Delft University of Technology
Master's Thesis in Computer Science

ANTARES: Automatic Diagnosis of Software/Hardware Systems

Shekhar Gupta

embedded software

TU Delft Delft University of Technology

ANTARES: Automatic Diagnosis of Software/Hardware Systems

Master's Thesis in Computer Science

Embedded Software Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Shekhar Gupta
S.Gupta-3@student.tudelft.nl

13th May 2011

Author
Shekhar Gupta (S.Gupta-3@student.tudelft.nl)
Title
ANTARES: Automatic Diagnosis of Software/Hardware Systems
MSc presentation
May 19, 2011

Graduation Committee

Prof. Dr. Ir. Arjan J.C. Van Gemund (chair)	Delft University of Technology
Prof. Dr. C. Witteveen	Delft University of Technology
Dr. Ir. Rui Abreu	University of Porto
Dr. Ir. Stefan Dulman	Delft University of Technology

Abstract

With the market becoming increasingly competitive, there is a pressure to deliver systems with more functionality and at the same cost, which thus leads to more complexity in terms of number of components. Moreover, the society is becoming increasingly dependent on these systems for its critical functions. This coupled with shrinking time-to-market and reducing life-cycle, creates a need to find ways to ensure reliability of these complex systems both efficiently and quickly. Due to large size and complexity of modern day systems, fault-finding problem is a non-trivial one. Traditionally, Model-Based Diagnosis (MBD) is used to locate faults in the hardware. A prerequisite for MBD is the accurate model of the components. However, modeling of such complex components requires huge effort, time and expertise. Earlier, a spectrum-based hardware solution named BACINOL was proposed to diagnose the hardware system without the aid of a component model. But BACINOL suffers from low diagnosis quality due to large size of ambiguity sets in the final diagnosis. In this thesis, we introduce a new spectrum-based hardware diagnosis technique ANTARES. It attempts to break these ambiguity sets by providing a better estimate of system's False Negative Rate (FNR) information to the diagnosis method. A series of experiments are performed on the ISCAS benchmark circuits to compare the performance of ANTARES with BACINOL and MBD. Results clearly show that ANTARES has better diagnosis quality as compared to BACINOL but has lower performance than MBD.

Preface

It is the journey to success that is more gratifying than the success itself. I began this incredible journey the moment I decided to join TU Delft and since then, it has been one of the most challenging and interesting one. The last 9 months while I was working on the thesis have been the most exciting part of this journey and I am really happy that I have successfully accomplished the task I began.

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Arjan J.C. Van Gemund for his immense help, guidance and motivation throughout my thesis. He has always been there for me whenever I had doubts, either in my thesis or in my personal life. I must admit that without him, it would have been very difficult to finish this assignment with such an excellence.

I would like to extend my special thanks to Rui Abreu for all the BARINEL related assistance he provided. I would also like to extend my thanks to Paulo Anita for providing his IT services with a great smile (always). I would like to thank Apurva Dargar for helping me review my thesis. Thanks to all the guys in my apartment with whom I spent an awesome year, specially for the late night *chai* (tea) sessions. Thanks to all my friends in Delft who made my master's journey enjoyable and memorable.

I would specially like to thank my sister Shikha who always inspired me to make the most of my life and my niece Anjani for making me smile whenever needed. This journey wouldn't have been possible without the efforts of my family, my father and mother back in India. Although they are not here in Delft, but their support and guidance throughout is what motivated me all along and I know, today they are very proud of me.

Shekhar Gupta

Delft, The Netherlands

13th May 2011

Contents

Chapter 1

Introduction

In the modern world, technological systems and devices such as television, mobile phones and satellites are becoming ubiquitous. Each of these systems provides a wide range of functionality while constituting a varying numbers of components. Every component has a different level of complexity and a fault in one of them might lead to system failure. As the dependency of society on these systems increases, their reliability becomes increasingly critical. Failure of the system can prove to be expensive for every second that it is not functional. For example, a fault in an analog to digital converter (ADC) can cause a failure in the lithographic tool that is used in the production of integrated circuits, potentially resulting in a huge production loss. Therefore, it is important to locate and fix faults in such kinds of systems as soon as possible. Due to the large complexity of any modern day system, it is impossible to perform manual fault diagnosis. Automated or computer-aided diagnosis techniques are emerging as an important solution to localize faults that are the root causes of system failures. For the last several years, diagnosticians have been trying to find highly accurate, low-cost and fully-automated solutions to diagnose faults in systems. The two predominant approaches for automated diagnosis are model-based diagnosis and spectrum-based fault localization.

1.1 Automated Fault Diagnosis

Model-based diagnosis (MBD) uses a descriptive model of the internal structure and behavior of the system to perform diagnosis. A model of the system is defined as a detailed behavior of components (behavioral model) and a description of how those components are connected to each other (structural model). The system behavior is measured in terms of observations. The basic principle of MBD can be understood as the analysis of inconsistency

between the expected value and the observed value of the system output. Deriving explanations for the inconsistencies is defined as diagnosis.

MBD has shown very good performance in locating faults in hardware at the expense of a huge cost. The cost can be divided into two parts: modeling cost and solution cost. Solution cost includes algorithmic cost and identification cost (time taken by diagnostician to find the fault). Traditionally, reduction of solution cost has been the main topic of interest among the diagnosis community.

A recent case study in the Dutch industry suggests that modeling cost is the bottleneck for the acceptance of MBD in the industry. At ASML, a LYDIA-based diagnostic engine was installed on over 3,000 service laptops world-wide, and has been successfully used for many years. It has been shown that the solution cost of MBD can be reduced from days to minutes by investing 25 man-days in the modeling process (approximately 2,000 LOC, comprising sensor modeling, electric circuits and some simple mechanics) [18]. Despite such a reduction in the solution cost, management discontinued the project once it became clear that only 80% of the model (LYDIA code) could be obtained automatically from the source code (graphical schematics data for electrical, VHDL for the logic circuits). MBD was rejected because of the fact that if a hardware component is upgraded or changed, the modeling process will have to be repeated leading to a huge manual effort as modeling is not completely automated.

On the other hand, the software engineering community also has problems with such detailed modeling. Due to the large size of software, modeling cannot be done efficiently. Therefore, a spectrum-based fault localization (SFL) approach has been used for software diagnosis. In SFL, the dynamic program execution profile of tests (program spectra) is combined with test outcomes (pass/fail) to compute the diagnosis. SFL does not require detailed modeling of the program block (statements); rather it uses a very generic model to perform diagnosis.

Because MBD has high modeling cost unlike SFL, it becomes interesting to adopt SFL for hardware diagnosis. BACINOL, an SFL approach for hardware diagnosis has been proposed recently. Exploiting only a structural model of the system to diagnose the fault. However, it suffers from low diagnosis quality as compared to MBD because SFL uses a very generic model of the components.

1.2 Problem Statement

From the previous section, it is clear that constructing models of hardware components for MBD is not preferable. Therefore, BACINOL was proposed

to perform hardware diagnosis without using the behavioral model of components. BACINOL uses a very generic model of the components and therefore has very low diagnosis accuracy as compared to MBD. This is caused by the fact that BACINOL generates a ranking-list (components ordered by probability of being at fault) where many components can have the same rank (probability); this set of components is defined as an ambiguity set. The large size of the ambiguity set lowers the diagnosis quality of BACINOL.

If we can break the ambiguity set present in the diagnosis produced by BACINOL, diagnosis quality of BACINOL can be increased. Ambiguity sets can be broken by more accurate Bayesian probability computation underlying SFL. In this thesis, we study the impact of exploiting False Negative Rate (FNR) information of a single or multiple fault candidates to improve that Bayesian probability computation. This FNR information can be obtained by computing Error Propagation Probability (EPP) behavior of the system. Our problem statement can now be defined to:

How can EPP of a system for single and multiple faults be computed using only the structural model of the system? How feasible is it to combine the FNR information derived from system's EPP with the existing spectrum-based hardware diagnosis (BACINOL)? And, in particular, what is the impact of including FNR information on the diagnosis quality?

1.3 Thesis Contribution

To solve the problem statement mentioned in the previous section, this thesis makes following contributions:

- In this thesis, we propose a new spectrum-based hardware diagnosis technique, ANTARES (Automatic diagNosis of sofTware/hardwARE Systems), which computes a diagnosis based on the known structural model of a system without using the behavioral model of the components involved and uses the system's FNR (externally computed) to enhance diagnosis quality.
- We propose three different EPP models to automatically estimate the FNR parameter (g) of every components in the hardware. The EPP is a critical feature of ANTARES that significantly increases the performance of SFL based hardware technique by eliminating the ambiguity sets in the final diagnosis.
- Traditionally, OR model has been used in SFL-based methods to compute the multiple-fault g values, which is theoretically not a very accurate model. We propose two models to estimate the multiple-fault g values: MIN model and Level-based model.

- A statistical approach is used to compute the quality of diagnosis techniques. We derive the diagnostic quality computation models that consider the presence of ambiguity sets in ranking list using this statistical approach.
- A series of experiments have been performed to assess the diagnosis quality of ANTARES with different EPP models. Performance of ANTARES is also compared to that of BACINOL, as well as to GDE, a state-of-the art MBD engine. All the experiments have been carried out on ISCAS/74XXX benchmark circuits.

Experimental results clearly indicate that the external g information given to ANTARES enhances the performance of ANTARES as compared to BACINOL. The most accurate FNR estimation gives the maximum improvement in the diagnosis quality.

1.4 Thesis Outline

Rest of the thesis is organized as follows. Chapter 2 introduces the main concepts and notations used in fault diagnosis. Chapter 3 introduces the statistical models to compute the diagnosis quality; these models are then used to address the problem of the ambiguity sets and a solution is also proposed at the end the chapter. In Chapter 4, we derive three EPP models by considering the FNR of single and multiple fault candidates. At the end of the chapter accuracy results of derived EPP models are shown. In Chapter 5, we present the ANTARES approach to diagnose hardware. The derived EPP models are used to enhance the diagnosis quality. Chapter 6 presents experimental results, along with discussions on the observations made on the results. Finally, Chapter 7 concludes the thesis with an overview of the results of ANTARES and addresses further research opportunities.

Chapter 2

Fault Diagnosis

This chapter gives an introduction to the terminology developed and used in the field of Fault Diagnosis. In this chapter, we present the assumptions underlying approaches to diagnostic problem solving, the theory behind Model Based Diagnosis (MBD) [19], and the details of Spectrum-based Fault Localization (SFL) technique [1]. Theoretical concepts and tools used in MBD and SFL are explained. Finally, we present a previous attempt to use SFL for hardware (BACINOL) [20].

2.1 Terminology

A *system* is composed of interacting components which are connected to each other in a certain manner. A system is said to *fail* when the observed behavior of the system is different from its expected behavior. This deviation in the behavior of system is defined as an *error*.

For example, if we turn the ignition key of car and nothing happens, this means that there is some fault in the system (car) due to which it has failed. This failure in the system is caused by the components at fault, which in this case can be the starting engine, the battery or any other component in that car, any of which might be working incorrectly. Therefore, a fault can be defined as an unintended difference between the implemented component and its intended design.

For a system to work flawlessly, fault diagnosis software can be implemented to continuously monitor the system such as a car, a robot, a satellite, a nuclear plant. This monitoring software finds the root cause of the system malfunction. Primary objective of fault diagnosis is to determine the location and the occurrence time of faults based on *accessible data* and knowledge about the behavior of the system. Input and output data are classified as the accessible data for any system; these accessible data are defined as

observations. Behavior of the system gives the information about correct output of the system. Therefore, it is important to know the behavior of the system to determine whether the system exhibits any error or not.

To understand the diagnosis problem more closely, consider a system shown in Figure 2.1. Component $c_1, \ldots, c_5$ are possible fault locations. Error can be observed at output nodes e and f. Inputs (a,b,c) and outputs (e,f) are the observations for the system. Different input vectors are given to the system and output vectors obtained are compared with the correct output (estimated from the behavior of the system). Any difference in the actual output and the correct output indicates an error. This error in turn indicates fault in the system.

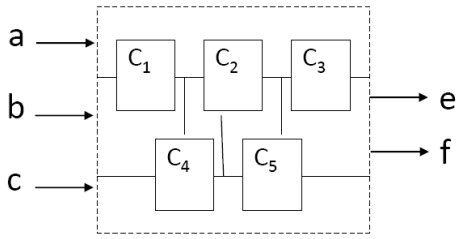

Figure 2.1: A general system consisting of five components

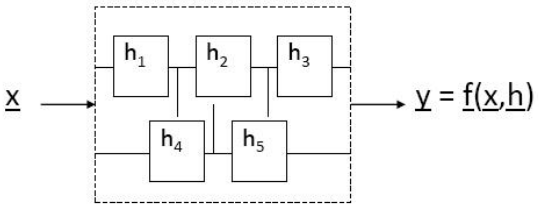

Figure 2.2: Conceptual view of system

Generally, we consider that a system consists of n components, where behavior is presented by system function f [21]. It applies that system functions according to $y = f(x, h)$, where x and y represent the system input and output, respectively, and where $h = (h_1, ..., h_n)$ indicates the health state of the n components (see Figure 2.2). h is defined as the health state of a component. $h = 1$ indicates a healthy component (not faulty) while $h = 0$ indicates one which is faulty. From this proposition we can define the output of the system to be a function of input and health state of each of its components. Hence mathematically, diagnosis can be understood as solving

the inverse problem $h = f^{-1}(x, y)$, i.e., find the combinations of component health states that explains the observed output for a given input.

Having discussed the basics of diagnosis, we now explain the various methods which are used to diagnose a system. Two classical methodologies are used to diagnose the faults in hardware and software. First method is *Model-based Diagnosis*. This diagnosis technique is originally developed for hardware. Second method is *Spectrum-based Fault Localization* technique. This method is fundamentally developed to detect faults in software. The following section describes the concepts behind Model-based Diagnosis.

2.2 Model Based Diagnosis

Model-based Diagnosis (MBD) refers to reasoning from first principles,that declaratively describes a system's structural and behavioral properties [19]. In simple words, MBD uses a system's structural and behavioral model, together with observations to locate faulty components. The structural model gives information about a component's connectivity. In recent years, MBD has proven successful in the field of fault localization in hardware.

MBD focuses on the logical relations between the components of a complex system. Therefore, the function of each component and the interconnections between components are all presented as a logical system called the *System Description* (SD). The expected behavior of the system is then a logical consequence of its SD. Thus the existence of faulty components leads to an inconsistency between the observed behavior of the system and SD. Therefore, the determination of faulty components (diagnosis) is reduced to finding those components whose abnormality can explain all the inconsistencies.

A classical example of MBD approach is the General Diagnostic Engine (GDE) proposed in [10]. The following section explains this methodology in more detail:

2.2.1 MBD methodology

Given a model of a system and some inputs, MBD predicts the expected values and compares them with the real (measured) values to identify faulty components. Following are the key terms originating from MBD:

- A *symptom* is any difference between a predicted value and a measured value.
- A *conflict* is a set of assumptions (components that may be faulty) that supports a system, and thus leads to an inconsistency.

- A *candidate* is a hypothesis for how the actual system differs from the model of that system.

To understand the basic principles of MBD, consider the three inverter circuit shown in Figure 2.3. The circuit has one input (x) and two outputs (y_1, y_2). The system contains 3 inverters as components. Each of which has an input a and output b (see Figure 2.4). The healthy, nominal behavior of an inverter is that the output is equal to the inverted input (Equation 2.1).

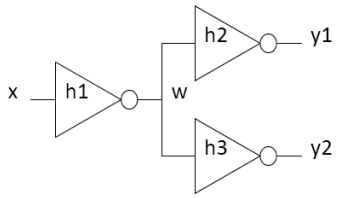

Figure 2.3: Three inverter circuit

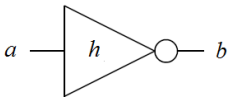

Figure 2.4: Inverter

$$h \Rightarrow (b \Leftrightarrow \neg a) \tag{2.1}$$

A model for the complete circuit can be developed by composition of single model. For three inverter circuit, following model equations can be derived by composition. These equations are defined as

$$\begin{aligned} h_1 &\Rightarrow (w \Leftrightarrow \neg x) \\ h_2 &\Rightarrow (y_1 \Leftrightarrow \neg w) \\ h_3 &\Rightarrow (y_2 \Leftrightarrow \neg w) \end{aligned}$$

Based on the observations, GDE generates a Dependency List (DL) [12] of the components that support the observed value. Suppose our observation is $(x, y_1, y_2) = (1, 0, 1)$. Output $y_1 = 0$ shows the system failure. Analyzing these system description equations with the given observation results in the following propositions:

$$\begin{aligned} h_1 &\Rightarrow \neg w \\ h_2 &\Rightarrow w \\ h_3 &\Rightarrow \neg w \end{aligned}$$

which equals:

$$(\neg h_1 \vee \neg w) \wedge (\neg h_2 \vee w) \wedge (\neg h_3 \vee \neg w)$$

Resolution yields the following conjunction of conflicts,

$$(\neg h_1 \vee \neg h_2) \wedge (\neg h_2 \vee \neg h_3) \tag{2.2}$$

Generation of above conflicts can be understood by the following propositions:

- Output $y_1 = 0$ produces a conflict. The path from primary input (x) to primary output (y_1) consists of components c_1 and c_2 are involved, hence dependency list $DL_1 =< c_1, c_2 >$.
- A conflict also arises when components with same SD produces different output. In the given circuit inverters c_2 and c_3 inspite of having the same input W produce different output, 0 and 1 respectively. Hence dependency list $DL_2 =< c_2, c_3 >$.

Conflicts generated by GDE (Equation 2.2) can be processed by BARINEL to get the multiple-fault diagnosis. BARINEL [1] is a spectrum-based approach to generate the diagnosis. That accepts conflicts as a spectrum matrix; therefore it is necessary to express these conflicts in the form of spectrum matrix. Spectrum matrix (A, e) representation is an SFL [1] way to represent the conflicts. Circuit spectra A has N rows and M columns, where N is the number of dependency lists and M is the number of components in the system. The $(i, j)^{th}$ element of the matrix is denoted by a_{ij}, we have $a_{ij} = 1$, if component c_j belongs to dependency list DL_i. Pass ('+')/fail('-') information for each of the dependency lists is stored in the error vector e. Since GDE only generates the conflicts (failed cases), all the entries in e are '-'. For the 3 inverter circuit example shown in Figure 2.3, conflicts are as presented in Table 2.1:

c1	c2	c3	e
1	0	1	-
0	1	1	-

Table 2.1: Matrix representation of MBD conflicts

We have encoded dependency list into spectrum matrix, and from now on we will use this spectrum matrix (A, e) to perform the rest of the diagnosis steps.

