University of California
Santa Cruz

Carafe: An Inductive Fault Analysis Tool For CMOS VLSI Circuits

A thesis submitted in partial satisfaction
of the requirements for the degree of

Master of Science

in

Computer Engineering

by

Alvin Lun-Knep Jee

June 1991

The thesis of Alvin Lun-Knep Jee is
approved:

F. Joel Ferguson

Kevin Karplus

Tracy Larrabee

Dean of Graduate Studies and Research

Copyright © by
Alvin Lun-Knep Jee
1991

Contents

Abstract **vii**

Acknowledgements **viii**

1. Introduction **1**

1.1 The Carafe Project 1

1.2 Thesis Organization 2

2. Background **3**

2.1 The Three Stages of IC Testing 3

2.2 Definitions 4

2.3 Fault Models 6

2.3.1 Stuck-at Fault Model 7

2.3.2 Bridge Fault Models 8

2.4 Inductive Fault Analysis 9

3. Carafe **13**

3.1 Carafe Defect Model 13

3.2 Locating the Faults 13

3.2.1 Bridge Faults 13

3.2.2 Break Faults 15

3.3 Reducing the Number of Reported Faults 16

3.4 Modeling the Faults for Simulation 17

3.5 Ranking the Faults 18

3.6 Implementation Details 19

3.6.1 Circuit Representation 19

3.6.2 Circuit Extraction 20

3.6.3 Bridge Fault Extraction 21

3.6.4 Break Fault Extraction 21

3.6.5 Sensitive Area Calculation 21

4. Results **24**

4.1 MCNC Standard Cells 25

4.2 Defect Coverages 27

5. Current Limitations of Carafe **30**

5.1 Break Faults 30

5.2 Analyzing Large Circuits 30

6. Applications of Carafe **32**

6.1 Grading Test Sets for Defect Coverage 32

6.2 Physical Design for Testability 32

6.3 Design for Manufacturability 33

7. Conclusions and Further Research **34**

References **36**

List of Figures

2.1 Typical Integrated Circuit Testing Stages. 3

2.2 Relationship between spot defects, circuit faults, and behavioral faults. . . . 6

2.3 1-bit full adder descriptions (a) Logical (b) Transistor level. 7

2.4 Example of a defect causing a bridge fault. 11

2.5 Example of a defect causing a break fault. 12

3.1 Bridge fault type 1. 14

3.2 Bridge fault type 2. 15

3.3 Break fault type 1 and an inconsequential break fault. 15

3.4 Modeling bridge and break faults as extra transistors. 18

3.5 Sensitive area for a break fault between point A and B. 19

3.6 Sensitive area for a break fault with a horizontal length. 22

4.1 Plot of the bridge fault sensitive areas for the C7552 ISCAS'85 circuit sorted from highest to lowest. 28

4.2 Plot of the cumulative sorted bridge fault sensitive areas for the C7552 ISCAS'85 circuit. 29

List of Tables

2.1 Percent of defect coverage required for various combinations of yield and desired defect level. 10

2.2 Percentage of circuit faults caused by spot defects. 12

4.1 Bridge faults for ISCAS'85 benchmark circuits. 24

4.2 Bridge faults to power and to ground 25

4.3 Bridge faults for logic gates broken down by effect. 26

4.4 Percentage of defects causing bridge faults broken down by effect. 27

Carafe: An Inductive Fault Analysis Tool For CMOS VLSI Circuits

Alvin Lun-Knep Jee

ABSTRACT

Traditional fault models for testing CMOS VLSI circuits do not take into account the actual mechanisms that precipitate faults. As a result, the failure modes of a circuit as predicted by these fault models may not reflect the realistic failure modes of the circuit.

This thesis reports on the Carafe software which determines the realistic bridge faults of a CMOS circuit based on its layout. Each fault found by Carafe is assigned a relative probability based on the geometry of the fault site and defect distributions of the fabrication process. Carafe improves upon previous software in that it is easier to use, more robust, and more time and memory efficient so that larger circuits can be analyzed.

