 we see that TSF check is performed after the forest of EPO-trees is built and after each analysis step in the ranking system. The new TSFs will be kept after each checking.

Sometimes TSFs alone cannot include all the real faults in the CUD. Then, we have to include more fault candidates than just TSFs in the final candidate list to increase diagnosability. In our system, the ranking of other candidates depends on the top-ranks they achieve in the branch ranking. For example, if a fault is detected in five branches and the top rank it gets in branch ranking among these five branches is one, which means it is in the top group in one branch, then we will use the top rank of this fault to compare with other faults.

An advantage of our ranking strategy is that a real fault candidate does not have to show its potential in every branch that it belongs to. We not only consider the overall matching performance of the candidates in branch ranking, we also consider their per-test performance in the final ranking. Every branch in each EPO-tree is unique. In multiple fault diagnosis, the interactions between faults are quite common. It is very possible that in some affected branches of certain injected fault, this fault may have a lower rank than other injected faults. If this situation occurs in most of this fault's affected branches and we consider the overall branch ranking performance, then we may give this fault a low rank. Therefore, for any fault candidate, no matter how many EPOs it interacted with or was masked by other candidates, as long as it can show a high potential even in one branch, it

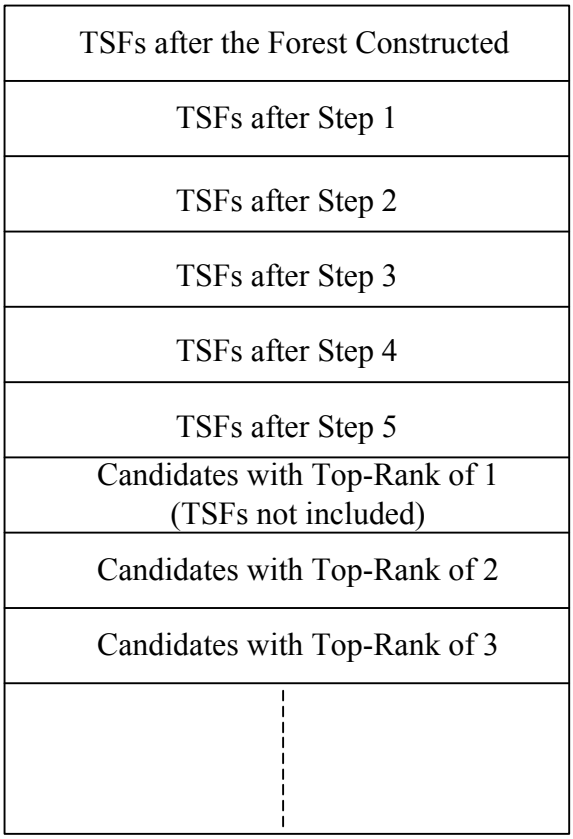

Figure 5.6: Arrangement of final candidate list.

is considered unique enough to be one of our final candidates. One more advantage that we have in comparison to previous per-test diagnosis works, which mainly rely on SLAT-based methods [27, 29], is that we are not constrained by the SLAT ability of test patterns.

The arrangement of final candidate list is shown in Figure 5.6. From the figure can we see that the list is levelized. In each level there may exist more than one fault candidate. So the next thing we need to do is to rank the fault candidates in each level in the list. The ranking procedure is similar to the one we use in steps two, three and five. We use the three counts from candidate reduction stages to rank the faults in each level in the list. One thing we need to mention is that this ranking is only performed in each level in the list, which means a candidate will not get out of the level it belongs to after the ranking.

Table 5.1: Diagnosis results for a single stuck-at fault.

Circuit	Diagnosability	FHR	Resolution	Time (s)
C2670	1.0	1.0	1.0	6.4
C3540	1.0	1.0	1.0	0.5
C6288	1.0	1.0	1.0	0.6
C7552	1.0	1.0	1.0	1.5

Table 5.2: Diagnosis results for two stuck-at faults.