2.2.2 Candidate Generation

Components in the dependency lists (conflict sets) are combined to obtain diagnosis candidates by computing Minimal Hitting Sets (MHS), i.e., minimal diagnosis candidates that cover all dependency lists. Following section describes MHS algorithm

Minimal Hitting Set

Normally speaking, it is a problem of selecting a minimal set that has a non-empty intersection with each set. This is a formulation of the minimal hitting set problem, which, in general, is NP-hard. MHS problem can be defined as:

Definition 1. *Let S be a collection of N non-empty sets $S = s_1, s_2, ..., s_N$. Each set $s_i \in S$ is a finite set of components element, where each of the M elements is represented by a number $j \in 1, ..., M$. A minimal hitting set of S is a set d such that:*

$$\forall s_i \epsilon S, s_i \cap d \neq \phi \wedge \nexists d' \subset : s_i \cap d' \neq \phi$$

In words, each member of S has at least one component of d as a member, and no proper subset of d is a hitting set.

The set S is encoded into spectrum matrix. Element $s_i \in S$ will be generated by each conflict row of the spectrum matrix. In a conflict row, if a component is member of set s_i then element a_{ij} is equal to 1. Consider the spectrum matrix shown in Table 2.1. For the first conflict row, $s_1 =< c_1, c_2 >$ and for the second row $s_2 =< c_2, c_3 >$. This gives us the MHS for set S as $d =< \{c_2\}, \{c_1, c_3\} >$. So we have two MHS $d_1 =< c_2 >$ and $d_2 =< c_1, c_3 >$

For this example, diagnosis candidate could be calculated easily because of the small size of the problem. However, for larger systems calculation of diagnosis candidates is not trivial because of exponential complexity of MHS algorithm. Abreu develops a low cost, statistic based technique, named STACCATO [3] to obtain MHS.

STACCATO

The key idea behind STACCATO is the fact, that the components that are members of more sets as compared to other components may be an indication that there is a MHS containing such component. The trivial case is those components that are involved in all the sets, which constitute a minimal hitting set of cardinality equal to 1. From the last section, $s_1 =< c_1, c_2 >$ and $s_2 =< c_2, c_3 >$, $< c_2 >$ is member of both the sets. Hence c_2 will be

a member of the MHS with cardinality equal to 1. For each candidate, a probability is estimated by using information produced by spectrum (A, e). These probabilities are used to ensure that all significant probability mass is captured by generated candidates; this reduces the computation complexity significantly.

Bayesian approach [6] is applied to deduce the multiple-fault diagnosis D. The D is an ordered set of all minimal diagnosis candidates, ordered by decreasing posterior probability. Following section illustrates this Bayesian approach.

2.2.3 Candidate Ranking

BARINEL estimates $Pr(d_k)$ for each of the diagnosis candidate d_k and ranks those candidates according to the probabilities. $Pr(d_k)$ is the probability that candidate d_k is responsible for the faulty behavior of the system. For each candidate, this probability depends on the extent to which it explains the observed behavior. Bayes' rule can be used to estimate the posterior probability that d_k is the true diagnosis given observation obs_i.

$$Pr(d_k \mid obs_i) = \frac{Pr(obs_i \mid d_k)}{Pr(obs_i)}.Pr(d_k \mid obs_{i-1})$$

The denominator $Pr(obs_i)$ is a normalizing term that is identical for all d_ks and does not require any direct computation. $Pr(d_k \mid obs_{i-1})$ is the prior probability of d_k. In the absence of any observations, $Pr(d_k \mid obs_{i-1})$ defaults to $Pr(d_k) = p^{|d_k|}(1-p)^{M-|d_k|}$, where p denotes the prior probability that the component c_j is at fault. By default we set $p_j = p.Pr(obs_i \mid d_k)$, which is defined as:

$$Pr(obs_i \mid d_k) = \begin{cases} 0 & \text{if } obs_i \wedge d_k \models \bot \\ 1 & \text{if } d_k \rightarrow obs_i \\ \epsilon & \text{otherwise} \end{cases}$$

The above clauses can be understood as following. If the diagnosis candidate d_k cannot explain the observed output at all, then that candidate has no role behind the erroneous output, hence posterior probability is 0. If only d_k can uniquely explain the observation then posterior probability will be 1. Many ϵ policies have been proposed to estimate $Pr(obs_i \mid d_k)$ [15]. In BARINEL, Abreu adopts the following ϵ policy [7]:

$$\epsilon = \begin{cases} g(d_k) & \text{if } e_i = + \\ 1 - g(d_k) & \text{if } e_i = - \end{cases} \tag{2.3}$$

In the Equation 2.3 term $g(d_k)$ is used rather than the individual component False Negative Rate (FNR) parameter g_c. FNR expresses the probability that a test that covers faulted component c does not capture an error at the output of the system. $g(d_k)$ is defined as an FNR parameter for the candidate d_k. $g(d_k)$ is used to represent the probability that system will correctly although components involved in d_k are faulty. $g(d_k)$ is estimated according to following model:

$$g(d_k) = \prod_{c \epsilon d_k} g_c \tag{2.4}$$

Equation 2.4 is defined as OR model [5], which derives from the fact that probability that error is not observed at the output is product of the probability that each of involved faulty component exhibits correct behavior. The OR model determines the 'g' value corresponding to the multiple-fault candidate $g(d_k)$ from single-fault 'g' values.

Spectrum matrix shown in Table 2.1 is processed by BARINEL and produces following ranking list shown in Table 2.2:

Candidate (d_k)	Probability ($Pr(d_k)$)
c_2	0.99
c_1,c_3	0.01

Table 2.2: Ranking generated by BARINEL for MBD conflicts

BARINEL generates a diagnostic report $D =< d_1, \ldots, d_{|D|} >$. Each of the candidates in the diagnostic report consists of one or more components and one component can be part of multiple candidates. Such kind of diagnosis is converted in 1-dimensional diagnostic report $R =< c_1, \ldots, c_{|M|} >$. This report is an ordered set of all the components, where components are ranked in increasing order of their fault probability. For 1-D ranking, instead of using $Pr(d_k)$, we use $Pr(c_j)$ as posterior probability. $Pr(c_j)$ is defined as the probability that component c_j is at fault. Table 2.3 shows the 1D mapping of ranking list shown in Table 2.2.

Candidate (c_j)	Probability ($Pr(c_j)$)
c_2	0.99
c_1	0.21
c_3	0.21

Table 2.3: 1-dimensional ranking for MBD conflicts

2.3 Spectrum Based Fault Localization

SFL is one of the most promising approaches towards fault localization in software. It has received a lot of attention for large software due to its simplicity and effectiveness [1]. Two essential kinds of information are collected for SFL, namely *program spectrum* and *error vector* (Spectrum matrix discussed in Section 2.2.1).

In MBD (Section 2.2.1) we construct circuit spectrum by data flow from primary input to primary output and logical reasoning as well. In SFL, a program spectrum is a collection of data that provides a specific view on the dynamic behavior of the software [1]. Generally speaking, it records the run-time profiles about various program blocks for a specific input. In MBD, these blocks are hardware components but in SFL we need to define the scope of every block. The blocks could be statements, branches, paths or basic blocks, etc. Consider the program P shown in Figure 2.5. Program is divided into four different blocks. Program spectrum shows for each of the input which entity was involved and which entity was not.

```
void RationalSort(int n, int *num, int *den) {
    /* block 1 */
    int i, j, temp;

    for(i = n-1; i >= 0; i--) {
        /* block 2 */
        for(j = 0; j < i; j++) {
            /* block 3 */
            if(RationalGT(num[j], den[j], num[j + 1], den[j + 1])) {
                /* block 4 */
                /* Bug: forgot to swap denominators */
                temp = num[j];
                num[j] = num[j+1];
                num[j+1] = temp;
            }
        }
    }
}
```

Figure 2.5: Faulty C Function

Apart from the program spectrum, the output associated with each input vector is also essential to SFL. It records whether a test case has failed or passed. Together with the information about involvement of blocks, the testing results give debuggers hints about the blocks which are more likely related to failure, and hence have higher possibility to contain the faults. This pass/fail information is defined as an error vector.

Given a program P with M blocks and executed by N input vectors, Figure 2.6 shows the essential information required by SFL.

$$
\begin{array}{c} \\ N \text{ spectra} \end{array}
\begin{array}{c} \overset{M \text{ parts}}{} \\ \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{pmatrix} \end{array}
\quad
\begin{array}{c} \text{errors} \\ \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{pmatrix} \end{array}
$$

Figure 2.6: Spectrum matrix representation

Input	Block					Error
	1	2	3	4	5	
$I_1 = \langle \rangle$	1	0	0	0	0	+
$I_2 = \langle ¾ \rangle$	1	1	0	0	0	+
$I_3 = \langle ⅞, ⅛ \rangle$	1	1	1	1	1	+
$I_4 = \langle ⅛, ⅞, ⅛ \rangle$	1	1	1	1	1	+
$I_5 = \langle ⅜, ⅝, ⅘, ¼ \rangle$	1	1	1	1	1	-
$I_6 = \langle ¼, ⅓, ½, ⅛ \rangle$	1	1	1	0	1	+

Figure 2.7: Spectrum matrix for faulty C program

If we recall the spectrum description presented in Section 2.2.1, matrix A represents the spectra and e records all the testing results associated with individual test cases. Similarly for SFL, the element a_{ij} in matrix A represents the coverage information of block b_j, by the test case t_i, with 1 indicating block b_j is executed, and 0 otherwise. e is a single columns matrix, if a test case t_i gives an error, the entry for element e_i is $'-'$ (failed run), and $'+'$ (passed run) otherwise. In MBD we only have failed runs (conflicts), therefore all the entries in error vectors e are $'-'$. But SFL considers failed and passed runs both, hence error vector is a mix of $'+'$ and $'-'$.

Figure 2.7 shows the spectrum matrix for the faulty C program presented in Figure 2.5. Program has 4 blocks and 6 different input vectors are given to construct 6×4 matrix.

For an input, if any program part is involved and output of the program is observed as an error, then that program part would be doubtful. Several different similarity coefficients [4] are proposed to quantify the probability that block b_j is faulty. Existing evaluation formulas include Jaccard [13], Ochiai [14], Tarantula [1], etc. Although similarity coefficient based approach is simple, diagnosis computed in this way is not optimal, in terms of accuracy. Therefore, Abreu proposed a solution, which combines spectrum-based diagnosis with model-based diagnosis. A Bayesian approach is used to diagnose multiple-fault candidates. In the next section, we describe the methodology that applies MBD principles on SFL.

2.4 Candidate Generation

Model-based reasoning can be used to generate possible fault candidates d_k. MBD approaches require modeling of every component and observation. Therefore, application of MBD on SFL requires some kind of modeling information of components. SFL generates a spectrum of program that does

c1	c2	c3	e
1	0	1	-
0	1	1	-
1	0	0	-
0	1	0	-
1	1	0	+

Table 2.4: (A,e) Spectrum matrix

not give detailed information about the behavioral modeling of the components involved, however it provides the same for the observations. A generic component model can be assumed to use model-based reasoning [7], here each entity of the program is considered as a component c_j. Component c_j is modeled in terms of the logic proposition

$$h_j \Longrightarrow (ok_{inpj} \Longrightarrow ok_{outj})$$

In the proposition, h_j models the health of the component c_j, just like MBD models. Variables ok_{outj} and ok_{inj} do not represent output and input values, rather they show the output and input correctness. The model specifies that if the component is healthy and the input is correct, then the output must be correct. The input value of a test vector is assumed to be correct and if we find an error at the primary output of the system, it means one or more components in the path from principal input to the principal output will be responsible for the error. Conflict row of the spectrum gives information about the involved components for observed error. To understand this principle, consider the spectrum (A, e) matrix shown in Table 2.4

We can logically infer the component health information from every conflict row of the spectrum (A, e). For this spectrum, we obtain the following propositions:

$$\begin{aligned}
\neg h_1 \vee \neg h_3 \quad & (c_1 \quad \text{and/or} \quad c_3 \quad \text{faulty}) \\
\neg h_2 \vee \neg h_3 \quad & (c_2 \quad \text{and/or} \quad c_3 \quad \text{faulty}) \\
\neg h_1 \quad & (c_1 \quad \text{faulty}) \\
\neg h_2 \quad & (c_2 \quad \text{faulty})
\end{aligned}$$

The above propositions are directly derived from the failing runs of the spectrum matrix. These propositions can be interpreted as a dependency list (DL) shown in Section 2.2.1.

STACCATO MHS algorithm presented in Section 2.2.2 is applied on the conflict sets above to generate minimal diagnosis candidates d_k. To obtain

the candidate ranking list Bayesian approach explained in Section 2.2.3 is used.

We can conclude that MBD principles can be combined with the spectrum based diagnosis to localize a fault in software. It should be noted that MBD on SFL technique does not require any detailed modeling of the components. The interesting question here is whether can SFL be used to diagnose a fault in the hardware to get rid of the hectic modeling of each component. This question is addressed by Wilson in [20] wherein he proposes *Bayesian Circuit Analysis by Topology (BACINOL)*. The next section presents the methodology involved in BACINOL.

2.5 BACINOL

BACINOL solves the diagnosis problem for hardware by executing all the steps as described for SFL. In software, each statement is considered as a component whereas the components are already defined for hardware (Section 2.1). The fundamental advantage that BACINOL offers is that each component here can be treated as a black box in contrast to MBD where a detailed model of every component was required. The generic model that can be used for BACINOL has been presented in Section 2.4.

2.5.1 Obtaining Spectra

Generally in software, instrumentation is used to generate spectrum information, but this method is not valid for hardware. Topological information provided by the manufacturer of hardware is used obtain a spectrum. This topological information provides structural model of the system.

Structural model is constructed from a set of nodes $V = I \cup G$, where I represents the inputs and G represents the components (gates), and a set of interconnection $(\alpha, \beta) \in E$ with $\alpha, \beta \in V$. In this approach, behavioral model of the components is not considered, hence each component is assumed to be a black box. Before going into the details of spectrum generation for hardware, we consider the following definition

Definition 2. *Define the set IN(v) as all* $n \epsilon V$ *for which there exists nodes* $v_1 = n, v_2, ..., v_n = v$ *such that* $(v_{i-1}, v_i) \epsilon A$ *for* $1 < i \leq n$. *Similarly, define the set OUT(v) as all* $n \epsilon V$ *for which there exists nodes* $v_1 = v, v_2, ..., v_n = n$ *such that* $(v_{i-1}, v_i) \epsilon A$ *for* $1 < i \leq n$.

Each row is determined by IN set of corresponding principle output. If component $c_j \epsilon IN(O_i)$, then $a_{ij} = 1$ else $a_{ij} = 0$. Observation for each output is considered separately and error vector combined with spectrum

matrix. Again consider the 3 inverter circuit presented in section 2.2.1 as an example. The circuit has two principle outputs y_1 and y_2. $IN(y_1) = \{c_1, c2\}$, this generates the first row of the spectrum (1, 1, 0). $IN(y_2) = \{c_3, c_1\}$ gives the second row (1, 0 , 1) of the spectrum. The observation we have is $(x, y_1, y_2) = (1, 0, 1)$, which means we have conflict at output y_1 (first row) whereas the output y_2 shows no inconsistency. Therefore we have following spectrum matrix (for a single observation):

c1	c2	c3	e
1	1	0	-
1	0	1	+

Table 2.5: Spectrum matrix for three inverter circuit

There is only one conflict in the matrix, this conflict will generate two minimal candidates c_1 and c_2. After applying Bayes rule on these candidates SFL produces the following ranking list:

Candidate (c_j)	Probability ($Pr(c_j)$)
c_2	0.68
c_1	0.37
c_3	0.05

Table 2.6: 1-dimensional ranking generated by BACINOL

Comparing the SFL spectrum matrix (Table 2.5) with that of MBD (Table 2.1). We observe that the MBD spectrum has two conflict rows because it uses modeling information. SFL however suffers from the absence of modeling information, which results in only one conflict in the spectrum matrix. MBD derives a single-fault (c_3) and double-fault (c_1, c_2) as the diagnosis, both of these diagnoses can correctly explain the inconsistency. Whereas SFL infers a single-fault c_2 and another single-fault c_1, the single-fault c_1 can not explain the observed behavior. Hence, we can conclude that SFL suffers from false positive as compared to MBD and this reduces diagnosis quality of SFL on hardware.

The generic model used by BACINOL is the primary reason for such false positive cases. The model is also responsible for the appearance of ambiguity sets in the final ranking list. Which we introduce in the next chapter. In addition, we also show the impact of size of ambiguity sets on the diagnosis quality.

Chapter 3

Ambiguity Set Problem

The last chapter summarized the different aspects of fault diagnosis. We saw that all the diagnosis techniques give a list in which components are ranked according to their fault probability $Pr(c_j)$. Faulty components can be found by traversing this list from top to bottom. Diagnosis quality of each technique is determined by how well components are ranked in the ranking list. In the first section, we present the concept of ambiguity sets. In the second section, we derive analytical models to measure diagnosis quality and we also show how the large size of ambiguity set lowers the diagnosis quality. In the last section, tie breaker for the ambiguity set is proposed.

3.1 Ambiguity Set

The ordered rank produced by BARINEL can have multiple components with the same rank in the ranking list. Components have same rank because BARINEL estimates the same posterior probability ($Pr(c_j)$) for all those components. This set of components is defined as an *ambiguity set*. One ranking list can have multiple ambiguity sets.