Keywords: Fault Models, VLSI Testing, CMOS, Inductive Fault Analysis

Acknowledgements

I would like to thank my advisor, Dr. F. Joel Ferguson, for introducing me to the project and being tolerant of the many schedule slips.

A number of fellow classmates helped with parts of the Carafe program. George Riusaki imported the basic corner-stitched-tile routines from the Berkeley Magic layout tool. David Prather and David Staepalere provided much appreciated help with the somewhat nasty "features" of using X and motif to create a graphical user interface for Carafe.

This research was supported in part by the Semiconductor Research Corporation Grant 90-DJ-141. Krzysztof Koźmiński of the Microelectronics Center of North Carolina supplied their standard cell implementations of the ISCAS'85 benchmark circuits.

1. Introduction

Studies have shown that the traditional method of testing for manufacturing defects in integrated circuits (ICs) may not be effective enough for current testing goals. The traditional method of generating tests for ICs assumes that a defect will always manifest itself as a signal line stuck at a specific logical value. In most CMOS technologies used today, a defect causes a short or open that may not behave as a signal line stuck at a logic value [GCV80][SMF85]. The traditional testing method does not consider the faulty behavior of a circuit and thus may not identify enough defective circuits.

The problem is that the traditional method does not consider the actual faults that can occur. In order to generate effective tests, we must first determine the faults that can occur. Inductive fault analysis (IFA) [Fer87] was developed to do just that. IFA uses the layout of a circuit along with certain defect parameters to determine exactly what alterations occur to the circuit as a result of defects. Tests can be generated to target these alterations and detect a higher percentage of defective ICs.

1.1 The Carafe Project

The IFA process was demonstrated by the development of the program FXT. FXT was meant to be primarily a research tool to establish the feasibility of the IFA process and was not intended for general use. Carafe is a program that implements the defect simulation and defect-to-fault translation parts of IFA as easy-to-use software for general distribution.

FXT showed that over 99% of the faults caused by small defects were either shorts or opens. For this reason, Carafe is designed to identify the possible short (bridge) and open (break) faults that can occur in a circuit based on its layout. Carafe reports these faults as extra transistors inserted into the extracted transistor netlist of the circuit. With these extra transistors, each fault may be simulated to determine the effect of the fault on the circuit as a whole. Carafe automatically generates scripts for fault simulation that can be used directly with the switch-level simulator COSMOS [BBB+87].

For each fault found by Carafe, the relative likelihood of occurrence is calculated and reported. This likelihood is made relative to all of the other faults that were found in the circuit and indicates the relative importance of detecting the fault. If a specific fault is very likely to occur, it is more important to detect it and warrants more expense to derive a test to detect it than if it were less likely to occur. This metric may also be used to aid in determining how to modify the layout of the circuit to make faults easier to detect or less likely to occur.

1.2 Thesis Organization

Chapter 2 reviews some of the concepts of testing ICs and describes the most prevalent fault models used. The inductive fault analysis procedure, which serves the basis for the Carafe program, is outlined.

Chapter 3 describes the scope and the features of the Carafe program. Carafe uses inductive fault analysis to provide information that can aid in determining how a defective circuit will behave and identify areas that are susceptible to defects. This chapter also contains a few details of the inner workings.

Chapter 4 presents experimental results and observations. Carafe was used to analyze the MCNC standard cell implementation of the ISCAS'85 benchmark circuits. The performance of Carafe and the results of analyzing the circuits are presented.

Chapter 5 is a brief discussion of Carafe performance and limitations.

Chapter 6 describes applications of the Carafe program.

The conclusion of this thesis is in Chapter 7. Here, the major points of the thesis are summarized. Other experiments using Carafe are proposed for future research.

2. Background

This chapter provides a review of IC testing. The first section describes the different stages of the IC testing process. The next section defines the terms used throughout the rest of this thesis. The third section describes the various fault models used to generate tests for ICs. The fourth section discusses the need to test for defects rather than faults that can occur in a circuit and describes a systematic process to determine the faults that arise from defects.