Circuit	Diagnosability	FHR	Resolution	Time (s)
C2670	0.97	1.0	2.0	8
C3540	0.95	1.0	1.0	0.6
C6288	0.99	1.0	2.0	1
C7552	0.93	1.0	2.0	1.7

5.4 Experimental Results of Candidate Ranking

We have performed multiple stuck-at faults diagnosis experiments on four ISCAS85 benchmark circuits to examine the effectiveness of the fault reduction and ranking procedures of our diagnosis method. These experiments were performed on a Linux Server with Ubuntu 10.04 operating system consisting of 2*1GHz CPU with 2 MB cache, 2G RAM and 2G swap space. For each benchmark circuit, we randomly constructed 200 faulty circuit instances. For each faulty circuit we randomly injected 1 to 4 stuck-at faults. The average results of the diagnosis are shown in Tables 5.1, 5.2, 5.3 and 5.4. Diagnosability is defined as the percentage of the injected faults that could be identified. First hit rank (FHR) is defined as the first index of the injected fault in the ranked fault list. After diagnosis is completed and an ordered list of suspected faults is generated, next follows a physical examination of the faulty device. One naturally examines the sites of faults in the list staring at the top. That is the reason FHR is an important measure; ideally it should be 1. In the present experimental scenario it is a measure of how quickly we can locate an injected fault. Resolution is defined as the ratio of the number of reported fault candidates over the number of injected faults. The

Table 5.3: Diagnosis results for three stuck-at faults.

Circuit	Diagnosability	FHR	Resolution	Time (s)
C2670	0.94	1.0	2.0	13
C3540	0.94	1.0	2.0	0.7
C6288	0.95	1.0	2.0	2.7
C7552	0.90	1.04	2.0	2.9

Table 5.4: Diagnosis results for four stuck-at faults.

Circuit	Diagnosability	FHR	Resolution	Time (s)
C2670	0.89	1.06	2.0	17
C3540	0.91	1.0	2.0	1.8
C6288	0.92	1.02	2.0	5.7
C7552	0.88	1.1	2.0	3.2

ideal value of resolution is 1 and larger values mean poor resolution. From these experimental results we can see that our method has very good diagnostic resolution and first hit rank (FHR), which is very crucial in net diagnosis to be discussed in the next chapter. Because we will expand the collapsed fault set to an uncollapsed fault set, if we have too many fault candidates (i.e., poor resolution), then we will encounter difficulty in locating the faulty net.

Chapter 6

Diagnosis of Nets

In previous chapters, we have described procedures that reduce and rank fault candidates. From the ranking procedure, we can get a short list of fault candidates that may be related to the nets that physically contain failures. Finding and fixing physical defects or their causes is extremely important for the success of VLSI industry. Identification of faulty nets without tying the diagnosis to specific fault models is an effective way of using the failure data. In this chapter, we introduce a detailed method to diagnose faulty nets by using the fault information we obtained in previous chapters.

6.1 Candidate List Extension

In previous chapters we used collapsed faults in the fault reduction and ranking procedures. In ranking procedure, we further collapse the faults grouped on the basis of the test vector simulation data resulting in functional (or vector-specific) equivalence [38, 39]. Before we begin with the net diagnosis we expand the diagnosed fault list by uncollapsing. This is because each fault represents a physical site on some net. If we only considered the collapsed faults, the physical net information will remain incomplete.

In the first stage of fault extension, we first extend each fault candidate to a group of functionally equivalent faults. We then add the equivalent faults of each fault into the groups they belong to. As shown in Figure 6.1, each single fault in the initial candidate list is extended to a group of equivalent faults. It should be noted that, after the extension, the rank of each group of equivalent faults remains the same as that of the single fault that they are expanded from.