To understand the logic behind the same posterior probability, consider the circuit shown in Figure 3.1, also assuming that c_5 is at fault. Spectrum matrix for the circuit is presented in Figure 3.2 for 4 observations. The spectrum matrix leads to three minimal diagnosis candidate c_2, c_3 and c_5. BARINEL generates following ranking R for the candidates

$$\begin{array}{ll} c_3 & (0.45) \\ c_5 & (0.45) \\ c_2 & (0.1) \end{array}$$

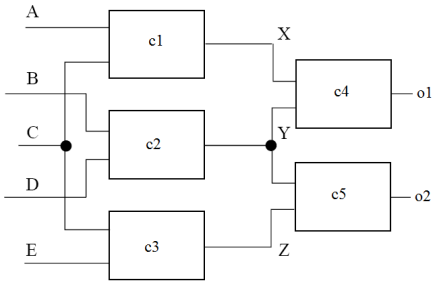

Figure 3.1: Hardware circuit

1	1	0	1	0	+
0	1	1	0	1	+
1	1	0	1	0	+
0	1	1	0	1	-
1	1	0	1	0	+
0	1	1	0	1	-
1	1	0	1	0	+
0	1	1	0	1	+

Figure 3.2: Spectrum matrix for the circuit

The ranking R has two components c_3 and c_5 with the same rank, both the components have same posterior probability (0.45). Following explains the reasons for this same ranking.

The ranking produced by the BARINEL is based on how components are involved in the pass and fail runs (pattern of spectrum matrix). Sometimes more than one component can have the same behavior in the spectrum. If components have exactly the same patterns in the spectrum matrix, there is no way BARINEL can discriminate among those components. Therefore, it gives same probability to all of those components. This is the primary reason for the appearance of the ambiguity sets in the ranking list.

Consider the spectrum matrix shown in Figure 3.2. Spectrum entries corresponding to c_3 and c_5 are exactly the same throughout the spectrum. In the conflict rows A_{4*} and A_{6*}, both the components are involved. In all the odd numbered rows where c_3 and c_5 are not involved, no conflicts are observed. In rows A_{2*} and A_{8*}, both c_3 and c_5 are involved but there are no conflicts observed. In these 2 rows, error generated by c_5 is masked by other components in the path. We can see that both c_3 and c_5 have exactly same pattern throughout the spectrum matrix, therefore BARINEL gives same posterior probability to c_3 and c_5. If we correlate this spectrum matrix with the circuit topology, we find that c_3 and c_5 are only involved in the cone of output O_2.

On the other hand, c_2 is involved in all the rows of spectrum. Involvement of c_2 in the passed row implies that the fault probability of c_2 is lower as compared to c_3 and c_5. Therefore, BARINEL computes the lower posterior for c_2 and assigns lowest rank to c_2.

In the next section, we will describe the methodology to estimate the diagnosis quality and will also show the impact of size of ambiguity set on the diagnosis quality.

3.2 Diagnosis Quality

Diagnosis quality is an important selection criterion for a diagnosis technique. The main objective of this thesis is to assess the improvement in the diagnosis quality of SFL on hardware (BACINOL) and compare it with the diagnosis quality of MBD. Actually there are many ways to interpret the diagnosis quality. It can be measured in terms of either time taken to diagnose the system or by complexity of the diagnosis algorithm. In this work we will quantify diagnosis quality in terms of *wasted effort*, which quantifies the time taken by a diagnostician while traversing the component ranking list produced by the diagnosis algorithm. Wasted effort can be defined as follows:

Definition 3. *Relative wasted effort W is equal to the number of healthy components (i.e. false positive) which must be inspected by a diagnostician before finding all the faulty components, divided by the total number healthy components in the system.*

For example consider the system with 10 components ($c_1 \dots c_{10}$)and component c_5 is the faulty component. Suppose, that BARINEL produces the following 1-D ranking list R:

$$\begin{aligned} c_2 &\quad (0.25) \\ c_7 &\quad (0.15) \\ c_3 &\quad (0.13) \\ c_5 &\quad (0.12) \\ c_9 &\quad (0.1) \\ c_6 &\quad (0.09) \\ c_1 &\quad (0.07) \\ c_4 &\quad (0.05) \\ c_{10} &\quad (0.03) \\ c_8 &\quad (0.01) \end{aligned} \tag{3.1}$$

In the list, c_5 is at 4^{th} position in the list, which means that the diagnostician must need to inspect component c_2,c_7 and c_3, before finally reaching to c_5. Hence before arriving at actual faulty component, three non-faulty components are inspected. There are total 9 non-faulty components in the system. Therefore, W = 3/9 = 0.33 .

Now consider the same system with 2 faulty components, assume that components c_5 and c_1 are faulty. In the ranking list, c_5 ranks at 4^{th} position and component 1 ranks at 7^{th} position. As mentioned above, before arriving on

c_5, three non-faulty components need to be inspected, but the diagnosis process will get over only when both of the faulty components will be detected. So, we continue inspecting the rest of the components. After checking components c_9 and c_6, we will hit the second faulty component, c_1. Hence, before detecting components c_1 and c_5, five non-faulty components(c_2, c_7, c_3, c_9, c_6) are inspected and one faulty component (c_5) is also detected. From this example, we can conclude that in case of multiple-fault, the ranking list needs to be traversed until faulty component with least probability (lowest rank) is not inspected. Therefore, the rank of the faulty component with the least probability determines W. This gives, W = 5/8 = 0.6 .

Note that in some cases the information that c_2, c_7, etc. are nominal can lead to an improved version of R. In this thesis, we do not take such an online rediagnosis model into account.

If we have a system with M components and the system has M_f faulty components, then the following factors need to be considered:

Let components $c_1 \dots c_k$ be the faulty components in system, $rank_{ck}$, the rank of the corresponding component c_k in the ranking list and $rank_{last}$, the rank of the component with the least probability. We define $rank_{last}$ as the

$$rank_{last} = \max(rank_{c1}, rank_{c2}, ...rank_{ck})$$

Total number of non-faulty components that need to be inspected before detecting lowest ranked faulty component = $rank_{last} - M_f$

Total number of non-faulty components in the system = $M - M_f$

Hence, W can be calculated as

$$W = \frac{rank_{last} - M_f}{M - M_f} \tag{3.2}$$

In the derivation of Equation 3.2, we have assumed that each of the component has a different fault probability and no faulty component is a part of an ambiguity set. Ambiguity set comes into the picture when the faulty component with least probability lies in an ambiguity set; otherwise, W can be computed with the Equation 3.2.

To understand how an ambiguity set changes the methodology to compute W, consider the ranking list shown below. Assume that there are 3 faulty components (c_7, c_4 and c_1).

$$
\begin{aligned}
c_2 \quad & (0.19) \\
c_7 \quad & (0.19) \\
c_3 \quad & (0.19) \\
c_5 \quad & (0.13) \\
c_9 \quad & (0.1) \\
c_6 \quad & (0.09) \\
c_1 \quad & (0.03) \\
c_4 \quad & (0.03) \\
c_{10} \quad & (0.03) \\
c_8 \quad & (0.03)
\end{aligned} \tag{3.3}
$$

Ranking list has two ambiguity sets and both of the ambiguity sets contain faulty components. But, we are concerned only with the lowest ranked faulty components. c_1 and c_4 have the least rank value (4^{th} rank). We define this rank as *rank of ambiguity set* (R_{as}). In the ambiguity set, we randomly pick the component to inspect. Hence statistically, all the faulty components are uniformly distributed over the complete ambiguity set [2]. Thus, for this case, the average number of non-faulty components, that will be inspected before detecting all faulty components is $\frac{2}{2+1}.4$.

If the size of the lowest ranked ambiguity set is given by S_{as} and the ambiguity set contains C_{as} number of faulty components, overall wasted effort will be calculated as

$$
W = \frac{R_{as} + \frac{C_{as}}{C_{as}+1}.S_{as} - C}{M - M_f} \tag{3.4}
$$

3.3 Ambiguity Set Problem

Wasted effort equation shown in Equation 3.4 depends on two parameters. First is the rank of ambiguity set (R_{as}) and second is the size of ambiguity set (S_{as}). In [20] diagnosis quality of BACINOL (application of SFL on hardware) is computed for 74XXX/ISCAS benchmark circuits and a lower diagnosis quality than MBD is observed. Application of SFL on 74XXX/ISCAS benchmark circuits suffers from ambiguity sets problem. Table 3.1 summarizes the average size of ambiguity sets for ISCAS benchmark circuits. These observations imply that increasing the size of the ambiguity set increases W. In BACINOL, we found that the size of ambiguity set is a major factor behind the lower performance of BACINOL as compared to MBD.

Circuit	Components	Possible Faults	Inputs	Outputs	Amb sets	Avg. Amb. Set. Size
74182	70	19	9	5	6	4
74283	104	36	9	4	9	5
74L85	108	33	11	3	4	14
74181	223	72	14	8	13	5
c432	423	150	36	7	15	10
c499	499	202	41	32	43	38
c880	880	327	60	26	39	10
c1355	1355	514	41	32	43	12
c1908	1908	857	33	25	57	15
c2670	2670	1147	233	140	119	10
c3540	3540	1634	50	22	112	15
c5315	5315	2254	178	123	284	8
c6288	6288	2416	32	32	62	40
c7552	7552	3488	207	108	251	14

Table 3.1: Summary of ISCAS benchmark circuits

Performance of SFL on hardware can be enhanced by reducing the size of ambiguity set. A tie breaker is needed to break the ambiguity set. In the next section, we propose a tie breaker for the ambiguity set.

3.4 Tie Breaker for Ambiguity Set

Section 2.2.3 shows that the BARINEL generates a ranking list based on the Bayesian update. The term $Pr(obs_i \mid d_k)$ gives the probability of an observation (obs_i) occurring, for a given diagnosis (d_k). The posterior probability is determined by an effective FNR parameter g_c (Section 2.2.3).

BARINEL estimates g_c by maximum likelihood estimation algorithm, which is dependent on the spectrum matrix. g values calculated with this method are an approximation of the actual g values. This algorithm gives the same g value to all of the components which are part of the ambiguity set. Hence, ambiguity cannot be broken with this g information.

The structural model of the hardware can be used for better estimation of g values. For example, consider the circuit shown in Figure 3.1. c_5 is closer to the principal output as compares to c_3. Therefore, probability that fault at c_5 can be seen at the output of circuit is higher as compared to c_3. This implies, that $g_3 > g_5$. Thus, if the structural model of the system is known,

g values can be estimated more accurately.

Therefore, we need an analytical model that can compute the g value for each of the components in the hardware by exploiting the topological information of the hardware. The next chapter develops three different probabilistic models to estimate the g value by computing error propagation probability of the component.

Chapter 4

Error Propagation Modeling

As the number of components in the hardware increases, the size of the ambiguity sets increase. Wasted effort equations clearly show that increasing the size of ambiguity sets lowers the diagnosis quality. The ambiguity sets can be broken by better estimation of FNR parameter (g). In the last chapter, we have shown that the g value calculated by BARINEL is not good enough to break the ambiguity sets. Therefore, we need a better methodology to compute the g values. In this chapter, we present three different probabilistic models to estimate the g values for every component. While developing the models, we assume that the structural model of the system is known. The g values calculated using probabilistic models are compared with the g values obtained from Monte Carlo simulations.

4.1 False Negative Rate

If a component is faulty in a system, it is not necessary that the fault in the component will always be observed at the output of the system. For some of the input vectors, the fault will be visible at the output and for some inputs the system will behave correctly. The error caused by the faulty component can either be suppressed by other faulty components or can be masked by other components in the system. As mentioned in Section 2.2.3, the fraction of test cases where the output stays nominal, despite the presence of a fault is called False Negative Rate (FNR).

For example, consider a simple logic circuit comprising of an INV gate ($c1$) connected to an AND gate ($c2$), as shown in Figure 4.1. Suppose, the INV gate has a fault (*stuck at 0*). For input $x = (X, 0)$ (X = don't care) an error at the output of c_1 will be masked by the fact that c_2 will always produce $y = 0$. However, for input $x = (X, 1)$ the inverter error will be propagated to y. Hence, we can conclude that for input vector $(X, 0)$, the system is

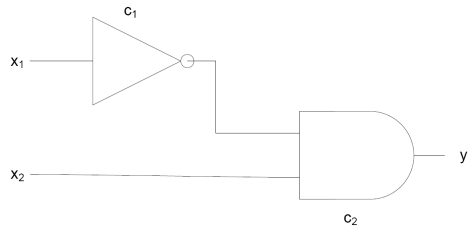

Figure 4.1: Example logic circuit

behaving correctly and for input vector $(X, 1)$ an error is observed at the system output. Thus the system's FNR is greater than zero.

FNR is similar to the concept of intermittency which is known in hardware rather than software. Most of the diagnosis techniques assume that the system is working non-intermittently. However, DeKleer addresses intermittent behavior of a component [9]. He defines conditional probability i_c as the probability that the component c is behaving correctly, given that c is faulty. i_c can be used to quantify the intermittent behavior of the component. [9] describes a simple approach to estimate the i_c. In this approach, a component is tested N times and the following factors are computed:

- $G(c)$ is the number of times c works correctly.
- $B(c)$ is the number of time c works incorrectly.

i_c is calculated as

$$i_c = \frac{G(c)}{G(c) + B(c)} \tag{4.1}$$

The above expression only computes the FNR of a single component in the system. BARINEL observes a principal output of the system to diagnose the fault. Therefore, the required g information to break the ambiguity set is determined by the FNR of the system. Hence, the objective of our work is to formulate the FNR of the complete *system* S, which is composed of components c_i. Just like DeKleer's approach, we can derive an expression that quantifies the FNR of a system. If we test the whole system N times and if it is known that components $c_1, \ldots, c_k$ are faulty, then the following factors will be considered:

- $G(S)$ is the number of times S is working properly, although one or more components are faulty.
- $B(S)$ is the number of times S is working incorrectly, given that one or more components are faulty.

If $g(S|c_1, \ldots, c_n)$ is the conditional probability that S behaves correctly, given that components $c_1, \ldots, c_n$ are faulty, the probability can be calculated as

$$g(S) = \frac{G(S)}{G(S) + B(S)} \tag{4.2}$$

In this work, the probability $g(S)$ is represented by the probability $g_{c_1,c_2,\ldots,c_k}$.

If we have information about the topology of the system, then our goal is to estimate the probability that the fault generated by a component c will be observed at the principal output of the system.

A faulty component c generates an error at the output of the component with the probability of $1 - i_c$ and this error will be propagated to the principal output of the system with the probability of $1 - g_c$. FNR evolves in the system because of error masking.

Let $epp(c)$ be defined as the *Error Propagation Probability (EPP)* of component c, which estimates the probability that an incorrect value is observed at the output of the faulty component. EPP of a component can be formulated as following

$$epp(c) = 1 - i_c \tag{4.3}$$

Let $epp(S)$ be defined as the EPP of a system S, which estimates the probability that an error generated by a faulty component will be observed at the principal output of the system. EPP of the system can be formulated as follows:

$$epp(S) = 1 - g_c \tag{4.4}$$

Therefore, $epp(S)$ is the probability that the system will produce an incorrect output when a component c is faulty.

In this chapter, we present analytical models to estimate the EPP of the system. We apply the following two steps to develop a system level EPP model:

1. First we derive the EPP model for a single component ($epp(c)$).
2. Subsequently, we derive the EPP model for a system ($epp(S)$) by composing EPP models of the components

The structural model of the system is utilized while deriving the EPP model for the system.

The following three kinds of probabilistic models have been developed:

1. Based on a deterministic component model
2. Based on an unknown component, but with known PDF information model
3. Based on black-box component model (no behavioral knowledge of the component is provided).

All models are simulated and validated by Monte Carlo (MC) simulations.

4.2 Deterministic Component Error Model

In this section, we derive a component error model based on the behavioral model of the component. If the behavioral model of components is known, it's easier to estimate the probability that a faulty component will produce an incorrect system output. The following variable are defined to formulate the EPP of a component

1. The *value truth probability* of a signal x_i is defined as the probability that the signal value will be 1, which is denoted by v_i.
2. The *error probability* of a signal x_i is defined as the probability that the signal will have an error, whose probability is denoted by e_i. The signal having error means that the correct value of the signal is flipped($x_i \rightarrow \neg x_i$).
3. The *health state* of a gate which denotes the probability of a gate being healthy is denoted by h. For sake of simplicity, only stuck-at-0 (SA0) and stuck-at-1 (SA1) fault models are considered for a faulty gate.

To derive the EPP model, first we need to think of the scenarios when an error will be observed at the output of the faulty component. Following are the two possible cases when an error will be observed at the output of the component

1. If the component is healthy but an input signal error propagates to the output of the component. An error will be detected, if the observed output value is different from the expected output value.

   For example, consider an AND gate with in_1 and in_2 as inputs and *out* as output. Suppose, $in_1 = 1$ and $in_2 = 1$ and one or both signals have error ($e_1 = 1$ and/or $e_2 = 1$). Because of the signal errors, one or both the input values will be flipped to 0 and hence the output would be 0, in this case the error will be detected because the correct

output is 1. However, an error at the output will not be observed, if the actual output is equal to the correct output.

If e_f is the probability that the input error will be propagated to the output of the component and h is the probability that the component will be healthy, then the EPP of the component is given by

$$epp = e_f h \tag{4.5}$$

2. An error will be observed, if the gate is faulty (non-healthy) and the correct output is the opposite of the stuck-at-fault value. For a faulty gate, the output only depends on the fault model; therefore the output value does not take into account the fact that the input signal has an error. For example, consider the same AND gate with SA1 fault. If $in_1 = 1$ and $in_2 = 0$, the correct output is 0, but because of SA1 fault, the output value will be observed as 1. For the AND gate, the error will be observed at the output of gate for all input combinations, except when both the input values are 1. When both input values are 1, then the correct output is equal to the stuck-at-value.

   Let p_x be the probability that the expected output is opposite of the value expected from stuck-at-fault and $1 - h$ is the probability that the gate is faulty. The probability that the error will be observed at the output is given by

$$epp = (1 - h)p_x \tag{4.6}$$

Combining Equations 4.5 and 4.6 gives

$$epp = e_f h + (1 - h)p_x \tag{4.7}$$

In [16] the following EPP model for a single gate is derived by Nasir

$$epp = e_f h + (1 - h)(1 - e_f) \tag{4.8}$$

The second term of Equation 4.8 is different from the second term of Equation 4.7. The second term in both the equations represents the EPP of the faulty gate. For the faulty gate, Nasir calculated that the error generated by a faulty component is propagated when the input signal has no error. The reason is that Nasir considers reliability issues of the component and so assumes bit flips as a fault. However, we consider the specific fault models of the gate, therefore we derive the EPP model specifically for SA0 and SA1 faults. The example presented in Section 4.1 shows that sometimes the error is masked by the output of the faulty gate, even though the input has no error. Therefore, we use the term p_x.