2.1 The Three Stages of IC Testing

Testing an IC usually occurs in multiple stages. Figure 2.1 shows the three typical stages of IC testing. Untested ICs enter the figure at the left and move right through each stage of the test. The ICs found to be bad are removed from further testing while the ICs emerging from the last stage are considered fully functioning. The cost of testing each IC becomes progressively larger in each stage; therefore, it would be best to detect as many defective ICs as possible in the early stages of the test.

The first stage in the test is the gross test. Gross testing is usually nothing more than applying power to the chip, applying a few test inputs to the IC and watching the outputs to see if they do anything at all. If the chip responds to clock signals and the outputs take on valid logic values, it is passed to the next stage.

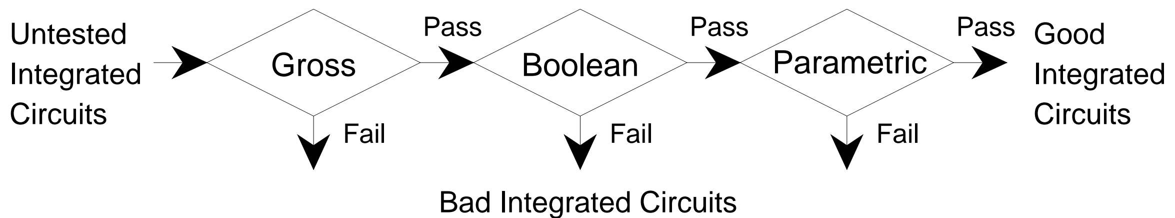

Figure 2.1: Typical Integrated Circuit Testing Stages.

A Boolean test is applied to the ICs that pass the gross test. This stage is sometimes referred to as DC testing since the speeds at which the input patterns are applied to the ICs are much slower than the speeds that the ICs were designed for and the detailed timing of the inputs and outputs are not considered. This step checks the functionality of the chip either by exercising the functions of the IC or, more commonly, by exercising each logical gate within the IC regardless of the overall function of the IC. This stage of the test has been given a great deal of attention in attempts to ensure that very few defective ICs move on to the next stage.

The last stage of testing checks the parametric aspects of the ICs. Parametric testing is also called AC testing since time-dependent aspects of the ICs are tested. Characteristics such as rise and fall times of the ICs outputs must be tested to ensure that they are within the design specifications. The amount of current that the IC uses during normal operation is also checked to make sure it is within the specifications.

ICs passing the last stage of the test are considered good. This may or may not be true depending on the effectiveness each stage of the test. Defective ICs that have passed one stage of the test are not guaranteed to be detected by the next stage. Documentation of faulty ICs passing all stages was presented by Williams and McCluskey [WB81, MB88].

2.2 Definitions

Since some of the terms used in this thesis have been used inconsistently in the literature, they will be defined here.

Defects: anomalies that occur during the fabrication of the IC. These include improperly aligned photolithographics masks, specks of dust on the masks, or uneven ion implant distributions. The defects that we will be concerned with in this thesis are localized defects or *spot defects*. Spot defects are usually caused by specks of dust on the photolithographic masks or contaminants in the processing chemicals.

Circuit Faults: changes in the electrical description of the circuit. These include broken wires, shorted wires, extra transistors, and missing transistors. In this thesis, the

circuit faults that we will be concerned with are caused by spot defects since circuit faults caused by large defects are usually easily detected.

Behavioral Faults: changes to the behavior of the the circuit. This is what traditionally is thought of as a fault. These faults are abstractions of circuit faults. Examples of behavioral faults include changes to the function of the circuit, *logical faults*; faults which cause the circuit to be slower, *delay faults*; and faults that increase the amount of current drawn in normal operation, I_{DD_Q} *faults*.

Figure 2.2 shows the relationship between spot defects, circuit faults, and behavioral faults. Many spot defects may cause the same circuit fault. For example, a long wire can be broken anywhere along its length by many different spot defects, but they will all cause the same circuit fault. Also, many circuit faults can cause the same behavioral fault.

The figure also shows that some circuit faults are not a result of spot defects. These circuit faults can be caused by global defects such as improper metalization or a scratch across the surface of the wafer. Again, we are concerned only with the effects of spot defects in this thesis, since faults caused by global defects are usually easily detected with gross testing.