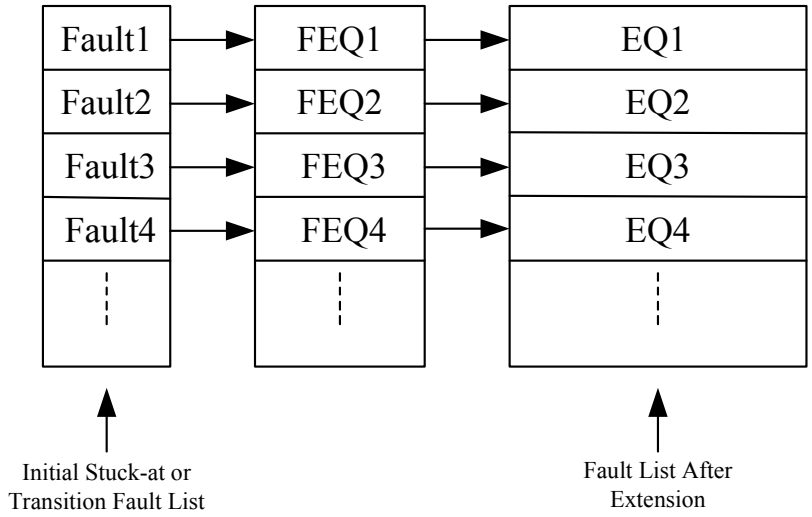

Figure 6.1: Candidate list extension.

6.2 Net Diagnosis

The target of our diagnosis is to locate possible faulty nets in the circuit. We utilize the extended fault list from last step to locate the suspicious nets.

From the netlist of the circuit, we can get the corresponding net for each fault. Then each group of equivalent faults can be transformed into a set of nets. As shown in Figure 6.2, similar to the fault extension procedure of the last section, each net set still has the same rank as the fault group that they were derived from.

After we transform the fault candidates to the net candidates, we do net ranking. First, we build a net pool, which includes all the net candidates of each rank group, if a net candidate is found in different rank groups, all of them will be included. The final net candidate list has two parts. The first part includes the nets for which we can find more than two members in the net pool. In other words, there are more than two fault candidates in the final fault list can locate the net. The second part includes the nets that can only be found once in the net pool. In both parts of the net ranking list, the rank of each net depends on the rank of the group that the net belongs to. If after this, there are still certain

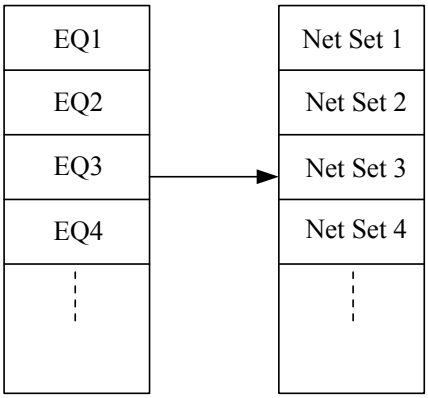

Figure 6.2: Faults to nets mapping.

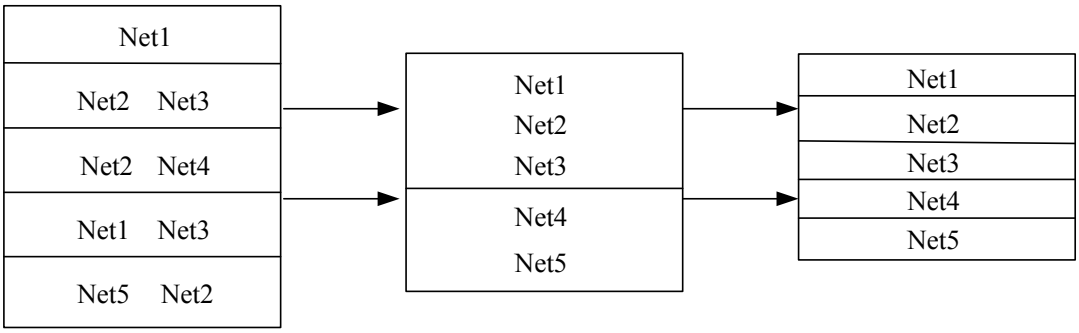

Figure 6.3: Net ranking.