In_1	In_2	e_1	e_2	Expected Output	Actual Output	e_f
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	1	1
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	1	1	0
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	0	1

Table 4.1: Truth table of AND gate with input signal errors

To determine the EPP, first the e_f value of the gate should be calculated. The simplest method to derive the expression for e_f is to use the truth table for the gate combined with error signal values at the input.

Consider an AND gate, Table 4.1 represents the truth table with error values at the input signals. First two columns represent the input single values. 3^{rd} and 4^{th} columns represent errors at the input signal, $e_i = 1$ means in_i has an error. In this table, we can easily observe the combinations of input signal values and the error values where the error can be observed ($e_f = 1$).

For example, consider the 10^{th} row, where $in_1 = 1$ and $in_2 = 0$, and suppose the input error probabilities are $e_1 = 0$ and $e_2 = 1$. In this case, in_2 flips and the gate sees the wrong inputs as $in_1 = 1$ and $in_2 = 1$. The gate produces 1 as the output but the expected output is 0, therefore the input signal error is observed at the output ($e_f = 1$).

Consider a different signal error combination for the same input values (11^{th} row), where $e_1 = 1$ and $e_2 = 0$, because of the signal error, the gate has inputs $in_1 = 0$ and $in_2 = 0$ and produces a 0 as the output, which is the same as the expected output. In this case, the input signal error does not propagate to the output ($e_f = 0$).

For the AND gate, we observe that if one of the inputs (in_1) has an error, then in order to propagate that error to the output, the other input (in_2) value should be 1. To determine the probability that error will be prop-

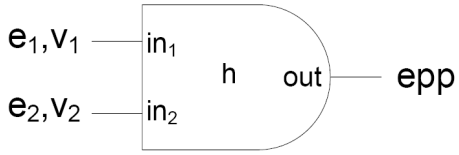

Figure 4.2: AND gate with EPP variable

agated to the output, it is important know the value of the other signal. For the binary case, only two values for a signal are possible, 0 and 1. v_i represents the probability that signal in_i has value 1, and $1 - v_i$ represents the probability that the signal has value 0. Therefore, the concept of value truth probability is introduced in the EPP model.

To determine the general expression of the probability that the input error will be propagated to the output (e_f) of an AND gate, the following cases should be considered:

- in_1 **at error and** in_2 **not :** Probability that in_1 is at error is e_1, probability that in_2 is not at error is $1 - e_2$ and probability that in_2 will be at 1 is v_2. So the probability that the error will be propagated to the output of gate is given by

$$e_f = e_1(1 - e_2)v_2$$

- in_2 **at error and** in_1 **not :** Probability that in_2 is at error is e_2, the probability that in_i will not be at error is $1 - e_1$ and the probability that in_1 will be at 1 is v_1. So the probability that the error will be propagated to the output of gate is given by

$$e_f = e_2(1 - e_1)v_1$$

- in_2 **and** in_2 **are at error :** Probability that both the inputs are at error is e_1e_2. Error will be propagated when inputs are 1 which has probability v_1v_2 or both inputs are 0 which has probability $(1-v_1)(1-v_2)$. So the probability that the error will be propagated to the output of the gate is given by

$$e_f = e_1e_2((1 - v_1)(1 - v_2) + v_1v_2)$$

Combining the equations derived in the above cases

$$e_f = e_1(1 - e_2)v_2 + e_2(1 - e_1)v_1 + e_1e_2((1 - v_1)(1 - v_2) + v_1v_2)) \quad (4.9)$$

For primary inputs of a binary gate, $v_1 = v_2 = \frac{1}{2}$.

Consequently,

$$e_f = \frac{1}{2}(e_1 + e_2 - e_1e_2) \tag{4.10}$$

According to Equation 4.7, for a faulty component we need to estimate p_x, which is the probability that the output value of the component is different from its stuck-at-value. The following explains the derivation of p_x for an AND gate. Suppose, the AND gate is SA1, for $in_1 = 1$ and $in_2 = 1$, the fault will go undetected. However, for the rest of the input combination, an error will be detected. Hence, for 3 out of 4 input value combination, an error will be observed in case of an SA1 fault, hence $p_x = \frac{3}{4}$. Consequently, from Equation 4.7 it follows that:

The EPP model for AND gate which is SA1

$$epp = \frac{h}{2}(e_1 + e_2 - e_1e_2) + \frac{3}{4}(1 - h) \tag{4.11}$$

For SA0 fault, for 1 out of 4 cases, an error will be observed at the output which implies $p_x = \frac{1}{4}$.

$$epp = \frac{h}{2}(e_1 + e_2 - e_1e_2) + \frac{1}{4}(1 - h) \tag{4.12}$$

If the fault type is unknown, it can be either SA0 or SA1. In this case, we can assume that the fault type is random. To obtain the EPP of the gate which is stuck at a random value, we just need to take the average of Equations 4.11 and 4.12, which results in the following expression:

$$epp = \frac{h}{2}(e_1 + e_2 - e_1e_2) + \frac{1}{2}(1 - h) \tag{4.13}$$

In other words, we can say that for random fault types, $p_x = \frac{1}{2}$.

We also compute the EPP for other logic gates. Following are the EPP equations for other logic gates:

- OR Gate:

$$epp = h(e_1(1 - v_2) + e_2(1 - v_1) + e_1e_2(2v_1v_2 - 1)) + \frac{1}{2}(1 - h) \tag{4.14}$$

- XOR Gate:

$$epp = h(e_1 + e_2 - 2e_1e_2) + \frac{1}{2}(1 - h) \tag{4.15}$$

- XNOR Gate:

$$epp = h(e_1 + e_2 - 2e_1e_2) + \frac{1}{2}(1 - h) \tag{4.16}$$

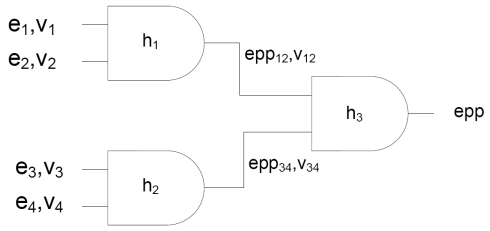

Figure 4.3: Circuit consisting of AND gates

- BF1 Gate:

$$epp = he_1 + \frac{1}{2}(1 - h) \tag{4.17}$$

- BF2 Gate:

$$epp = he_2 + \frac{1}{2}(1 - h) \tag{4.18}$$

BF1 and BF2 are gates with bridging fault.

An important observation that we can make out of the above the equations is that only AND and OR gates include the value truth probabilities (v_1 and v_2) in their EPP models. For the rest of the gates, the EPP models do not require value truth probabilities. For AND and OR gates, it is important to know the value of the other input signals in order to propagate the error, because the error can be masked by the other input signals of the gate. For the remaining gates, EPP does not depend on the value of other signals.

This model gives the EPP a single component whose behavioral model is known. In the next section we develop the EPP model for a complete circuit (system).

4.2.1 Circuit Error Model

In this section, the single component error model is used to calculate the probability that the primary input signal error will propagate to the primary output of the circuit. If a multilevel logic circuit is given, we start from the primary input and move towards the primary output. For each gate, we calculate the output error probability (EPP) using the value truth probabilities (v_i), the input error probabilities (e_i) and the gate fault probability (h). For the next level, the input error probability corresponds to the output error probability of the previous level. The value truth probability for the next level depends on the type of the gate and it is determined by the value truth probability of the previous level.

For example, consider the circuit shown Figure 4.3, first we calculate the output EPP for the first level (epp_{12}) and (epp_{34})

$$epp_{12} = \frac{h_1}{2} \cdot (e_1 v_2 + e_2 v_1 - e_1 e_2 (1 - 2(v_1 + v_2) + 2 v_1 v_2)) + \frac{1}{2}(1 - h_1) \quad (4.19)$$

$$epp_{34} = \frac{h_2}{2} \cdot (e_3 v_4 + e_4 v_3 - e_3 e_4 (1 - 2(v_3 + v_4) + 2 v_3 v_4)) + \frac{1}{2}(1 - h_2) \quad (4.20)$$

We also need to calculate the value truth probability for the next level. For the AND gate, the output is 1 when both the inputs are 1, hence the value truth probability for the second level (v_{12} and v_{34}) of the circuit can be derived as

$$v_{12} = v_1 v_2 \quad (4.21)$$

$$v_{34} = v_3 v_4 \quad (4.22)$$

The final output error probability is

$$epp = \frac{h_3}{2} \cdot (e_{12} v_{34} + e_{34} v_{12} - e_{12} e_{34} (1 - 2(v_{12} + v_{34}) + 2 v_{12} v_{34})) + \frac{1}{2}(1 - h_3) \quad (4.23)$$

In the expansion of Equation 4.23, the higher order(greater than 1) exponent terms of v_1, v_2, v_3 and v_4 appear, when e_{12} is multiplied with v_{12} and e_{34} is multiplied with v_{34}. However, these higher order exponent terms need to be suppressed to the first order exponent, to get the correct EPP expression ($v_i^2 \rightarrow v_i$). This is called *high order exponent suppression* and it is essential in order to avoid duplicate counting of the terms in the final expression [17].

To understand the phenomena of the suppression, consider an AND gate, whose both inputs are short and the input signal has value truth probability v_1. The probability that the output of the AND will be 1 is v_1 rather than v_1^2. To get an accurate analytical expression for EPP, the higher-order exponent terms of $v_1, \ldots, v_n$ should be suppressed to first order.

Equations 4.13 to 4.18 conclude that except AND and OR gates, the EPP model for other gates is independent of the value truth probabilities. Suppression is not required in order to derive the correct EPP model for the multilevel circuit consisting of gates apart from AND and OR gates. Suppression has an exponential space complexity. Therefore, in the rest of the thesis, we will not apply suppression for any kind of gate while composing the EPP expression for the multilevel circuit. Hence, the final expressions derived for all the circuits are approximate. In the last section, we will present the errors in EPP calculation caused by this approximation.

We use Monte Carlo (MC) simulations to validate the correctness of EPP models developed in the section. The correct EPP value is computed by MC simulations and is compared with value obtained by EPP. Error in EPP estimation is computed with the accuracy till 3 decimal with 95% of confidence interval. MC simulations confirm the correctness of EPP model for single gate and composition of EPP models for a circuit.

EPP models derived in the previous section had complete behavioral knowledge of the components. In the next section, we present the EPP model for a system where we do not have the exact behavioral model of the components.

4.3 Error Model of a Component with known PDF

In this section, we propose an approach where we abstract from the specific gate models, such that the individual v_is are no longer required. The only information that we have is the Probability Density Function (PDF) of value 1 as the outcomes of the component. In simple words, we know how many entries in the component truth table have 1 as the output value, however we do not have the exact truth table.

In case of binary gates, a PDF reduces to one parameter a which is defined as the *truth probability* of the component (i.e. the fraction of 1s in the truth table output). Consider the truth table of the 2 input AND gate, 1 appears only once and the truth table has 4 entries, thus $a = \frac{1}{4}$. Note, that in this approach we do not distinguish between any gates that have $a = \frac{1}{4}$ (i.e. 4 binary gates, of which AND is only one example).

Practically, the a information for a component can be obtained during system testing, by observing the output values of a component. Hence, there is no need to develop the exact behavioral model of the component to determine the a value. This is the primary motivation behind development of this EPP model.

Section 4.2 illustrates the methodologies to detect an error for the deterministic component. An error will be detected only when the actual output will be different from the correct output. To determine the EPP of the non-deterministic component, we need to compute the two following probabilities

1. The probability that the actual output will be 1 or 0, this probability will be given by $Pr(1)(a)$ or $Pr(0)(1-a)$ respectively.

2. The probability that the actual output is different from the correct output, this conditional probability is given by $Pr(1|0)$ or $Pr(0|1)$. Here $Pr(1|0)$ expresses the probability that error at input signals will

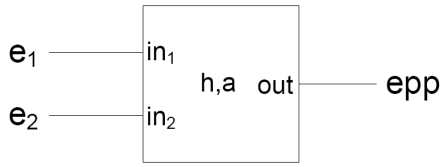

Figure 4.4: Unknown gate with EPP variables

change output value from 0 to 1 and $Pr(0|1)$ expresses the opposite change in output value (1 to 0)

Consider a component with two inputs in_1 and in_2, the PDF parameter of the component is a and component has a output out. Following approach is then used to derive the EPP for the binary gates, based on the a information

1. Suppose, that the correct input values will produce $out = 1$ (probability a)
2. Any error at either input (probability $e_1 + e_2 - e_1e_2$) will lead to $1 \rightarrow 0$ transition of out (probability $Pr(0|1)$)

From the above prepositions, e_f (probability that error at input will be propagated to the output) can be calculated, if $out = 1$ corresponds to the correct input values.

$$e_f = (e_1 + e_2 - e_1e_2) \cdot a \cdot Pr(0|1) \tag{4.24}$$

Similarly, we can derive the e_f, if $out = 0$ corresponds to the correct input values

$$e_f = (e_1 + e_2 - e_1e_2) \cdot (1 - a) \cdot Pr(1|0) \tag{4.25}$$

Combining Equation 4.24 and 4.25, gives the following overall e_f expression

$$e_f = (e_1 + e_2 - e_1e_2) \cdot (a \cdot Pr(0|1) + (1 - a) \cdot Pr(1|0)) \tag{4.26}$$

Complete EPP model can be derived by combining Equation 4.7 and 4.26 as follows

$$epp = h \cdot (e_1 + e_2 - e_1e_2) \cdot (a \cdot Pr(0|1) + (1 - a) \cdot Pr(1|0)) + (1 - h) \cdot p_x \tag{4.27}$$

As discussed in Section 4.2, if we assume that the fault type is random, then $p_x = \frac{1}{2}$. Hence EPP model can be written as

$$epp = h \cdot (e_1 + e_2 - e_1e_2) \cdot (a \cdot Pr(0|1) + (1 - a) \cdot Pr(1|0)) + \frac{(1 - h)}{2} \quad (4.28)$$

The above expression can be easily generalized in terms of a, if $Pr(0|1)$ and $Pr(1|0)$ can be expressed in terms of a. The following approach derives the EPP expression in terms of a:

Suppose, for an n input truth table, the possible number of $1s$ that can come as output are x_1. So, a can be computed as

$$a = \frac{x_1}{2^n}$$

In the truth table, $2^n - x_1$ entries correspond to 0. If we know that one entry in the truth table is 1, then $2^n - 1$ unknown entries remain in the truth table. The probability ($Pr(0|1)$) that error at input signals leads to a truth table entry corresponds to value 0, out of remaining $2^n - 1$, can be represented as

$$Pr(0|1) = \frac{2^n - x_1}{2^n - 1}$$

Above expression can be simplified in terms of a, as

$$Pr(0|1) = \frac{1 - a}{1 - \frac{1}{2^n}} \quad (4.29)$$

For our case, $n = 2$ and that corresponds to

$$Pr(0|1) = (1 - a) \cdot \frac{4}{3} \quad (4.30)$$

Similarly, we can derive an Equation for $Pr(1|0)$, according to the following relation

$$Pr(1|0) = \frac{a}{1 - \frac{1}{2^n}} \quad (4.31)$$

For our case, $n = 2$ and that corresponds to

$$Pr(1|0) = a \cdot \frac{4}{3} \quad (4.32)$$

Combining the Equations 4.28, 4.30 and 4.32 gives the following expression

$$epp = \frac{8}{3} \cdot h \cdot (e_1 + e_2 - e_1e_2) \cdot a \cdot (1 - a) + \frac{1 - h}{2} \tag{4.33}$$

We can substitute different values of a, to get their EPP expression.

For example, we consider $a = \frac{1}{4}$.

Using Equation 4.33 we can easily calculate the e_f

$$\begin{aligned} e_f &= (e_1 + e_2 - e_1e_2)(\frac{1}{4} \cdot 1 + \frac{3}{4} \cdot \frac{1}{3}) \\ e_f &= \frac{1}{2} \cdot (e_1 + e_2 - e_1e_2) \end{aligned} \tag{4.34}$$

We can validate the correctness of above model by using the deterministic EPP model for all the truth tables corresponding to $a = \frac{1}{4}$. Possible truth tables (for binary gates) outcomes for $a = \frac{1}{4}$ is:

$$\begin{aligned} T_1 &= \{0, 0, 0, 1\} \\ T_2 &= \{0, 0, 1, 0\} \\ T_3 &= \{0, 1, 0, 0\} \\ T_4 &= \{1, 0, 0, 0\} \end{aligned} \tag{4.35}$$

If a component behaves according to any of these tables, e_f of the component can be given as

$$e_f = \frac{e_f(T_1) + e_f(T_2) + e_f(T_3) + e_f(T_4)}{4} \tag{4.36}$$

e_f value for each of the truth tables can be computed as

$$e_f(T_1) = e_f(T_2) = e_f(T_3) = e_f(T_4) = \frac{1}{2} \cdot (e_1 + e_2 - e_1e_2) \tag{4.37}$$

Combining Equations 4.36 and 4.37 gives

$$e_f = \frac{1}{2} \cdot (e_1 + e_2 - e_1e_2) \tag{4.38}$$

Equations 4.34 and 4.38 are same, this proves the correctness of the model presented in Equation 4.33.

Similarly, we have derived and validated the EPP models for various values of a.

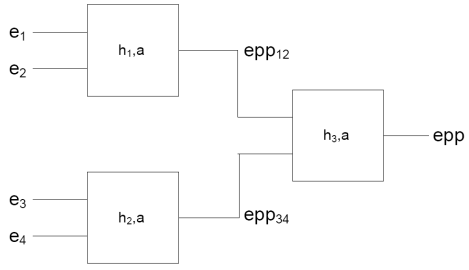

Figure 4.5: Circuit consisting of unknown components

4.3.1 Circuit Error Model

To calculate the EPP model for the complete circuit, we can apply the composition on the EPP model of a component (Equation 4.33). The model of a component does not involve the value truth probability (v_i) in their expression. Therefore, there is no need to use suppression in the composition. This is the advantage of the model, because the automation of EPP calculation can be performed without having exponential space complexity.