Finally, we see that some behavioral faults do not seem to be caused by circuit faults. These behavioral faults may not be possible given the layout of the circuit. A behavioral fault may describe a bridge between two nodes that may never occur if the two nodes are on opposite sides of the IC. The bridge may still occur, but everything in between the two nodes must also be bridged together and such a bridge would be easy to detect. Another reason for behavioral faults not caused by circuit faults is that the faulty device may not even exist when the circuit is implemented. An example of this is shown in the 1-bit full adder of Figure 2.3. Here, we have both a logical description of the adder and one possible transistor-level implementation of the adder. A logical fault may involve an exclusive-OR gate or an AND gate, but in the implementation of the adder, it is difficult to determine which transistors are part of the AND gate and which are part of the exclusive-OR gate.

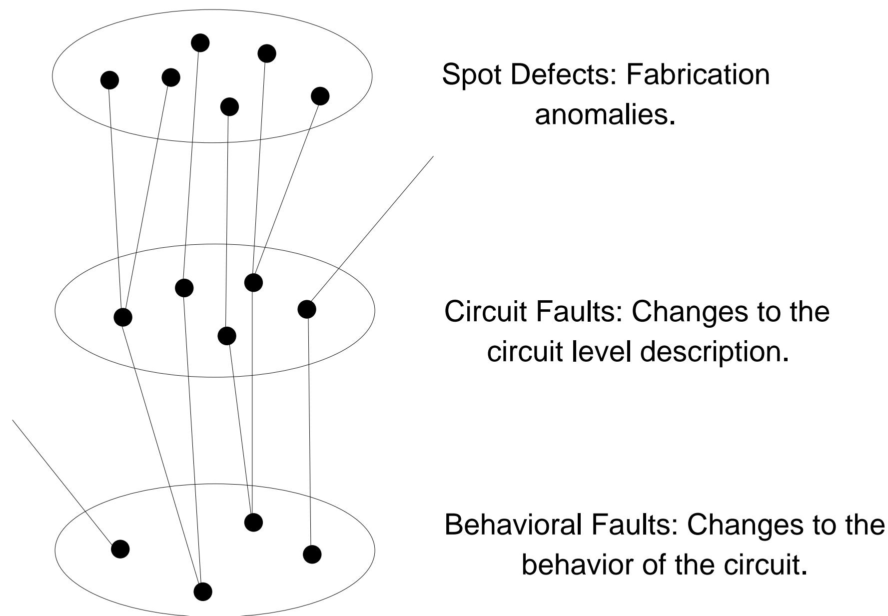

Figure 2.2: Relationship between spot defects, circuit faults, and behavioral faults.

2.3 Fault Models

In order to test for defective ICs, we must predict how a defective IC behaves by developing a *fault model*. Since predicting the behavior of an entire faulty IC is difficult, fault models usually deal with a much smaller unit to test such as a single logic gate.

The job of the fault model is to predict the behavior of defects. The closer the fault model's prediction comes to reality, the more effective the test set is likely to be at identifying defective ICs. Unfortunately, the fault model that describes the behavior of every defective circuit in every detail is usually too complex to be useful for large circuits. This is analogous to circuit simulation. Treating transistors as pure switches requires less simulation time, but may not be accurate in some cases. Using a set of differential equations to model transistors as in SPICE would be very accurate, but would require a great deal of time for even MSI chips.

Some of the fault models currently being used will be discussed in the following sections. To simplify the task of testing ICs, all of the fault models assume that only one fault is present in the circuit at a time. When testing new ICs, it may be that there are more than