nets that have the same rank, then their number in the net pool is used to break the tie. A case is shown in Figure 6.3. From the figure we note that Net1, Net2 and Net3 are found more than once in the initial list, so they are ranked in the first part of the candidate list as shown in the middle part in the figure. We also see that the occurrence of Net2 is more frequent than Net1 in the list, but because the top rank of Net1 is higher than Net2, Net1 is still ranked higher than Net2 in the final list as shown in the right part of the figure. The reason why we use such ranking strategy is because our previous work on diagnosing multiple faults produced very good FHR (first hit rank) and resolution and we must retain that advantage in net ranking.

Table 6.1: Diagnositic coverage of single stuck-at faults [52].

Circuit	# of Patterns	Diagnostic coverage
C432	69	100%
C880	60	100%
C1355	88	100%
C1908	135	98.78%
C2670	150	98.94%
C3540	170	97.17%
C6288	166	99.52%
C7552	296	99.35%

6.3 Experimental Results

The net fault diagnosis algorithm in this thesis was implemented using the Python programming language [45], including the scripts for fault injection, diagnosis and result analysis. We use a commercial fault simulator Fastscan [30] from Mentor Graphics as the simulation tool. All experiments used ISCAS85 benchmark circuits in order to examine the performance of the proposed diagnosis algorithms. In this section we will discuss the experimental results of the net diagnosis.

6.3.1 Diagnostic Patterns

In these experiments, we used test patterns that were generated for high diagnostic coverage of stuck-at faults [52] and that of transition faults [53]. The diagnostic coverage is defined as the ratio of number of distinguished equivalent fault groups to the number of total equivalence collapsed faults. High diagnostic coverage means that most of non-equivalent faults in the circuit can be distinguished. If two faults can be distinguished that means that the faults are not equivalent and certain test pattern produces different responses in presence of those faults. The diagnostic coverages for benchmark circuits by test patterns used are listed in Tables 6.1 and 6.2. From the tables we can see that most single faults in the circuit can be distinguished by the test patterns.

Table 6.2: Diagnositic coverage of single transition faults [53].

Circuit	# of Pattern	Diagnostic Coverage
C432	159	100%
C880	159	100%
C1355	305	100%
C1908	395	98.78%
C2670	251	98.94%
C3540	566	97.17%
C6288	205	99.52%
C7552	490	99.35%

Table 6.3: Fault dictionaries for stuck-at faults and vectors of Table 6.1.

Circuit	Constructing Time (s)	Data Size (MB)
C432	4.5	0.12
C880	6	0.17
C1355	12	0.56
C1908	22	1.5
C2670	32	1.7
C3540	35	2.5
C6288	81	5.7
C7552	140	10.2

6.3.2 Fault Dictionary Construction

Our dictionary stores the single fault simulation information in order to save the time by avoiding repeated fault simulation during diagnosis. Fault simulation is done without fault dropping. The dictionary stores all failing information about simulated faults, failing patterns and failing primary outputs. We use Fastscan as the fault simulation tool [30], which can simulate stuck-at and transition faults. The CPU time to construct the dictionaries and their sizes are listed in Tables 6.3 and 6.4. As before these experiments were also performed on a Linux Server with Ubuntu 10.04 operating system consisting of 2*1GHz CPU with 2 MB cache, 2G RAM and 2G swap space.

Table 6.4: Fault dictionaries for transition faults and vectors of Table 6.2.