For example, consider the circuit shown in Figure 4.5,. Each of the components in the circuit has the following a value

$$a = \frac{1}{2}$$

EPP at the first level of circuit is denoted by epp_{12} and epp_{34}, which can be computed as

$$e_{12} = \frac{2}{3} \cdot h_1 \cdot (e_1 + e_2 - e_1 e_2) + \frac{1}{2} \cdot (1 - h_1)$$
$$e_{34} = \frac{2}{3} \cdot h_2 \cdot (e_3 + e_4 - e_3 e_4) + \frac{1}{2} \cdot (1 - h_2)$$

The EPP at the output of the circuit is given by

$$epp = \frac{2}{3} \cdot h_3 \cdot (e_{12} + e_{34} - e_{12} e_{34}) + \frac{1}{2} \cdot (1 - h_3) \tag{4.39}$$

We use MC simulations to validate the correctness of EPP models developed in the section. The correct EPP value is computed by MC simulations and is compared with value obtained by EPP. Error in EPP estimation is computed with the accuracy till 3 decimal with 95% of confidence interval. MC simulations confirm the correctness of EPP model for single gate and composition of EPP models for a circuit.

4.4 Black Box Component Error Model

In the context of our work where we need to estimate g value to enhance the performance of SFL, no modeling information can be assumed. Neither do we have the behavioral model of the component, nor do we have the output PDF information. Thus, every components can be assumed to be a black-box. In that case we need to abstract even further, and simply assume that the outcome of a component is randomly distributed over all possible outcomes.

This means that all the possible outcomes are assumed to have equal probability i.e. $a = \frac{1}{N}$ to appear at the output of a component, where N is the number of possible outcomes of the component. For binary gates, $N = 2$, hence both 0 and 1 can appear at the output with equal probability of $\frac{1}{2}$.

The EPP model of the component can be derived with the methodology described in Section 4.3. The EPP expression would be

$$epp = h.(e_1 + e_2 - e_1e_2) \cdot (a \cdot Pr(0|1) + (1 - a) \cdot Pr(1|0)) + \frac{(1 - h)}{2} \quad (4.40)$$

There is no information available about the truth table and the number of entries with output 1 are unknown. Since outcomes are randomly distributed, half of the output entries can assumed to be 1, therefore $a = \frac{1}{2}$. Because of the unknowns truth table, however we cannot compute $Pr(0|1)$ or $Pr(0|1)$, as the number of 0's and 1's are unknown. Rather, it holds $Pr(0|1) = Pr(0) = (1a) = \frac{1}{2}$ and, similarly, $Pr(1|0) = a = \frac{1}{2}$. Substituting values of a and conditional probabilities give the following:

$$epp = h \cdot (e_1 + e_2 - e_1e_2) \cdot (\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2}) + \frac{(1 - h)}{2}$$

$$epp = \frac{h \cdot (e_1 + e_2 - e_1e_2)}{2} + \frac{(1 - h)}{2} \quad (4.41)$$

4.4.1 Circuit Error Model

Reconsider the circuit shown in Figure 4.5. Assume each component as a black-box and that we do not have any behavior model of each. Therefore, $a = \frac{1}{2}$ for every component and EPP of the component is the same as we derived in Equation 4.41. EPP of the circuit can be derived by composing EPP model of the components. The methodology discussed in Section 4.3.1 can be used to compose the EPP model.

Single component EPP model does not involve value truth probabilities (v_i). Therefore, suppression is not required during the composition of the black-box EPP models.

We use MC simulations to validate the correctness of EPP models developed in the section. The correct EPP value is computed by MC simulations and is compared with value obtained by EPP. Error in EPP estimation is computed with the accuracy till 3 decimal with 95% of confidence interval. MC simulations confirm the correctness of EPP model for single gate and composition of EPP models for a circuit.

4.5 FNR Calculation

To obtain the value of g_1, for a component c_1, we can use models derived in the previous sections. We already know the following relation between g value and the EPP.

$$g_1 = 1 - epp(S)$$

We need to calculate the EPP for a system S, given that component c_1 is faulty. An error in the system S is generated by the faulty component c_1. The generated error then propagates from output of c_1 to the principal output of the system. To compute the probability that the generated error will be observed at the output of the system ($epp(S)$. First we need to compute the probability that the generated error will be observed at the output of the faulty component ($epp(c_1)$). The following section describes the EPP calculation of a faulty component.

4.5.1 EPP of Faulty Component

An error at the output of a component will be observed, if the actual output is different from the stuck-at-value. Suppose, the component c_1 is SA1, an error at the output of c_1 will be observed when the expected output is different from the correct output.

If $Pr(SA1)$ is the probability that the component will be SA1 and $Pr(0)$ is the probability that the output of the component will be . An error will be observed when the component is SA1 and the actual output value is 0. $epp(c_1)$ can be computed as

$$epp(c_1) = Pr(SA1) \cdot Pr(0) \tag{4.42}$$

Similarly, we can compute the $epp(c_1)$, when component is SA0

$$epp(c_1) = Pr(SA0) \cdot Pr(1) \tag{4.43}$$

Combining Equations 4.42 and 4.43 results in the following

$$epp(c_1) = Pr(SA1) \cdot Pr(0) + Pr(SA0) \cdot Pr(1) \tag{4.44}$$

In this thesis, we are considering only two kinds of faults, SA1 and SA0. Both these faults are equally probable. So

$$Pr(SA1) = Pr(SA0) = \frac{1}{2}$$

We do not have any information about the behavioral model of the component, so it can be assumed to be a black-box. Therefore, the output of a component can be 1 or 0 with an equal probability. Hence,

$$Pr(0) = Pr(1) = \frac{1}{2}$$

Substitution of $Pr(0)$ and $Pr(1)$ in Equation 4.44 gives

$$epp(c_1) = \frac{1}{4} + \frac{1}{4}$$

$$epp(c_1) = \frac{1}{2} \tag{4.45}$$

Hence, we can conclude that a faulty component generates EPP of value $\frac{1}{2}$, which is effectively averaged over all possible behaviors of the component.

After computing the EPP of the faulty component ($epp(c_1)$), the next step is to estimate the EPP of the system ($epp(S)$). To compute $epp(S)$, we compose the EPP model of every component involved in the path from c_1 to the principal output. The composition is described in the next section.

4.5.2 EPP of a System

We have three models to calculate the EPP of a system. In this section, composition of the deterministic EPP model and the black-box EPP model is used to compute the EPP of the system. We do not discuss the known PDF EPP model because it has the same methodology as that of the black-box EPP model.

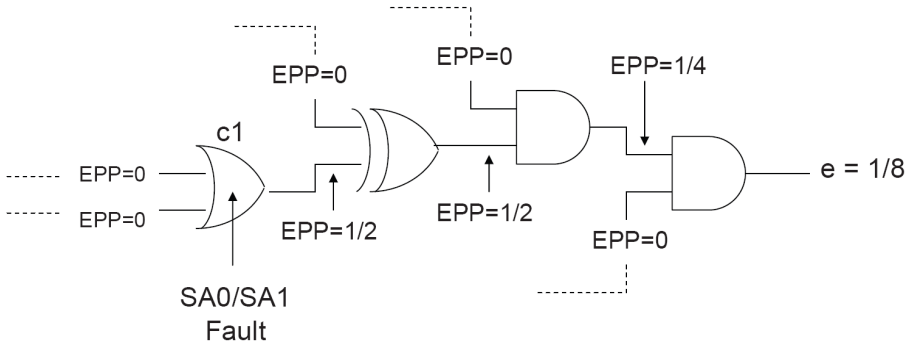

Figure 4.6: General logic circuit

Deterministic Model

The first model is the deterministic model. Composition of deterministic models suffers from the suppression problem. If we assume that the circuit has no reconvergent subcircuit, single-fault g can be computed without suppression because higher-order terms of v_i do not appear when composing the EPP model for a system with single faults (discussed later). However, computing the g factor for the multiple-fault candidates is not performed due to exponential number of possible multiple-fault combinations, that need to be taken into account. Rather, we compute the single-fault g, and estimate g for multiple-fault candidates based on a mode that computes multiple-fault g values using the single-fault g values (e.g., OR-model, Section 2.2.3).

The use of the OR model implies that we only need to compute the EPP for a circuit with one faulted gate. This has interesting consequences for the EPP calculation as shown by the following.

Consider the circuit shown in Figure 4.6.

As there is only one faulty gate, all inputs outside the path from c_1 to the output have all inputs have $e_i = 0$. Consequently, all deterministic gate models (Equations 4.13 - 4.18) reduce to

- AND gate

$$e_1 v_2 \tag{4.46}$$

- OR gate

$$e_1(1 - v_2) \tag{4.47}$$

- X[N]OR gate

$$e_1 \tag{4.48}$$

Here we have chosen e_1 as the gate input that is within the error propagation path from the faulty gate to the primary output (i.e., $e_2 = 0$). It can be easily seen that the composition of the above reduced EPP models do not suffer from the suppression problem (unless there are reconvergent subcircuits).

Typically, the estimation of g_j is static, i.e., the actual signal values v_i are not known. Consequently, in the above EPP models we will assume $v_2 = 1/2$. This implies that for AND and OR gates, the EPP is attenuated by a factor 2 while X[N]OR gates pass on the EPP. In the example circuit in Figure 4.6 it follows that $e = 1/8$, and, consequently, $g_1 = 7/8$.

Note that the deterministic model applies only if the circuit does not include reconvergent subcircuits, because it introduces the suppression problem. However, the deterministic model can be applied at minimal loss of accuracy as long as reconvergence is small. This can be shown in experimental results.

Black Box Model

The goal of this thesis is to improve the quality of a diagnosis technique (SFL on hardware) that does not require the modeling of components to compute the diagnosis. Hence, the black-box component EPP model is our only option, as it does not require any kind of information regarding the behavior of the component. Assume that except component c_1, each of the involved components is healthy ($h = 1$). Hence each component will propagate error according to following (substitute $h = 1$ in Equation 4.41) :

$$epp = \frac{e_1 + e_2 - e_1 e_2}{2}$$

As there is only one faulty gate (c_1), all inputs outside the path from c_1 to the output have $e_i = 0$. If we assume that there is no re-convergent sub-circuit in the system, then there is only one path through which error propagates to the principal output. There we have chosen e_1 as the gate input error probability that is within the error propagation path from the faulted gate to the primary output (i.e. $e_2 = 0$). Therefore, the above equation reduces to

$$epp = \frac{e_1}{2} \tag{4.49}$$

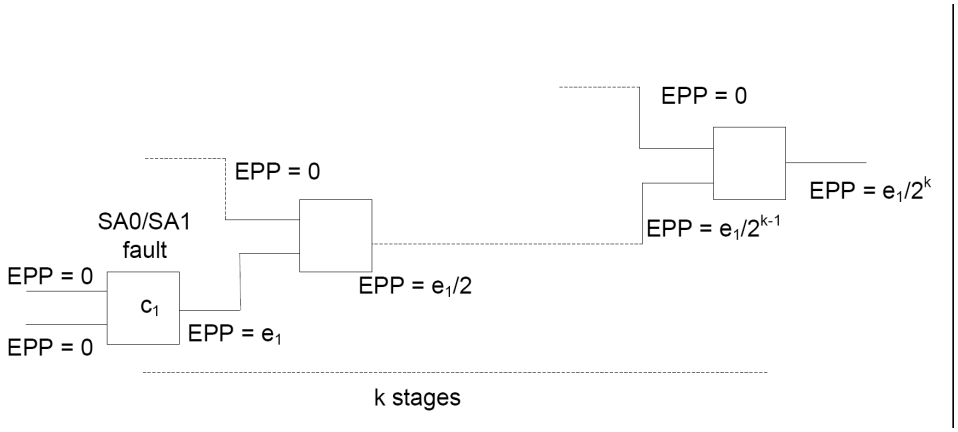

Figure 4.7: General circuit, consisting of black-box components

From the above equation we can infer that from the faulted gate to principal output, EPP reduces with half at each stage. As we can seen in Figure 4.7, faulted gate c_1 is located at the k^{th} stage from the principal output ($k = 0$ for principal output). If EPP at the output of c_1 is given by e_1, EPP of the system can be expressed as:

$$epp(S) = \frac{e_1}{2^k} \tag{4.50}$$

In Equation 4.45, we derived the value of EPP at the output of faulty component c_1. Therefore,

$$e_1 = \frac{1}{2}$$

Substituting value of e_1 is Equation 4.50 gives

$$epp(S) = \frac{1}{2^{k+1}}$$

Consequently, g_1 can be computed by the following equation

$$g_1 = 1 - \frac{1}{2^{k+1}} \tag{4.51}$$

Equation 4.51 implies that if the structural model of the system (value of k) is known, g value for every component can be calculated very easily. Reconsider the circuit shown in Figure4.6, assume that all the gates are black boxes and we do not know the behavior of the component. Let the value of k for component c_1 be 3, which makes $g_1 = 1 - \frac{1}{16}$. Similarly, we can compute the g value for all the components that are a part of the system.

4.6 EPP Accuracy Results

In this section, we present accuracy results for all the three EPP models for a number of circuits from the 74XXX/ISCAS85 benchmark [8, 11]. The benchmark circuits have been already mentioned in Table 3.1.

The predictions of the deterministic model (Equation 4.13 - 4.18) and the known PDF model model(Equation 4.33) and the black box model (Equation 4.41), are compared to the results of Monte-Carlo simulation (random inputs and fault injections), yielding prediction errors ϵ_1, ϵ_2, and ϵ_3, respectively, according to

$$\epsilon_i = |\frac{e_i - e_{MC}}{e_i}|$$

Each circuit output is considered as a separate (sub) circuit (cone).

Table 4.2 shows the mean (E), variance (V), lower bound (L), and upper bound (U) of the relative errors ϵ_1 through ϵ_3, respectively, for single-fault injections (rounded to 3 decimals). We show the accuracy results per principal output (PO) for the smaller 74XXX circuits, while for the ISCAS circuits we summarize the error statistics for all principal outputs (PO) combined.

Not all ISCAS circuits have been tested. Some of the circuits include gates with a fanin greater than 5 for which we haven't derived the EPP models. Furthermore, the two largest circuits took too long for the Monte Carlo simulations to finish in time. As expected, the deterministic model performs very well. The average prediction error, due to reconvergence, is quite small. Outliers are within 10%, with the exception of c880. The probabilistic models perform less accurately, but still with an average error well below 10%, with outliers up to 60%.

Table 4.3 and 4.4 show similar data for double and triple faults, respectively. For multiple-fault it was expected that average error will higher for deterministic EPP model because we ignored the suppression while composing the EPP model. But average error is still in the order of percents (less than 10%), although the upper bound increases (up to 36%). However, the error increases with fault cardinality as the need for suppression increases. The average error of the probabilistic models is much higher (up to 19%) with outliers up to 62%, and tends to increases with cardinality. For all cardinalities the difference between both probabilistic models is relatively small. The significance of the latter result is that knowledge of the average truth probability of a gate (2^{nd} model) does not significantly improve prediction performance compared to no information at all (3^{rd} model).

4.7 Discussion

In this chapter, we have derived three different EPP models for better estimation of FNR parameter g. This section summarizes all the important findings of this chapter:

- A deterministic EPP model exploits the structural model of the system and the behavioral model of all the components to estimate the g value.
- Because of the fact that dealing with suppression entails an exponential space complexity; deterministic models have been approximated by ignoring the suppression while composing the EPP model. Accuracy results clearly show that, despite the approximation, the deterministic model is the most accurate model as compared to the other last EPP two models.
- BARINEL computes the multiple-fault g values using OR model, therefore only the single-fault g value is needed. single-fault g value computation does not suffer from the suppression problem, if a circuit has no reconvergence subcircuits.
- EPP models clearly show that the g value of a component increases, if the number of stages between the faulty component and the principal output increases. This proposition can be understood analytically, if a faulty component is away from the principal output, the probability that the error generated by the faulty component will be masked by other component, is higher. Therefore, the probability that the faulty behavior of the component will not be observed at the principal output increases.
- For AND, OR, NAND, NOR and INV gates the known PDF EPP model and the black-box EPP model are same, on the other hand, for XOR, NXOR, BF1 and BF2 gate the models are different. Since, ISCAS/74XXX circuits have a huge number of AND, OR, NAND, NOR and INV gates, therefore the accuracy results obtained for these probabilistic models are very close to each other.
- As expected, error in EPP computation increases with the fault cardinality, except for deterministic EPP. Deterministic EPP model is always pretty close to the accurate and can be assumed as the most accurate. As the fault cardinality increases difference between deterministic model and other two EPP increases.