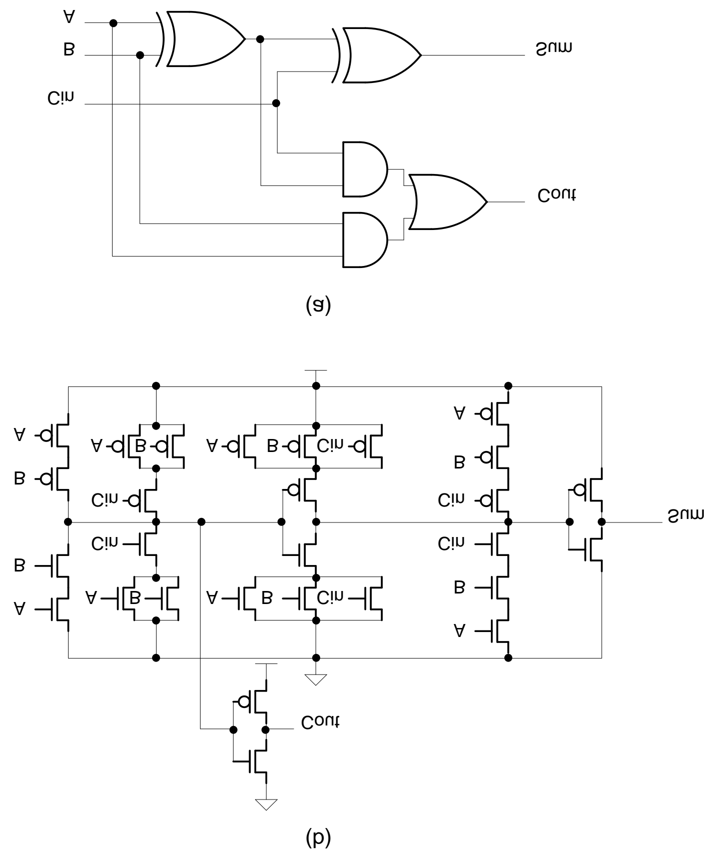

Figure 2.3: 1-bit full adder descriptions (a) Logical (b) Transistor level.

one fault present, but in most cases, multiple faults can be detected by using the tests that were generated to detect single faults [ABF90]. The first fault model we will consider the single-stuck-at fault model.

2.3.1 Stuck-at Fault Model

The most popular logical fault model is the single-stuck-at (SSA) fault model. The SSA fault model deals with a circuit at the logic gate level. A fault in this model is represented as a signal line stuck at a logic value of 0 of 1. Since the faults generate only logical values, as opposed to arbitrary analog voltages, generating tests from the SSA fault model

is computationally relatively simple. When the SSA model was developed, the primary fabrication technology was TTL. In TTL logic, many defects do indeed manifest themselves as a logical stuck fault. Unfortunately, many of the defects that can occur in current CMOS processes are not manifested as logical stuck-at faults [FS88].

2.3.2 Bridge Fault Models

In bipolar logic, break faults tend to cause signal wires to be stuck at a specific logic value. Bridge faults do not have that effect and may not show up as stuck-at faults. The resulting logic value of the wires being bridged together can usually be determined by examining what is driving each wire. The following are some of the bridge fault models that have been developed to test for bridge faults. The first two models, wired-AND and wired-OR, were developed for bipolar logic and have limited application to CMOS logic. The last two bridge fault models, voting model and extra current model, were developed because of the ineffectiveness of the first two models in CMOS applications.

Wired-AND Bridge Model and Wired-OR Bridge Model

Bridge faults usually create a new logical function. In bipolar, the logical functions that represent a bridge fault are logical AND and the logical OR. These functions are used to determine the final logic value of all the signal wires that are bridged together. In the case of an AND bridge fault, for example, the logic value that is propagated to all of the logic gates driven by the bridged signal wires is the logical AND of the logic values of all the bridged wires. These two bridge fault models enable fairly simple test generation, since the faulted values that are propagated through the circuit are valid logic values and the function to calculate the resulting value is simple.

Bridge Fault Voting Model

This bridge fault model deals not with logic gates, but with the transistors that make up the logic gate [Ack88]. This model was created because in CMOS technologies, knowing the

logic values on the output of the bridged gates is sometimes not enough to determine the resulting logic value. This model takes into account the strength of the transistors driving the output to determine the final voltage of the bridge fault and in turn the logic value to propagate.

Since this model computes the logical value of the bridge fault, only valid logic values are propagated through the circuit, thus this model provides a more accurate representation of bridge faults for CMOS circuits with a small additional overhead compared to the previous bridge fault models.