Circuit	Constructing Time (s)	Data Size (MB)
C432	6	0.17
C880	20	1.4
C1355	32	1.2
C1908	45	3
C2670	84	1.8
C3540	120	5.1
C6288	328	11.1
C7552	510	14.3

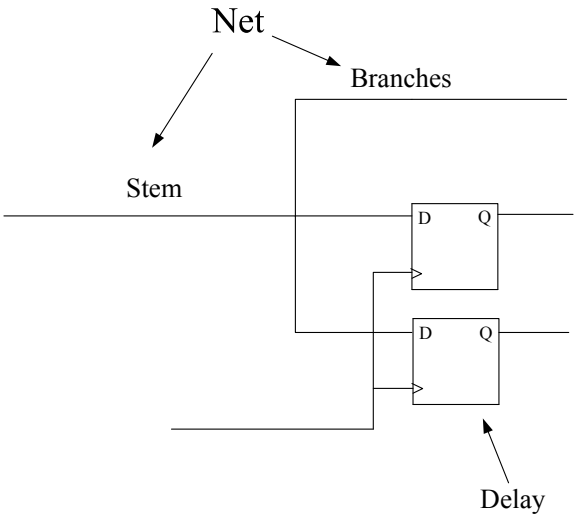

Figure 6.4: Transition fault injection.

6.3.3 Experimental Results of Net Diagnosis

We use eight benchmark circuits to conduct experiments. For each fault model and each benchmark circuit, we randomly constructed 20-50 faulty circuits for diagnosis. For each injected net fault, we randomly selected 2 to 4 fault-sites which could be either the stem or branches of the net. For stuck-at model, we injected one single stuck-at fault on each fault site and for transition fault model we injected a D-flip-flop on each fault site to represent the delay behavior as shown in Figure 6.4.

Table 6.5: Results of diagnosing single faulty net with stuck-at faults.

Circuit	Net Diagnosability	FHR	Net Resolution	Time (s)
C432	1.0	1.1	2.0	0.08
C880	1.0	1.1	2.0	0.1
C1355	0.98	2.0	3.0	1.06
C1908	0.92	1.4	3.0	7.5
C2670	0.98	1.4	3.0	7.4
C3540	0.98	1.04	3.0	0.5
C6288	0.98	1.1	3.0	3
C7552	0.96	1.1	3.0	3

Table 6.6: Results of diagnosing circuits with two faulty nets with stuck-at faults.

Circuit	Net Diagnosability	FHR	Net Resolution	Time (s)
C432	0.975	1.2	3.0	0.2
C880	1.0	1.1	3.0	0.15
C1355	0.75	2.6	5.0	16
C1908	0.86	1.3	3.0	13
C2670	0.95	1.14	4.0	12
C3540	0.96	1.1	3.0	0.7
C6288	0.9	1.06	3.0	4
C7552	0.9	1.2	3.0	11

The experimental results of net fault diagnosis with stuck-at fault model are listed in Tables 6.5 and 6.6. The results of net fault diagnosis with transition fault model are listed in Tables 6.7 and 6.8. Similar to the multiple stuck-at faults diagnosis in Chapter 5, the Diagnosability is defined as the percentage that the net with injected faults can be identified. First hit rank (FHR), defined as the index of the actual net with injected fault in ranked net list. It is a measure of how decisively we have found the real faulty net. Resolution is the ratio of the number of reported net candidates over the actual number of faulty nets. From the experimental result we can see that our diagnosis algorithm generally has very good diagnosability and resolution.

Table 6.7: Results of diagnosing single faulty net with transition faults.

Circuit	Net Diagnosability	FHR	Net Resolution	Time (s)
C432	1.0	1.05	2.0	0.48
C880	0.9	1.2	2.0	0.8
C1355	1.0	1.0	1.0	0.9
C1908	0.96	1.2	2.0	7.2
C2670	1.0	1.16	2.0	0.9
C3540	0.98	1.2	2.0	1.7
C6288	1.0	1.08	2.0	2.8
C7552	0.98	1.3	2.0	7.5

Table 6.8: Results of diagnosing circuits with two faulty nets with transition faults.