Circuit	PO	$\mathsf{E}[\epsilon_1]$	$\mathsf{V}[\epsilon_1]$	$\mathsf{L}[\epsilon_1]$	$\mathsf{U}[\epsilon_1]$	$\mathsf{E}[\epsilon_2]$	$\mathsf{V}[\epsilon_2]$	$\mathsf{L}[\epsilon_2]$	$\mathsf{U}[\epsilon_2]$	$\mathsf{E}[\epsilon_3]$	$\mathsf{V}[\epsilon_3]$	$\mathsf{L}[\epsilon_3]$	$\mathsf{U}[\epsilon_3]$
74182	00	.000	.000	.000	.001	.000	.000	.000	.001	.000	.000	.000	.001
74182	01	.005	.000	.000	.030	.062	.015	.000	.349	.062	.015	.000	.349
74182	02	.007	.000	.000	.031	.062	.015	.000	.352	.062	.015	.000	.352
74182	03	.006	.000	.000	.050	.038	.007	.000	.281	.038	.008	.000	.281
74182	04	.003	.000	.000	.061	.013	.001	.000	.126	.013	.001	.000	.126
74L85	00	.004	.000	.000	.015	.004	.000	.000	.008	.004	.000	.000	.008
74L85	01	.013	.000	.000	.064	.084	.009	.000	.344	.084	.009	.000	.344
74L85	02	.013	.000	.000	.062	.084	.009	.000	.343	.084	.009	.000	.344
74181	00	.006	.000	.000	.034	.034	.005	.000	.198	.034	.005	.000	.198
74181	01	.013	.000	.000	.070	.040	.005	.000	.286	.040	.005	.000	.286
74181	02	.016	.000	.000	.087	.047	.004	.000	.265	.047	.004	.000	.265
74181	03	.015	.000	.000	.063	.067	.011	.000	.360	.076	.015	.000	.376
74181	04	.010	.000	.000	.067	.057	.009	.000	.331	.066	.013	.000	.376
74181	05	.004	.000	.000	.045	.045	.008	.000	.326	.056	.012	.000	.375
74181	06	.000	.000	.000	.033	.032	.006	.000	.327	.040	.010	.000	.377
74181	07	.027	.000	.000	.055	.045	.000	.000	.112	.049	.001	.000	.122
74283	00	.013	.000	.000	.054	.049	.007	.000	.282	.050	.007	.000	.282
74283	01	.008	.000	.000	.056	.068	.008	.000	.270	.078	.011	.000	.280
74283	02	.004	.000	.000	.048	.056	.007	.000	.232	.067	.010	.000	.253
74283	03	.001	.000	.000	.032	.044	.006	.000	.210	.056	.010	.000	.253
74283	04	.000	.000	.000	.002	.031	.004	.000	.209	.042	.009	.000	.250
c17	all	.003	.000	.000	.033	.043	.002	.000	.127	.043	.002	.000	.127
c499	all	.002	.000	.000	.011	.006	.000	.000	.171	.005	.000	.000	.254
c880	all	.002	.000	.000	.288	.007	.010	.000	.450	.007	.012	.000	.450
c1355	all	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
c1908	all	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
c2670	all	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
c3540	all	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
c5315	all	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
c6288	all	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000
c7552	all	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000	.000

Table 4.2: Accuracy results of probabilistic model (single faults)

Circuit	PO	E[ϵ_1]	V[ϵ_1]	L[ϵ_1]	U[ϵ_1]	E[ϵ_2]	V[ϵ_2]	L[ϵ_2]	U[ϵ_2]	E[ϵ_3]	V[ϵ_3]	L[ϵ_3]	U[ϵ_3]
74182	00	.000	.000	.000	.001	.000	.000	.000	.001	.000	.000	.000	.001
74182	01	.017	.000	.000	.126	.144	.024	.000	.417	.144	.024	.000	.417
74182	02	.009	.000	.000	.031	.094	.019	.000	.352	.094	.019	.000	.352
74182	03	.019	.001	.000	.160	.057	.013	.000	.362	.057	.013	.000	.362
74182	04	.013	.001	.000	.122	.007	.000	.000	.128	.007	.000	.000	.128
74L85	00	.012	.000	.000	.039	.005	.000	.000	.013	.004	.000	.000	.013
74L85	01	.023	.000	.000	.161	.160	.014	.000	.353	.160	.014	.000	.353
74L85	02	.024	.000	.000	.174	.140	.011	.000	.367	.140	.011	.000	.367
74181	00	.009	.000	.000	.054	.052	.005	.000	.283	.052	.005	.000	.283
74181	01	.020	.000	.000	.072	.064	.008	.000	.336	.064	.007	.000	.336
74181	02	.024	.000	.000	.119	.083	.007	.000	.301	.083	.008	.000	.301
74181	03	.018	.000	.000	.082	.113	.018	.000	.379	.132	.007	.000	.396
74181	04	.016	.000	.000	.162	.074	.012	.000	.477	.087	.017	.000	.508
74181	05	.005	.000	.000	.084	.067	.010	.000	.326	.086	.012	.000	.377
74181	06	.000	.000	.000	.030	.032	.006	.000	.325	.043	.006	.000	.375
74181	07	.032	.000	.000	.055	.056	.001	.000	.120	.046	.001	.000	.128
74283	00	.026	.000	.000	.130	.073	.009	.000	.301	.073	.009	.000	.301
74283	01	.017	.000	.000	.142	.117	.008	.000	.240	.142	.011	.000	.278
74283	02	.013	.000	.000	.126	.119	.010	.000	.318	.140	.015	.000	.346
74283	03	.009	.001	.000	.250	.100	.010	.000	.250	.126	.015	.000	.312
74283	04	.000	.000	.000	.003	.057	.007	.000	.210	.077	.013	.000	.253
c17	all	.030	.004	.000	.218	.047	.002	.000	.125	.046	.002	.000	.125
c499	all	.004	.000	.000	.017	.011	.000	.000	.170	.008	.000	.000	.254
c880	all	.003	.000	.000	.288	.014	.002	.000	.450	.014	.002	.000	.450

Table 4.3: Accuracy results of probabilistic model (double faults)

Circuit	PO	E[ϵ_1]	V[ϵ_1]	L[ϵ_1]	U[ϵ_1]	E[ϵ_2]	V[ϵ_2]	L[ϵ_2]	U[ϵ_2]	E[ϵ_3]	V[ϵ_3]	L[ϵ_3]	U[ϵ_3]
74182	00	.000	.000	.000	.004	.000	.000	.000	.004	.000	.000	.000	.004
74182	01	.017	.001	.000	.150	.074	.022	.000	.448	.074	.022	.000	.448
74182	02	.022	.001	.000	.126	.132	.025	.000	.417	.132	.025	.000	.417
74182	03	.019	.000	.000	.051	.090	.014	.000	.283	.090	.014	.000	.283
74182	04	.023	.002	.000	.187	.036	.004	.000	.188	.036	.004	.000	.188
74L85	00	.018	.000	.000	.072	.005	.000	.000	.015	.005	.000	.000	.015
74L85	01	.040	.001	.000	.204	.188	.009	.000	.394	.188	.009	.000	.394
74L85	02	.058	.004	.000	.246	.190	.016	.000	.414	.190	.016	.000	.414
74181	00	.018	.000	.000	.075	.098	.009	.000	.271	.098	.009	.000	.271
74181	01	.032	.001	.000	.145	.117	.012	.000	.403	.117	.012	.000	.403
74181	02	.036	.001	.000	.152	.116	.008	.000	.384	.116	.008	.000	.384
74181	03	.037	.002	.000	.195	.130	.016	.000	.520	.154	.021	.000	.535
74181	04	.020	.000	.000	.112	.123	.014	.000	.477	.153	.020	.000	.382
74181	05	.005	.000	.000	.057	.124	.015	.000	.357	.160	.023	.000	.382
74181	06	.000	.000	.000	.005	.051	.007	.000	.327	.070	.014	.000	.377
74181	07	.035	.000	.000	.052	.077	.006	.000	.125	.088	.000	.000	.135
74283	00	.039	.001	.000	.352	.133	.015	.000	.351	.133	.015	.000	.351
74283	01	.016	.000	.000	.358	.140	.010	.000	.342	.170	.015	.000	.358
74283	02	.015	.000	.000	.340	.118	.007	.000	.309	.145	.014	.000	.340
74283	03	.004	.000	.000	.264	.104	.007	.000	.210	.137	.013	.000	.264
74283	04	.007	.001	.000	.313	.052	.007	.000	.250	.071	.013	.000	.313
c17	all	.050	.007	.000	.221	.054	.001	.000	.125	.054	.001	.000	.125
c499	all	.006	.000	.000	.030	.017	.001	.000	.171	.012	.001	.000	.255
c880	all	.005	.000	.000	.290	.019	.003	.000	.453	.019	.003	.000	.453

Table 4.4: Accuracy results of probabilistic model (triple faults)

Chapter 5

ANTARES

Chapter 2 introduced the idea of an SFL based diagnosis technique (BACINOL) which is used for fault diagnosis in hardware without requiring any detailed modeling of the components involved. But BACINOL suffers from lower diagnosis quality because of large ambiguity sets. In this chapter, we present ANTARES, an SFL based diagnosis technique, that improves the performance of BACINOL by supplying better estimated g values to break the ambiguity sets efficiently. In the last chapter, we formulated the FNR of the faulty components and developed three different probabilistic models to estimate g value for each of the component in the hardware. This chapter describes the methodology to combine the estimated g values with BARINEL. In addition, this chapter also discusses the different policies to estimate multiple-fault g information.

5.1 ANTARES

Inspired by the application of SFL on hardware and problem of ambiguity sets in the technique, we present a new approach to diagnose the hardware: *Automatic diagNosis of sofTware/hardwARE Systems* (ANTARES). ANTARES breaks the ambiguity sets by supplying better estimated FNR parameter (g) to the SFL. ANTARES uses the EPP models developed in the last chapter to estimate the g values. ANTARES also avoids two complex problems of traditional hardware diagnosis technique (MBD): the complexity of model construction and the computational complexity.

In the next section, we explore the methodology by which g information is provided to ANTARES.

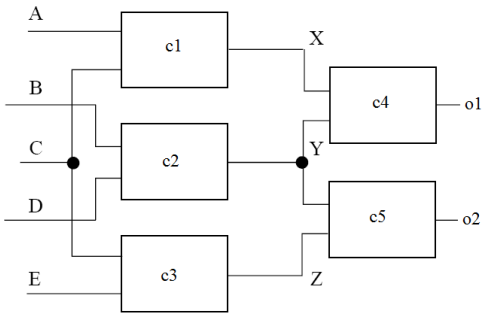

Figure 5.1: Circuit with two principal outputs

5.2 Obtaining g Matrix

In previous chapters, we considered various systems with only one principal output. But a larger system can, in general, have more than one principal output. For each of the faulty components in the system, the probability that the fault can be observed at any of the principal outputs is different, depending upon the topology of the system. Parameter g_c is the probability that the fault at a component c will not be observed at the principal output of the system. For each of the principal outputs, there is a different path from the faulty component to the principal output. An error can be observed with a higher probability at the principal output which is nearer to the faulty component in the system topology. In this case, for each component c, there will be N values of g_c, one for each of the N principal outputs.

Reconsidering the circuit shown in Figure 5.1, the circuit has 2 principal outputs, O_1 and O_2. Suppose, we are interested in finding g information for component c_1. Using the Equation 4.51, we obtain, $g_1 = 0.75$ for output O_1 and $g_1 = 0$ for output O_2.

Therefore, the g value for every component is defined as $g(i, j)$, where i is the principal output index and j is the component index. If the output O_1 has index value 1 and the output O_2 has index value 2, this implies, $g(1, 1) = 0.75$ and $g(2, 1) = 1$. c_1 is not in the subcircuit (cone) of O_2, therefore fault at will never be observed at the O_2.

Using the structural model of the system, g value for each of the components can be precomputed using EPP models. Because of the fact that the g has the different value for each of the principal outputs, the structure of the g information is a 2-dimensional matrix; we define this matrix as the G matrix. If a system has M components and N principal outputs, then G is an $N \times M$ matrix. An element $g(ij)$ from the matrix G corresponds to the g value of the j^{th} component whose behavior will be observed at the i^{th}

principal output of the system.

Considering again the circuit shown in the Figure 5.1, it has five components and two principal outputs. g information is estimated for each component corresponding to every output and will be stored in 2×5 matrix (shown below).

$$G = \begin{pmatrix} 0.75 & 0.75 & 1 & 0.5 & 1 \\ 1 & 0.75 & 0.75 & 1 & 0.5 \end{pmatrix}$$

5.3 Multiple Fault g Model

$Pr(d_k|(A, obs))$ is computed for each minimal diagnosis candidate $d_k \epsilon D$. If there is only one component in the diagnosis candidate, g value can be directly obtained from G matrix. But cardinality of diagnosis candidate can be more than one. For example, if $d_k = \{c_1, c_2\}$, the g value for this d_k will be a composition of g_1 and g_2, represented as g_{12}. The task here is to calculate this composite g value for multiple-fault candidates.

The most straightforward way to estimate the g value for multiple-fault candidate is to use EPP models. However, precomputing the g value for multiple-fault candidates is typically not performed in view of the (theoretical) exponential number of possible multiple-fault combinations, that have to be taken into account. Imagine the number of possible combinations of faulty components required for a system with 500 components and diagnosis candidate cardinality equal to 5. Therefore, the precomputation of g values with this approach is not a good choice.

$g(d_k)$ can be computed using the methodology developed in Section 4.5.2. This section computes the g value of a component in two steps. In the first step, it injects a fault in the component followed by applying system level EPP model. To compute $g(d_k)$, we can use the same steps but this time, faults will be injected in all the components that are a member of d_k. Then system level EPP model will be used to obtain the $g(d_k)$. With this approach, we can compute $g(d_k)$ on the fly and therefore precomputation of g value is not required.

BARINEL computes multiple-fault g values using the single-fault g values. Abreu uses this methodology with the OR model [5] to estimate multiple-fault g values. In this thesis, we also study a *MIN mode*l and *Level based model* for multiple-fault g calculation. All these models are described in the further sections

5.3.1 OR Model

According to OR model, the probability that a test that involves M_f faulted components will still pass it, when each component would yield a pass, i.e., the product of the individual g_j. OR model can be defined as follows

$$g(d_k) = \prod_{c \epsilon d_k} g_c \tag{5.1}$$

But theoretically, the proposition behind OR is not completely correct, it might be possible that some components, that are a part of M_f faulted components, do not pass the test. Error generated by one faulty component c_i can be masked by another faulty component c_j, although both the components c_i and c_j are behaving incorrectly, but no error will be observed at the output of the system.

OR model is basically developed for software, wherein there is no notion of the topology of the system. The g information can be inferred from the topological information of the system. We discussed in the last section that if a faulty component c_i is closer to the principal output as compared to the component c_j, this implies that $g_j > g_i$. Therefore, the topology can give a better estimation of the g values. In the next section, we present the *MIN Model* to estimate the g value for multiple-faults candidates.

5.3.2 MIN Model

MIN model takes the advantage of available structural model of the system to estimate multiple-fault g information. MIN model can be defined as follows

$$g(d_k) = \min_{c \epsilon d_k}(g_c) \tag{5.2}$$

To understand the logic behind MIN model, consider a system S with two faulty components c_l and c_m. All the components here can be treated as a black-box and the structural model of the system is known. To calculate the g value corresponding to both the faulty component (g_{lm}), location of c_l and c_m in the system is considered. This section presents three different configurations of both the faulty components in the system. For each of the configuration, we compute the g_{lm} analytically and compare it with g_{lm} obtained from MIN model.

1. Figure 5.2 shows the first configuration, where component c_l is in the IN set (Section 2.5.1)of output O_1 and therefore the error generated by c_l will be observed at the output O_1. On the other hand, the

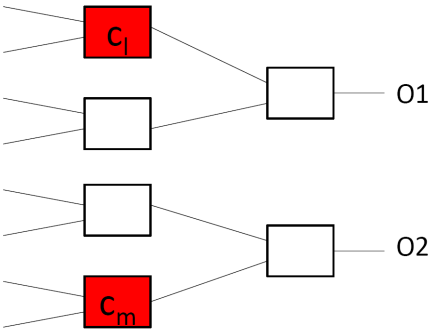

Figure 5.2: First configuration of circuit

component c_m is not in the IN set of O_1 and hence the fault at c_m will not cause any incorrect behavior at O_1. Therefore, g_{lm} will be determined by only g value of component c_l. Hence,

$$g_{lm}(O_1) = g_l$$

g values for c_l and c_m can be calculated using the methodology, described in Section 4.5.2.

$$g_l(O_1) = 1 - \frac{1}{4}$$
$$g_m(O_1) = 1$$

Since $g_l < g_m$:

$$\min(g_l, g_m) = g_l$$

As we can see, that g_{lm} calculated with MIN model is equal to the g_{lm} calculated analytically.

We can also use the OR model as shown below:

$$\begin{aligned} g_{lm}(O_1) &= g_l * g_m \\ g_m(O_1) &= 1 \\ g_{lm}(O_1) &= g_l \end{aligned}$$

For this configuration, OR model also produces correct g value.

2. Configuration presented in Figure 5.3 shows that both the components c_l and c_m are in the IN set of output O_1. An important thing to note

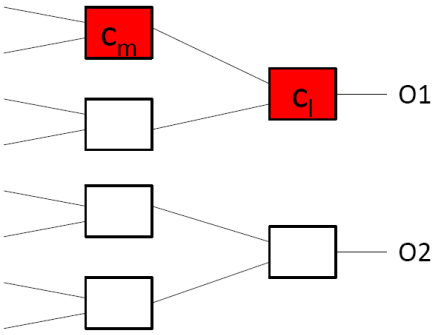

Figure 5.3: Second configuration of circuit

here is that both c_l and c_m are in the same path from the primary input to the principal output.

In the last chapter, we illustrated that a faulty component generates $epp = \frac{1}{2}$ at the output of the component. Faulty component c_m generates an error with $epp = \frac{1}{2}$ and this error propagates to the principal output of the system (O_1). This error propagates till the other faulty component c_l. A faulty component ignores the incoming error signal probabilities (e_i) and generates $epp = \frac{1}{2}$ at it's output [1]. Hence, the error generated by component c_m will not arrive at O_1. Therefore, the incorrect behavior observed at the O_1 will be only because of the fault at c_l. It can hence be concluded that the g_{lm} is equal to g_l and it independent of g_m.

g_{lm} at the output O_1 can be determined as:

$$g_{lm} = g_l$$

The g_{lm} is calculated using the analytical approach. Now, we will calculate the same using MIN model. For example, from Equation 4.51, g_l and g_m can be computed as follows

$$g_l = 1 - \frac{1}{4}$$
$$g_m = 1 - \frac{1}{16}$$

Clearly $g_l < g_m$ and therefore using the MIN model, g_{lm} can be calculated as follows

$$g_{lm} = \min(g_l, g_m) = g_l \tag{5.3}$$

[1]Note that our fault models are SA0 and SA1 only

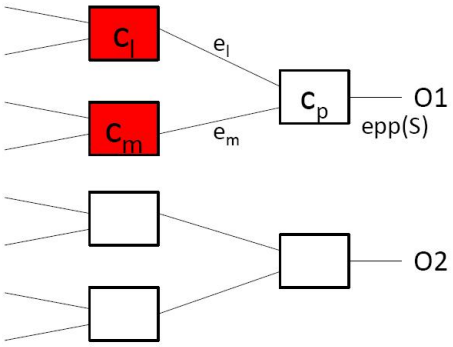

Figure 5.4: Third configuration of circuit

As can be seen that g_{lm} calculated using both the approaches is exactly the same, this shows the correctness of MIN model.