Extra Current Model

All of the other models discussed here detect the presence of a fault by propagating erroneous logic values to the outputs of the IC where they can be checked. The extra current model does not not propagate erroneous logic values to the output, but uses an inherent feature of many CMOS design styles to indicate a defective IC.

Certain CMOS logic design styles use very little power once the logic gates stabilize into a logic state. For this reason, CMOS is heavily used in battery-powered applications such as wrist watches and handheld calculators. It is this feature that allows an alternate method of detecting defective CMOS ICs. Since many of the defects that can occur in CMOS circuits cause the circuit to draw more current than defect-free ICs, monitoring the amount of current the IC draws can be used to identify defective ICs. To detect a bridge fault, the test pattern must set the inputs so that the bridged wires will be driven to different logic values, causing extra current which can be detected at the power supply pins of the IC.

2.4 Inductive Fault Analysis

The goal of testing is to detect defective ICs so that they are not shipped to customers. The success of testing is measured by the fraction of good ICs among all the ICs that were shipped. This measure is known as the quality level (QL). The effectiveness of a given test at detecting defective ICs is given by the fraction of all defects that were not detected by

% Yield	Defects Per Million (DL)			
	50	100	200	300
90	99.953	99.905	99.810	99.715
80	99.978	99.955	99.910	99.866
70	99.986	99.972	99.944	99.916
60	99.990	99.980	99.961	99.941
50	99.993	99.986	99.971	99.957

Table 2.1: Percent of defect coverage required for various combinations of yield and desired defect level.

the test. This metric is called the test transparency (TT). Williams and McCluskey derived the relationship between QL, TT and the overall fabrication process yield (Y) to be

$$QL = Y^{TT}. \tag{2.1}$$

Since TT cannot be measured directly, it is estimated from the fault coverage (C), the percentage of faults detected by the test patterns, by

$$TT = 1 - C. \tag{2.2}$$

QL is computed from the defect level (DL), the number of defective ICs shipped using the following equation:

$$QL = 1 - DL. \tag{2.3}$$

DL is usually measured in defects per million. Table 2.1 shows the defect coverage required for different yields to achieve a given defect level. A commonly accepted defect level is 200 DPM. To achieve this, we can see that greater than 99.9% of the defects must be detected by the tests if the yield is 80% or less.

As we have just seen, detecting a high percentage of the possible defects is very important for low defect levels. Since traditional fault models do not take defects into account at all, it is unlikely that tests generated from those fault models will consistently detect enough defects to ensure adequate quality levels [FS88]. The last two bridge fault models presented

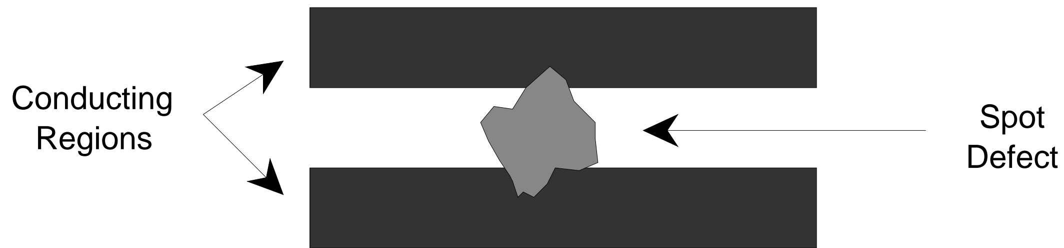

Figure 2.4: Example of a defect causing a bridge fault.

in the previous section dealt with defects directly, but in order for those models to be feasible, the list of possible defects for the IC must be known.

Inductive Fault Analysis (IFA) was developed by Ferguson, Maly and Shen to determine the list of possible defects that can occur in an IC based on its layout and the fabrication process. Clearly, the list of defects is directly dependent on the layout of the IC rather than an abstracted representation of the IC such as a logic gate representation. IFA performs defect simulation on the layout of the IC using predetermined defect sizes to determine the effect of the defect on the circuit. Figure 2.4 shows a defect that causes extra material to deposited that causes a bridge fault to occur. A similar defect, depending on the fabrication technology, could prevent material from being deposited and thus cause a break fault as shown in Figure 2.5.