Circuit	Net Diagnosability	FHR	Net Resolution	Time (s)
C432	0.9	1.2	3.0	0.7
C880	0.8	1.3	3.0	2.3
C1355	0.86	1.3	3.0	13
C1908	0.93	1.4	3.0	35
C2670	0.97	1.06	2.0	4.6
C3540	0.95	1.1	2.0	2.3
C6288	0.92	1.3	2.0	2.8
C7552	0.96	1.08	2.0	13.1

Chapter 7

Conclusion

Traditional gate faults are closely related to the fault models and not necessarily to physical defects. Therefore, from a practical viewpoint it makes sense to diagnose a faulty net on a VLSI chip than to locate a "modeled" fault. Our use of stuck-at and transition faults models is for a practical reason, i.e., availability of tools for test generation and fault simulation. These models are used only for the possibility and simplicity of analysis they offer. In identifying faulty nets no assumption is made about the actual fault on them except that those nets 'may' have caused the observed and simulated errors. The fault models may, or may not be, used as suggestions. We verified against the injected multiple-faults.

The main ideas forwarded in this thesis are:

1. Because the number and variety of actually possible defects are enormous, almost unimaginable, diagnosis should focus on the physical location of defect and not on its type.

2. Once the faulty area of defect is located, the algorithms for finding the actual defect mechanism cannot be a purely analytical because that will require assumptions, leading back to fault models. That part of the procedure should be experimental or at best, interactive.

3. Fault models should be used without implication or assumption that they might represent actual defects. Their use is beneficial and should be encouraged because it permits analysis and use of tools; simplest models are the best.

In the future, arbitrary defects such as bridges, opens, short, etc., should be examined to evaluate the presented diagnosis algorithms.

Bibliography

[1] M. E. Amyeen, S. Venkataraman, A. Ojha, and S. Lee, "Evaluation of the Quality of N-Detect Scan ATPG Patterns on a Processor," in *Proc. International Test Conf.*, 2004, pp. 669–678.

[2] T. Bartenstein, D. Heaberlin, L. Huisman, and D. Sliwinski, "Diagnosing Combinational logic Designs using the Single Location At-a-Time (SLAT) Paradigm," in *Proc. International Test Conf.*, 2001, pp. 287–296.

[3] B. Benware, C. Schuermyer, S. Ranganathan, R. Madge, P. Krishnamurthy, N. Tamarapali, K. H. Tsai, and J. Rajski, "Impact of Multiple-Detect Test Patterns on Product Quality," in *Proc. International Test Conf.*, 2003, pp. 1031–1040.

[4] V. Boppana, R. Mukherjee, J. Jain, and M. Fujita, "Multiple Error Diagnosis Based on Xlists," in *Proc. Design Automation Conf.*, 1999, pp. 100–110.

[5] R. K. Brayton, *SIS: A System for Sequential Circuit Synthesis.* University of California, Berkeley, Tech. Report, 1992.

[6] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits.* Springer, 2000.

[7] K. De and A. Gunda, "Failure Analysis for Full-Scan Circuits," in *Proc. International Test Conf.*, 1995.

[8] A. L. D'Souza and M. S. Hsiao, "Error Diagnosis of Sequential Circuits Using Region-Based Model," in *Proc. 14th International Conf. on VLSI Design*, 2001, pp. 103–108.

[9] J. M. Galey, R. E. Norby, and J. P. Roth, "Techniques for the Diagnosis of Switching Circuit Failures," in *Proc. Second Annual Symp. on Switching Circuit Theory and Logical Design*, 1961, pp. 152–160.

[10] M. R. Grimaila, S. Lee, J. Dworak, K. M. Butler, B. Stewart, H. Balachandran, B. Houchins, V. Mathur, J. Park, L. Wang, and M. R. Mercer, "REDO: Random Excitation and Deterministic Observation," in *Proc. 17th IEEE VLSI Test Symp.*, 1999, pp. 268–274.

[11] Y. Higami and Y. Kurose, "Diagnostic Test Generation for Transition Faults Using a Stuck-at ATPG Tool," in *Proc. International Test Conf.*, 2009, pp. 1–9.

[12] S.-Y. Huang, "On Improving the Accuracy of Multiple Defect Diagnosis," in *Proc. 19th IEEE VLSI Test Symp.*, 2001, pp. 34–39.