If we have the same configuration as shown in Figure 5.3, the composite g value will be equal to the g value of the component closest to the principal output. In simple words, faulty component closest to the principal output is only the one responsible for the observed incorrect behavior of the system.

If we apply OR model in this configuration, $g_{lm} = g_l * g_m$, but $g_m \neq 1$, hence $g_l * g_m \neq g_l$. This proposition shows that OR model does not produce the correct g information for this configuration.

3. The most interesting configuration is shown in the Figure 5.4. Both the faulty components are in the IN set of output O_1, but they are not in the same path from the primary input to the principal output of the system. Therefore, this configuration does not converge to single-fault g calculation problem directly.

   For component c_p, input error probabilities are e_l and e_m. EPP at the output O_1 of c_p can be given by

$$\begin{aligned} epp(S) &= \frac{e_l + e_m - e_l e_m}{2} \\ &= \frac{e_l(1 - e_m) + e_m}{2} \end{aligned} \tag{5.4}$$

   Since $e_m \geq 0$ and $e_l \geq 0$, this leads to the following propositions:

$$\begin{aligned} epp(S) &\geq \frac{e_l}{2} \\ &\geq \frac{e_m}{2} \end{aligned} \tag{5.5}$$

We now consider the following g equations in terms of error signal probabilities

$$g_l = 1 - \frac{e_l}{2} \tag{5.6}$$

$$g_m = 1 - \frac{e_m}{2} \tag{5.7}$$

$$g_{lm} = 1 - epp(S) \tag{5.8}$$

Equations 5.5 to 5.8 deduce that $g_{lm} \leq g_l$ and $g_{lm} \leq g_m$. These propositions give the following expression

$$g_{lm} \leq min(g_l, g_e) \tag{5.9}$$

Equation 5.9 proves the correctness of the MIN model.

MIN model takes advantage of structural behavior of the system. But Equation 5.9 shows that MIN model still involves some approximation. Therefore, we develop more accurate model that uses the single-fault g values and also exploits the structural model of the system.

5.3.3 Level Based Model

Consider the configuration shown in Figure 5.5. c_l and c_m are two faulty components and they are l and m stages away from the principle output. Assume that c_l generates an $epp(c_l) = e_l$, at the output of the components. Paths from the faulty components to the principal output meet at a component c_p which is p stages away from the principal output. Assume that circuit does not include reconvergent subcircuit.

For this system, we derive the equation for g_{lm} using system level EPP model. Then we express this equation in terms of g_l and g_m. The equation is derived in the following manner,

As we can see in the configuration, c_l is l level away from the O_1 . g_l is given by (see Equation 4.51)

$$g_l = 1 - \frac{e_l}{2^{l-1}} \tag{5.10}$$

Similarly, we can derive g_m

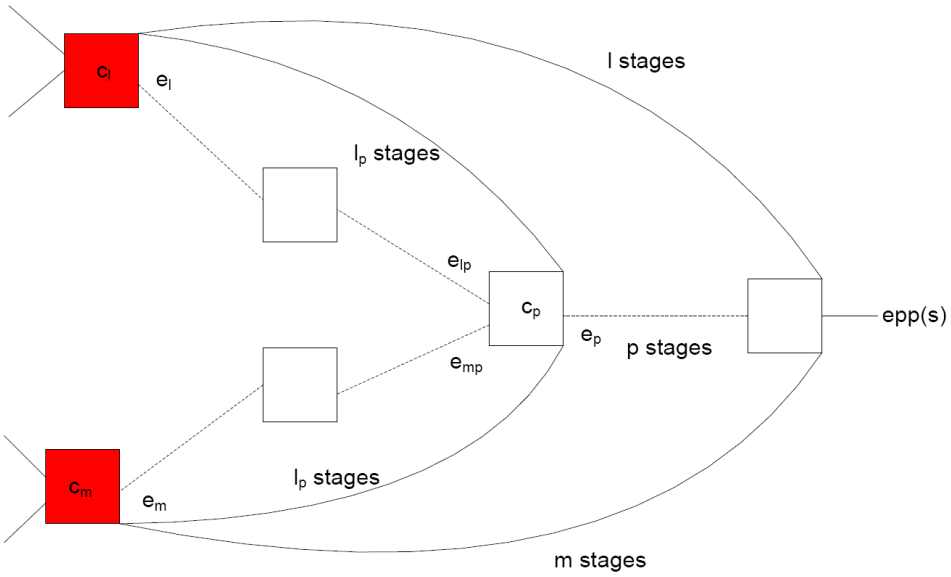

Figure 5.5: General configuration of components

$$g_m = 1 - \frac{e_m}{2^{m-1}} \tag{5.11}$$

The equations derived above are the single-fault g values. Now we consider that both faulty components c_l and c_m are present in the system. Flow of error from these two components meet at the component c_p. Input error probabilities for the c_p can be given as

$$e_{lp} = \frac{e_l}{2^{l_p-1}} \tag{5.12}$$

$$e_{mp} = \frac{e_m}{2^{m_p-1}} \tag{5.13}$$

EPP at the output of c_p is given by

$$e_p = \frac{e_{lp} + e_{mp} - e_{lp}e_{mp}}{2} \tag{5.14}$$

e_p can propagate till O_1 like a single-fault error propagation. EPP at the O_1 is given by

$$epp(S) = \frac{e_p}{2^{p-1}} \tag{5.15}$$

g_{lm} can be calculated using

$$g_{lm} = 1 - epp(S) \tag{5.16}$$

Combining Equations 5.12 - 5.16 results into following expression

$$g_{lm} = 1 - (\frac{e_l}{2^{l_p+p-1}} + \frac{e_m}{2^{m_p+p-1}} - \frac{e_l e_m}{2^{l_p+m_p+p-2}}) \tag{5.17}$$

As can be seen in Figure 5.5, following equations can be derived

$$l = l_p + p \tag{5.18}$$
$$m = m_p + p \tag{5.19}$$

Substituting values of g_l ,g_m, l and m from Equations 5.10 , 5.11 5.18 and 5.19 in Equation 5.17 results into the following expression

$$g_{lm} = g_l + g_m + 2^p(1 - g_l)(1 - g_m) - 1 \tag{5.20}$$

In the previous sections, methodology used by ANTARES to utilize g information has been explained. ANTARES combines the G matrix with spectrum matrix (A, e) to generate a final ranking list. Next section shows an example, where circuit is diagnosed with ANTARES and performance is compared with previous SFL on hardware approach (BACINOL).

5.4 ANTARES Example

Consider the circuit shown in Figure 5.1, assume that c_5 is the faulty component. Spectrum shown in Figure 3.2 can be obtained by giving 4 input vectors to the circuit. G matrix derived in Section 5.2 will be used by ANTARES to break the ambiguity set. ANTARES gives following ranking list:

c_5 (0.5)
c_3 (0.4)
c_2 (0.1)

We saw in the Section 3.1 that ranking list generated by BACINOL for the same circuit had an ambiguity set. This ambiguity set has been broken by the G matrix very efficiently. Faulty component (c_5) has the first rank

without any ambiguity set. This example shows that ANTARES improves the diagnosis quality using external g information.

However, as compared to MBD diagnosis, it is still not a good enough technique because ANTARES does not include double-fault minimal candidate c_3, c_2 in the ranking list. The MBD has more conflicts, therefore it includes double-candidate c_3, c_2 in the final diagnosis.

Therefore, we can conclude that theoretically ANTARES has a better performance than BACINOL but it has lower performance than MBD. To compare the diagnosis quality of ANTARES with BACINOL and MBD, we have to perform a number of experiments. Next chapter summarizes the experimental setup and the results obtained.

Chapter 6

Experimental Results

In this chapter, the experiments and results will be presented which were performed using the methodologies described in the earlier chapters. First, the setup for the experiments is presented. This is followed by a summary of the results and a brief analysis of the obtained results. Results have been presented in terms of impact of following factors on the diagnosis quality: the number of observation, the accuracy in g calculation, the fault-cardinality and the multiple-fault g computation model.

6.1 Objective

There are two main objectives of the experiments that have been performed:

1. To assess the *diagnosis quality* of ANTARES as compared to BACINOL for various systems.

2. To compare the diagnosis quality of ANTARES with the MBD method. For the experimental purpose, GDE is used as an MBD reference method. GDE generates the conflict matrix (A, e) by using detailed modeling information of the components in the system.

Wasted effort is used as the diagnosis quality metric for the system. Wasted effort is estimated with the equations derived in Chapter 3. The methodology used by Wilson [20] to estimate the diagnosis quality of BACINOL is different from the one used in this thesis. Therefore, we do not use the results produced by Wilson for the comparison. We recalculate the diagnosis quality of BACINOL using our statistical approach.

6.2 Circuit, Faults and Observations

The test circuits used in this paper are the well-known ISCAS85 benchmark circuits, such as described by Brglez and Fujiwara (1985) and Hansen et al.(1999) in [8] and [11] .We have also used 4 circuits from the 74-series of integrated circuits (also described in Hansen et al.,1999). These circuits have been chosen because of their availability and usage in other similar experiments. A summary of the main topological properties of these circuits has been presented in the Table 2.2.3.

For these circuits, it was chosen to inject the faults randomly to approximate a real-world scenario. The experiments have been carried out for single as well as multiple faults.

The observation vectors have also been generated randomly after completion of fault injection. Because of the random input vectors, passed and failed spectrum rows are obtained and this spectrum matrix is then used for the computation of the diagnosis. The main advantage of the random input vector is that it does not require the behavioral model of the System-Under-Test (SUT).

6.3 Implementation

To implement ANTARES, BARINEL code has been used to compute the diagnosis of hardware. However, instead of instrumenting a program to create a spectrum, a module has been written in C to calculate the spectra from the ISCAS85 circuits. The module is driven by the input parameters for one or more experiments. The module generates the spectra of the circuit; the input parameters required to produce the same are: the name of the circuit to use, the number of faults to be injected (M_f) and the number of observations (n_{obs}). The module works by emulating the circuit as a tree of interconnect nodes. Spectrum matrix is generated by the following two steps

1. IN set is computed for each of the principal outputs of the system by traversing the circuit from the principal output to the primary input recursively. The IN set for each principal output gives the rows of the matrix A.

2. Faults are injected in the nodes and the random input vector is applied to the input nodes. Each node calculates both its nominal value and the actual value (i.e., taking the fault into account). The output values are then compared with the nominal values to check if the system exhibits a failure for the current input vector. If it does, a spectrum line

is written for each output. If the actual output value differs from the nominal output value, the spectrum is marked as failed ($-$), otherwise, it is marked as successful ($+$).

The above steps are executed n_{obs} times. The final matrix corresponding to all the observations is recorded into the file *A.txt*.

To estimate the g information for each of the component, another module is written in C. The input parameters for this module are the name of circuit and the number of observations (n_{obs}). The module computes g value for each of the component using all EPP models developed in the Chapter 4. In addition, Monte Carlo (MC) simulations have been done to compute the most accurate g values. The g information is stored in the form of a 2-dimensional matrix. This matrix is replicated n_{obs} times and it results in $N \times n_{obs}$ rows in the g matrix and M columns. Here, N is the number of principal outputs and M is the number components in the system. This final g matrix is written in a file named *g.txt*.

Spectrum matrix file *A.txt* is processed with STACCATO to generate the minimal diagnosis candidates. Diagnosis candidates are stored in a file, named *hs*. Finally, BARINEL reads *A.txt*, *g.txt* and *hs* to compute the posterior probability for each of the candidates. This posterior probability is then used to compute the diagnosis (D). The diagnosis D is further mapped to the 1-D ranking (R) by using $-d$ option of BARINEL. R is recorded in a file called *rank.txt*. A program has been written in C which traverses the raking-list stored in the file *rank.txt* and computes W to diagnose the faults.

6.4 Experimental Setup

To assess the performance of the ANTARES, experiments have been performed to analyze the relationship between the number of observations n_{obs} and the diagnosis quality (W). For every fault setup, 50 observation vectors are generated, which are used incrementally to compute the diagnosis. To reduce the variance in W and to get more accurate results, experiments are repeated over 200 fault sets. The experiments are performed for $M_f = 1, 2$ and 3 injected faults, so that the influence of the number of faults in the system can be determined as well.

ANTARES uses the black-box EPP models to compute the g values. The accuracy results presented in Section 4.6 show that the black-box EPP is the least accurate model as compared to the other two EPP models. To investigate the effect of the accuracy in the g value calculation on W, other two EPP models are also used to compute the g information. The g value has also been calculated using Monte Carlo simulations and this g information

can be considered as the best possible g estimation. Therefore, it is expected that the ANTARES will give the best diagnosis quality by using the g values computed using the Monte Carlo simulations.

Multiple-fault g values are calculated by applying OR model on single-fault g value. MIN model and Level-based model proposed in Section 5.3.2 and 5.3.3 are also used to compute the multiple-fault g values. Hence, all these three models are compared experimentally. We also calculate the multiple-fault g values directly with EPP model as well as with Monte Carlo simulations. This experiment aims to validate the correctness of MIN, OR and Level-based model.

To analyze the effect of the g information on the diagnosis quality, we calculate W without supplying any g information to ANTARES. ANTARES without g is same as BACINOL. Diagnosis quality of MBD is computed using GDE generated conflict matrix. Since we do not have any information about the number of observations used to derive those conflict matrices, we cannot directly compare the diagnosis quality of GDE with the diagnosis quality of ANTARES in terms of number of observations.

6.5 Influence of The Number of Observations

In Figures 6.1 - 6.3, W is plotted against n_{obs} for $74XXX$ circuits. Average W are computed for ANTARES (for all EPP models) and for BACINOL. As expected, both the curves show a decreasing trend with the increase in number of observations. The curves become asymptotic after a sufficient number of observations (more than 20 for our experiments).

It can be clearly seen that ANTARES with the deterministic EPP model has a better diagnosis quality as compared to the other methods. Another observation that can also be made from the plots is that known PDF EPP model and black-box EPP model give a similar kind of performance boost to the ANTARES, with exception of circuit 74283. The reason for this is the presence of a majority of AND, OR, NAND, NOR and inverter gates for most of the $74XXX$ circuits. Since we know that both the known PDF EPP model and the black-box EPP model are same for these gates, the performance of ANTARES is same with these two models. Therefore, both of these models improve the diagnosis quality of the ANTARES by the same amount. In circuit 74283, XOR gate is also present and for XOR gate the known PDF EPP model is different from the black-box EPP model. For XOR gate, black-box EPP model is less accurate than the other EPP models; therefore it gives the least performance improvement to ANTARES.

Hence, we can conclude from the plots that inclusion of external g information enhances the performance of ANTARES. ANTARES has much better

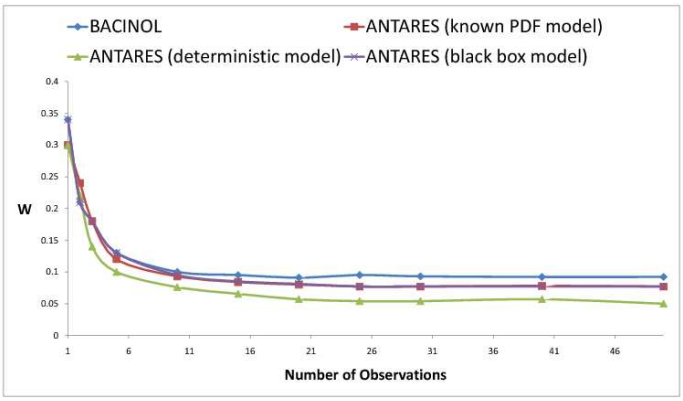

Figure 6.1: Plot of W vs. Number of Observations for 74182 circuit. W is computed for single-fault.

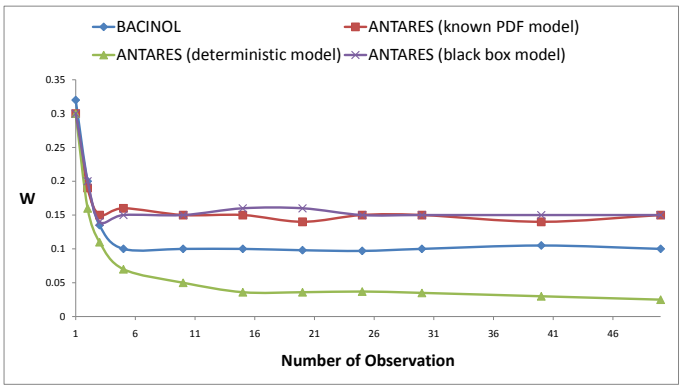

Figure 6.2: Plot of W vs. Number of Observations for 74283 circuit. W is computed for single-fault.

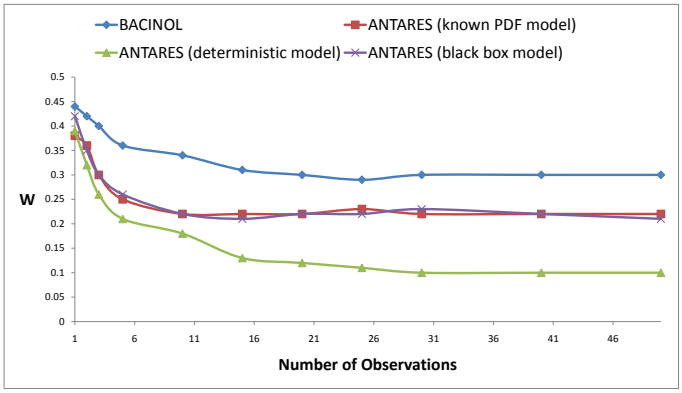

Figure 6.3: Plot of W vs. Number of Observations for 74L85 circuit. W is computed for single-fault.

diagnosis quality than BACINOL, if g is estimated using the deterministic model. This observation implies that the accurate estimation of g is an important factor to ANTARES. In the next section, we analyze the diagnosis quality of ANTARES by analyzing the accuracy in the g estimation.

6.6 Influence of Accuracy in g calculation

Figure 6.4 compares the diagnosis quality of the ANTARES for various g values; each of the g value is computed using a different EPP model. For all the W computation $n_{obs} = 50$, therefore W is calculated at the asymptotes of the previous plots. Diagnosis quality of the BACINOL is also compared with the ANTARES. Results clearly show that for all the circuits except 74283 and 74181, ANTARES has better diagnosis quality. However, for 74283, ANTARES has a better diagnosis quality, if g value is computed using the deterministic EPP model. These results conclude that the better g estimation provides a better diagnosis quality to ANTARES.