[13] S.-Y. Huang, "Diagnosis of Byzantine Open-Segment Faults," in *Proc. 11th Asian Test Symp.*, 2002, pp. 248–253.

[14] S.-Y. Huang, "A Symbolic Inject-and-Evaluate Paradigm for Byzantine Fault Diagnosis," *Jour. Electronic Testing: Theory and Application (JETTA)*, vol. 19, pp. 161–172, 2003.

[15] S.-Y. Huang and K. T. Cheng, "Errormacer: design error diagnosis based on fault simulation techniques," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 18, pp. 1341–1352, 1999.

[16] Y. Huang, "On N-Detect Pattern Set Optimization," in *Proc. 7th International Symp. on Quality Electronic Design*, 2006, pp. 445–450.

[17] L. M. Huisman, "Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT)," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 23, no. 1, pp. 91–101, Jan. 2004.

[18] L. M. Huisman, *Data Mining and Diagnosing IC Fails*. Springer, 2005.

[19] K. R. Kantipudi, "Minimizing N-Detect Tests for Combinational Circuits," Master's thesis, Auburn University, Dept. of ECE, May 2007.

[20] K. R. Kantipudi and V. D. Agrawal, "On the Size and Generation of Minimal N Detection Tests," in *Proc. 19th International Conf. on VLSI Design*, 2006, pp. 425–430.

[21] K. R. Kantipudi and V. D. Agrawal, "A Reduced Complexity Algorithm for Minimizing N-Detect Tests," in *Proc. 20th International Conference on VLSI Design*, 2007, pp. 492–497.

[22] C.-Y. Kao, C.-H. Liao, and C. H.-P. Wen, "An ILP-Based Diagnosis Framework for Multiple Open-Segment Defects," in *Proc. 10th International Workshop on Microprocessor Test and Verification*, 2009, pp. 69–72.

[23] C.-Y. Kao, C.-H. Liao, and C. H.-P. Wen, "Diagnosing Multiple Open-Segment Defects Using Integer Linear Programming," vol. 27, no. 6, Dec. 2011.

[24] S. Kundu, S. Chattopadhyay, I. Sengupta, and R. Kapur, "Multiple Fault Diagnosis Based on Multiple Fault Simulation Using Particle Swarm Optimization," in *Proc. 24th International Conf. on VLSI Design*, 2011, pp. 364–369.

[25] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, July 1982.

[26] D. B. Lavo, *Comprehensive Fault Diagnosis of Combinational Circuits*. PhD thesis, University of California, Santa Cruz, Dept. of Computer Engineering, 2002.

[27] D. B. Lavo, I. Hartanto, and T. Larrabee, "Multiplets, Model, and the Search for Meaning: Improving Per-Test Fault Diagnosis," in *Proc. International Test Conf.*, 2002, pp. 250–259.

[28] D. B. Lavo, T. Larabee, and B. Chess, "Beyond the Byzantine Generals: Unexpected Behavior and Bridging Fault Diagnosis," in *Proc. International Test Conf.*, 1996, pp. 611–619.

[29] C. Liu, "An Efficient Method for Improving the Quality of Per-Test Fault Diagnosis," in *Proc. International Conference on Computer-Aided Design*, 2004, pp. 648–651.

[30] Mentor Graphics, *FastScan and FlexTest Reference Manual*, 2004.

[31] S. D. Millman and J. P. Garvey, "An Accurate Bridging Fault Test Pattern Generator," in *Proc. International Test Conf.*, 1991, pp. 411–418.

[32] D. Nayak and D. M. H. Walker, "Simulation-Based Design Error Diagnosis and Correction in Combinational Digital Circuits," in *Proc. 17th IEEE VLSI Test Symp.*, 1999, pp. 70–78.

[33] I. Pomeranz and S. M. Reddy, "Forming N Detection Test Sets from One-Detection Test Sets Without Test Generation," in *Proc. International Test Conf.*, 2005, pp. 527–535.