To understand the effect of the accuracy of g estimation on the diagnosis quality, reconsider Equation 3.4 derived in the Section 3.2 to estimate W:

$$W = \frac{R_{as} + \frac{C_{as}}{C_{as}+1}.S_{as} - C}{M - M_f}$$

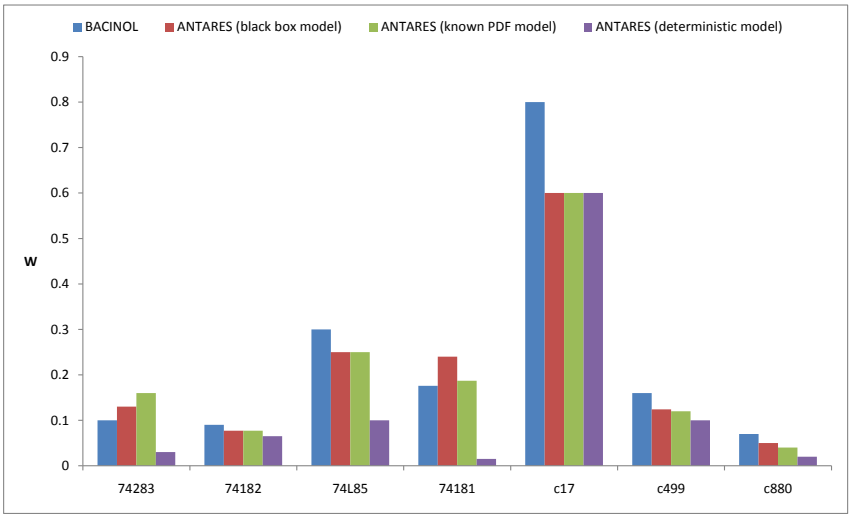

Figure 6.4: Diagnosis quality of ANTARES (with all EPP model) and BACINOL for different circuit with single-fault. Average over 200 fault sets, 50 observations

W is directly proportional to the rank of the ambiguity set R_{as} and the size of the ambiguity set S_{as}. The term S_{as} is multiplied with a term which is less than one, therefore as compared to R_{as}, S_{as} has less impact on the value of W. It is expected that the size of the ambiguity set reduces if a better estimated g values are provided. However, the impact of external g values on R_{as} is not known. Table 6.1 summarizes R_{as} and S_{as} values for ANTARES and BACINOL.

It can be clearly seen that ANTARES breaks the ambiguity sets successfully with all the EPP models. However, known PDF EPP model and black-box EPP model increase the R_{as} whereas with the deterministic EPP model, R_{as} either improves or stays the same. Therefore, ANTARES with the deterministic EPP model gives the best diagnosis quality as compared to the other two probabilistic models.

By observing Table 6.1 closely, 2 questions can be raised:

1. Why does the R_{as} increase when S_{as} goes down?
2. Normally, R_{as} increases for all the circuits with the reduction of S_{as} for the known PDF and the black-box EPP models. But then why does

EPP Model	Circuit									
	74283		**74L85**		**74182**		**74181**		**c499**	
	R_{as}	S_{as}	R_{as}	S_{as}	R_{as}	S_{as}	R_{as}	S_{as}	R_{as}	S_{as}
BACINOL	1.5	5	3.1	15	0.1	4	10.5	4.3	15	38
ANTARES (deterministic model)	0.84	1.6	3.2	1.3	0.1	1.5	0.7	1.6	15	10.1
ANTARES (known PDF model)	4	1.8	7.2	2.44	0.66	2.6	13.4	2	23	7
ANTARES (black-box model)	5	1.8	7.2	2.6	0.63	2.6	17	2.1	23	6.8

Table 6.1: Average rank and size of ambiguity set for single faults. Averages over 200 fault sets and 50 observations are used.

Circuit	ANTARES (deterministic EPP model)	ANTARES (Monte Carlo simulations)
74283	0.034	0.035
74L85	0.105	0.100
74182	0.061	0.060
74181	0.014	0.016

Table 6.2: Wasted effort values for ANTARES with g value derived from deterministic model and MC simulations to detect single-fault. Averages over 200 fault sets, 50 observations are used.

> the ANTARES has a slightly better diagnosis quality as compared with the BACINOL, except for 74283 and 74181?

To answer the first question, let's consider a circuit of 50 components with only one faulty component. If 15 out of these 50 components, including the faulty one have the same rank (first rank) in the ranking list, then these 15 components form an ambiguity set. After giving g information for these components, this ambiguity set eventually breaks and each component of the set gets an individual ranking. It is very unlikely that the faulty component will get the first rank instead, the faulty component will get a rank lower than 1. This explains that as the size of ambiguity set goes down, the rank of the ambiguity set increases.

Circuit 74L85, c499 and c880 have large ambiguity sets, relative to the number of components in the circuit. The g information calculated with the two probabilistic models reduces the size of the ambiguity sets significantly. Therefore, despite of the higher rank of the ambiguity set, the diagnosis quality increases as compared to the BACINOL. But circuits 74283 and 74181 have small size of ambiguity sets and therefore reduction in the size of ambiguity set does not have enough impact on the diagnosis quality.

This observation motivates us to study ANTARES with the best possible g

Circuit	BACINOL	ANTARES (deterministic EPP model)	ANTARES (known PDF EPP model)	ANTARES (black-box EPP model)	GDE
74283	0.096	0.034	0.134	0.155	0.050
74L85	0.320	0.105	0.250	0.250	0.060
74182	0.090	0.061	0.0.080	0.080	0.025
74181	0.176	0.014	0.187	0.247	0.03
c17	0.8	0.6	0.6	0.6	NA
c499	0.164	0.092	0.124	0.129	0.0037
c880	0.065	0.021	0.042	0.052	0.006

Table 6.3: Wasted effort values for single-fault. Averages over 200 fault sets and 50 observations are used.

estimation. Now, we calculate g value using the MC simulations. Since MC simulations take very long time for the bigger circuits, so we use only 74XXX for these experiments. Results shown in Table 6.2 imply that ANTARES with g value calculated using MC simulations has the same diagnosis quality as ANTARES with deterministic model g value. This clearly shows that the error in g value calculation (Table 4.2) makes no difference in the diagnosis quality of the ANTARES. Hence, the g value calculated by the deterministic model can be assumed as the best possible estimation.

6.7 Influence of Fault Cardinality

If the number of faults in a system increase, W also increases. Multiple-fault diagnosis can be understood in terms of the single-fault diagnosis. If a system has multiple faulty components, we inspect the system and detect the first fault. After finding the faulty component, we fix that component and search the next faulty one. Hence, the diagnosis effort required to detect all the faulty components in the system is proportional to the diagnosis effort required to find a single-fault. Therefore, the diagnosis effort increase as the fault-cardinality increases. Results for multiple-fault components presented in Tables 6.3 - 6.5 also support this theory.

From the tables it can be observed that the diagnosis quality of the GDE is far better than ANTARES and BACINOL. However, if most accurate g values are given to ANTARES, it is comparable to GDE. As we mentioned earlier, we do not have any information about the number of observations used to derive the GDE conflicts, therefore we cannot further comment about GDE and ANTARES comparison.

Circuit	BACINOL	ANTARES (deterministic EPP model)	ANTARES (known PDF EPP model)	ANTARES (black-box EPP model)	GDE
74283	0.303	0.210	0.343	0.355	0.210
74L85	0.550	0.483	0.545	0.545	0.144
74182	0.275	0.201	0.245	0.250	0.140
74181	0.314	0.230	0.389	0.502	0.110
c17	0.8	0.575	0.642	0.6	NA
c499	0.182	0.123	0.153	0.164	0.01

Table 6.4: Wasted effort values for double fault. Averages over 200 fault sets, 50 observations are used.

Circuit	BACINOL	ANTARES (deterministic EPP model)	ANTARES (known PDF EPP model)	ANTARES (black-box EPP model)	GDE
74283	0.447	0.375	0.515	0.516	0.410
74L85	0.671	0.610	0.675	0.675	0.195
74182	0.365	0.303	0.350	0.350	0.203
74181	0.48	0.46	0.572	0.608	0.287
c17	0.8	0.575	0.735	0.735	NA
c499	0.203	0.158	0.174	0.182	0.02

Table 6.5: Wasted effort values for triple fault. Averages over 200 fault sets, 50 observations are used.

6.8 Influence of Multiple Fault g Calculation Model

By default ANTARES uses the OR model to compute the multiple-fault g values using the single-fault g values. In Section 5.3.1 we discussed that theoretically OR model is not a very accurate model. In Section 5.3.2 we developed a MIN model that uses topological information to compute multiple-fault g values. Later, in Section 5.3.3 we developed Level-based model which is theoretically the most accurate model as compared to the other two models. To evaluate the impact of these models on the diagnosis quality we run ANTARES with the MIN model and the Level-based model for all 74XXX circuits and some ISCAS circuits (c17 and 499). Results for all the circuits show that the diagnosis quality is same for all the models. Further, we use the MC simulation to compute the multiple-fault g values, but again the diagnosis quality is same as it is with OR model. These observations imply that although theoretically OR model is not completely accurate, experimentally it is accurate enough to diagnose the system efficiently.

Recalling the Bayes' rule used in the BARINEL to compute the posterior probability (Section 2.2.3), a prior probability p is allocated to all the diagnosis candidates in the beginning of the diagnosis process. Value of the prior probability depends on the cardinality of the diagnosis candidates. For a single-fault candidates $p \approx 0.1$ and for a double-fault candidates $p \approx 0.01$. Hence, the prior probability reduces exponentially as the fault-cardinality increases. Therefore, most of the time, the single-fault candidates have higher posterior probability as compared to the multiple-fault candidates, because of which single-fault candidates get higher rank in the ranking-list. Since multiple-fault candidates start with a very low prior probability, it is potentially unlikely that multiple-fault g information can increase their posterior probability. Therefore, the role of the multiple-fault g information does not seem to be very critical to the BARINEL. So, from diagnosis point of view, it does not make a big difference, if we use OR model or MIN model or Level-based model or even MC simulations to compute multiple-fault g information. This reasoning explains the observation we made earlier that why the multiple-fault g models, better than the OR model (theoretically), cannot improve the diagnosis quality.

As far as the single-fault diagnosis is concerned, it is independent of multiple-fault g computation model. During the experiments, we observed that if we remove the function that computes the multiple-fault g values, it does not affect the diagnosis quality of ANTARES.

Chapter 7

Conclusions

A spectrum-based hardware diagnosis technique named ANTARES has been proposed in the thesis,. To enhance the performance of ANTARES, FNR parameter g is used to estimate the posterior probability more accurately. Three different EPP models are derived for the better estimation of g values. GDE generated conflict matrices are used to compute diagnosis quality of MBD. Statistical models have been developed and utilized to compute the diagnosis quality and obtain the performance metrics for all the diagnosis methods.

This chapter presents the conclusion that can be drawn from the results of the experiments performed for various diagnosis methods. Subsequently, some suggestions are made for future research opportunities in this area.

7.1 Conclusions

As mentioned in the introduction chapter, the goal of this thesis is to study the feasibility of spectrum-based hardware diagnosis techniques and enhance the performance of the technique by providing better FNR parameter value g. Here, we give an overview of the conclusions drawn while achieving the goal.

- By comparing the performance of ANTARES with the existing spectrum-based hardware diagnosis method BACINOL, it can be concluded that the external g information used by ANTARES significantly improves diagnosis quality by eliminating the ambiguity sets from the diagnosis.
- It is observed that the performance of ANTARES is significantly better as compared to BACINOL while slightly lower than MBD when the g value is computed with the deterministic EPP model. This observation

implies that the diagnosis quality of ANTARES is proportional to accuracy in g value estimation.

- We approximated the deterministic EPP model by ignoring the suppression in the composition of EPP model. However, results show that the g values calculated with the Monte Carlo simulations give the same amount of improvement to ANTARES, as does the deterministic EPP model. Therefore, we can conclude that the approximation that was made earlier does not affect the diagnosis quality of ANTARES and the deterministic EPP model can be considered as the most accurate EPP model possible for ANTARES.

- The black-box EPP model is considered to be the appropriate choice for our purpose because this EPP model does not exploit any kind of behavioral model of the components in the hardware. However, diagnosis quality of ANTARES with black-box EPP model is much lower than the GDE. At the same time, however ANTARES is better than the BACINOL. Lower diagnosis quality of black-box EPP model is due to the non-uniform distribution of logic gates in benchmark circuit (number of AND and OR gates are greater than number of XOR and NXOR gates). The black-box EPP model assumes a real-world system where the components (logic gates) are uniformly distributed over the hardware. Therefore, we expect that the black-box EPP model will have much better diagnosis quality for such real-world systems.

- For most of the ISCAS/74XXX benchmark circuits, known PDF EPP models and black-box EPP models give the same amount of improvement to the diagnosis quality of ANTARES because of the presence of large number of AND, OR, NOR, NAND and INV gates in these benchmark circuits. For these gates both probabilistic models behave exactly the same.

- Results clearly show that the multiple-fault g calculation models developed in this thesis produce the same diagnosis quality as the OR model. Low posterior probability for multiple-fault candidates as compared to high posterior probability for single-fault candidate is the primary reason behind this observation. Therefore, OR model is sufficient to compute multiple-fault g information. [1]

Since we do not have any information about the number of observations used by GDE to generate the conflict matrices, we cannot draw any conclusion about the GDE results obtained. But still, we can say that GDE offers better performance than ANTARES, at least for ISCAS/74XXX benchmark circuits. However, ANTARES with deterministic EPP model has very

[1] Assuming the random observations. If we supply MFMC observations this may not hold true.

good diagnosis quality, but the EPP model uses the behavioral model of the components and this contradicts our goal of avoiding the modeling of components. Due to the lack of gate randomness in benchmark circuits, the black-box EPP model does not give a high diagnosis quality as expected. Therefore, we expect that for real world systems where components may be more randomly distributed over the system, black-box EPP model will have significantly better diagnosis quality, without any modeling of the components.

7.2 Future Work

Given the limited time for this MSc thesis work, a number of topics remain for future work. The future work includes the following directions:

- During this work, we found that the accuracy of the EPP model to be a very critical parameter for ANTARES. We have the deterministic model as the most accurate model, but this model exploits component modeling. On the other hand, the black-box EPP model that does not require such modeling is not sufficiently accurate for the system with lower randomness in terms of components. If we have more information about the number of the components that have specific behavior in the system, then a more accurate EPP model can be derived. For example, if we know how many ANDs, ORs any other logic gates are in the circuit, accuracy of EPP model will increase. The impact of this new EPP model on diagnosis quality of ANTARES can be investigated experimentally.
- More information related to generation of GDE matrices (number of observations, which outputs are used to observe the system behavior), g values can also be given to GDE. It will be very interesting to observe the performance of GDE with external g information.
- In this thesis, we performed experiments with fault-cardinality of three. Experiments can be done with higher fault-cardinality.
- More experimental results can be obtained for the rest of the ISCAS benchmark circuits. Since larger circuits have high execution time, program level optimizations can be done to reduce the execution time.

Bibliography

[1] Rui Abreu. *Spectrum-based Fault Localization*. PhD thesis, Delft University of Technology, 2009.

[2] Rui Abreu and Arjan J.C. Van Gemund. Prioritized minimal hitting set computation in model-based diagnosis.

[3] Rui Abreu and Arjan J. C. van Gemund. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In *SARA*, 2009.

[4] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. An evaluation of similarity coefficients for software fault localization. In *PRDC*, pages 39–46, 2006.

[5] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. An observation-based model for fault localization. In *WODA*, pages 64–70, 2008.

[6] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. A new bayesian approach to multiple intermittent fault diagnosis. In *IJCAI*, pages 653–658, 2009.

[7] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. Spectrum-based multiple fault localization. In *ASE*, pages 88–99, 2009.

[8] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran. In *IEEE International Symposium on Circuits and Systems*, 1989.

[9] Johan de Kleer. Diagnosing multiple persistent and intermittent faults. In *IJCAI*, pages 733–738, 2009.

[10] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artif. Intell.*, 32(1):97–130, 1987.

[11] Mark C. Hansen, Hakan Yalcin, and John P. Hayes. Unveiling the iscas-85 benchmarks: A case study in reverse engineering. *IEEE Design & Test of Computers*, 16(3):72–80, 1999.

[12] Pablo H. Ibargengoytia, L. Enrique Sucar, and Eduardo Morales. A probabilistic model approach for fault diagnosis. In *IN: ELEVENTH INTERNATIONAL WORKSHOP ON PRINCIPLES OF DIAGNOSIS*, pages 79–86, 2000.

[13] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[14] James A. Jones, Mary Jean Harrold, and John Stasko. Visualization of test information to assist fault localization. In *In Proceedings of the 24th International Conference on Software Engineering*, pages 467–477, 2002.

[15] J.De Kleer. Getting the probabilities right for the measurement selection. In *DX*, 2006.

[16] Nasir Mohyuddin, Ehsan Pakbaznia, and Massoud Pedram. Probabilistic error propagation in logic circuits using the boolean difference calculus. In *ICCD*, pages 7–13, 2008.

[17] Kenneth P. Parker and Edward J. Mccluskey. Probabilistic treatment of general combinational networks. *IEEE Transactions on Computers*, 24:668–670, 1975.

[18] Jurryt Pietersma and Arjan J. C. van Gemund. Benefits and costs of model-based fault diagnosis for semiconductor manufacturing equipments. In *Proc. of 17th International Symposium on Systems Engineering (INCOSE 2007)*, 2007.

[19] Raymond Reiter. A theory of diagnosis from first principles. *Artif. Intell.*, 32(1):57–95, 1987.

[20] Michel Wilson. Bacinol: Bayesian circuit analysis. MSc thesis, Delft University of Technology, 2010.

[21] Peter Zoeteweij, Jurryt Pietersma, Rui Abreu, Alexander Feldman, and Arjan J. C. van Gemund. Automated fault diagnosis in embedded systems. In *SSIRI*, pages 103–110, 2008.