[34] S. M. Reddy, I. Pomeranz, H. Tang, S. Kajihara, and K. Kinoshita, "On Testing of Interconnect Open Defects in Combinational Logic Circuits With Stems of Large Fanout," in *Proc. International Test Conf.*, 2002, pp. 83–89.

[35] E. M. Rudnick, W. K. Fuchs, and J. H. Patel, "Diagnostic Fault Simulation of Sequential Circuits," in *Proc. International Test Conf.*, 1992, pp. 178–186.

[36] P. G. Ryan, S. Rawat, and W. K. Fuchs, "Two-Stage Fault Location," in *Proc. International Test Conf.*, 1991, pp. 963–968.

[37] J. Segura and C. F. Hawkins, *CMOS Electronics: How It Works, How It Fails.* Wiley-IEEE, 2004.

[38] M. A. Shukoor, "Fault Detection and Diagnostic Test Set Minimization," Master's thesis, Auburn University, Dept. of ECE, May 2009.

[39] M. A. Shukoor and V. D. Agrawal, "A Two Phase Approach for Minimal Diagnostic Test Set Generation," in *Proc. 14th IEEE European Test Symp.*, May 2009, pp. 115–120.

[40] N. Sridhar and M. S. Hsiao, "On Efficient Error Diagnosis of Digital Circuits," in *Proc. International Test Conf.*, 2001, pp. 678–687.

[41] C. E. Stroud, *A Designer's Guide to Builr-In Self-Test.* Springer, 2002.

[42] 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 Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 21, no. 3, pp. 362–368, 2002.

[43] E. N. Tran, V. Kasulasrinivas, and S. Chakravarty, "Silicon Evaluation of Logic Proximity Bridge Patterns," in *Proc. 24th IEEE VLSI Test Symp.*, 2006, pp. 78–83.

[44] R. Ubar and D. Borrione, "Generation of Tests for the Localization of Single Gate Design Errors in Combinational Circuits Using the Stuck-at Fault Model," in *Proc. XI Brazilian Symposium on Integrated Circuit Design*, 1998, pp. 51–54.

[45] G. van Rossum and J. F. L. Drake, *Python Tutorial Release 2.6.3.* Python Software Foundation, Oct. 2009. docs@python.org.

[46] S. Venkataraman and S. B. Drummonds, "A Technique for Logic Fault Diagnosis of Interconnect Open Defects," in *Proc. 18th IEEE VLSI Test Symp.*, 2000, pp. 313–318.

[47] S. Venkataraman and S. B. Drummonds, "Poirot: Applications of a Logic Fault Diagnosis Tool," *IEEE Design & Test of Computers*, vol. 18, no. 1, pp. 19–29, Jan. 2001.

[48] S. Venkataraman, S. Sivaraj, E. Amyeen, S. Lee, A. Ojha, and R. Guo, "An Experimental Study of N-Detect Scan ATPG Patterns on a Processor," in *Proc. 22nd IEEE VLSI Test Symp.*, 2004, pp. 23–29.

[49] J. Waicukauski and E. Lindbloom, "Failure Diagnosis of Structured VLSI," *IEEE Design & Test of Computers*, vol. 6, pp. 49–60, Aug. 1989.

[50] Z. Wang, M. Marek-Sadowska, and J. Rajski, "Analysis and Methodology for Multiple-Fault Diagnosis," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 558–576, Mar. 2006.

[51] Z. Wang, K. H. Tsai, M. Marek-Sadowska, and J. Rajski, "An Efficient and Effective Methodology on the Multiple Fault Diagnosis," in *Proc. International Test Conf.*, 2003, pp. 329–338.

[52] Y. Zhang and V. D. Agrawal, "A Diagnostic Test Generation System," in *Proc. International Test Conf*, 2010. Paper 12.3.

[53] Y. Zhang and V. D. Agrawal, "Reduced Complexity Test Generation Algorithms for Transition Fault Diagnosis," in *Proc. International Conf. on Computer Design*, Oct. 2011, pp. 96–101.