IFA was implemented in the program FXT, which was used to analyze several circuits from a CMOS standard cell library. The resulting circuit faults caused by defects simulated by FXT are summarized in Table 2.2. Since transistor-stuck-on faults can be treated as bridge faults between the source and drain of the transistor, we can see that >99% of the defects cause either bridge faults or break faults.

Figure 2.5: Example of a defect causing a break fault.

% Defects	Circuit Fault Type
∼50%	Bridge
∼40%	Break
∼10%	Transistor-Stuck-On
<1%	Others

Table 2.2: Percentage of circuit faults caused by spot defects.

3. Carafe

The IFA process was shown feasible by the FXT software; however, FXT was primarily a research tool and was not intended to be used as a general IFA tool. The goal of Carafe is to provide a robust and stable implementation of the IFA process in an easy-to-use package.

3.1 Carafe Defect Model

Carafe models all spot defects as small squares. A square defect model was chosen to make the calculations faster than can be done with circular defects. Since large, global defects, such as mask misalignment and scratches across the surface of the wafer, usually cause drastic changes to the behavior of a circuit. These defects are easily detected by most test sets, so they are not considered by Carafe. Since greater than 99% of all the faults caused by local spot defects are either shorts between two different electrical nodes or an open in a once continuous electrical node, Carafe is geared towards finding the possible bridge and break faults that can occur in a circuit given its layout.

3.2 Locating the Faults

The possible bridging and break faults in a circuit that can be caused by spot defects depend on the feature sizes of the circuit's physical layout. Smaller features are more prone to be affected by defects than large features. Features separated by a great distance are clearly less likely to bridge together than features that are relatively close to one another. Thus, Carafe requires the layout of the circuit to determine the possible bridge and break faults.

3.2.1 Bridge Faults

In Carafe, there are two types of bridging faults. The first type is a bridging fault between two regions on the same layer. The other bridging fault type is between two

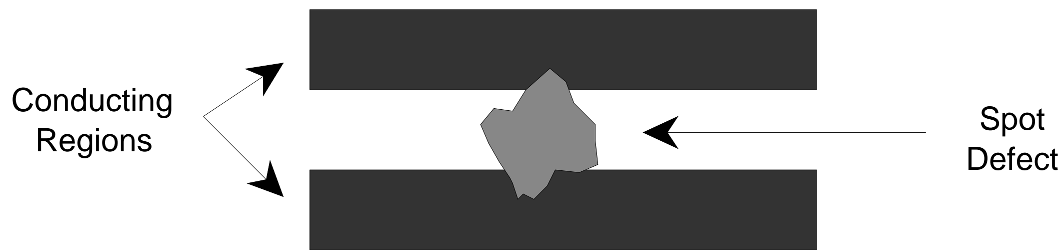

Figure 3.1: Bridge fault type 1.

regions on different layers, such as polysilicon and metal. The first bridging type is caused by extra material being deposited (or not etched away) between two regions of conducting material that are physically close to each other. An example is shown in Figure 3.1. A speck of dust on a photolithographic mask may produce such defects if the size of the dust particle is large enough. The other type of bridging faults occurs when the oxide insulation between two layers of material is not formed correctly and the two layers are allowed to come in contact with each other. Figure 3.2 shows two different types of conducting layers that would not normally be electrically connected together but are connected due to a defect causing the insulating layer to be breached. Note that a bridge that occurs between one part of an electrical node with another part of itself is not a fault as far as Carafe is concerned. Only bridge faults between different electrical nodes are reported by Carafe.

The first type of bridge faults are found by taking each region of material and checking to see if another region of the same material lies within a certain distance away from the original region. The distance used is the largest defect diameter under consideration. Since Carafe models defects as small squares, the diameter is the length of one side of the defect. The process is repeated for every region of every type of material.

The second type of bridging fault can only occur between overlapping conducting regions on different layers. For example, a bridge between a region of polysilicon and a region of metal can only happen if the two regions are overlapping since we are assuming only single spot defects. Finding these bridges entails finding all of the overlaps of one type of material with the other.

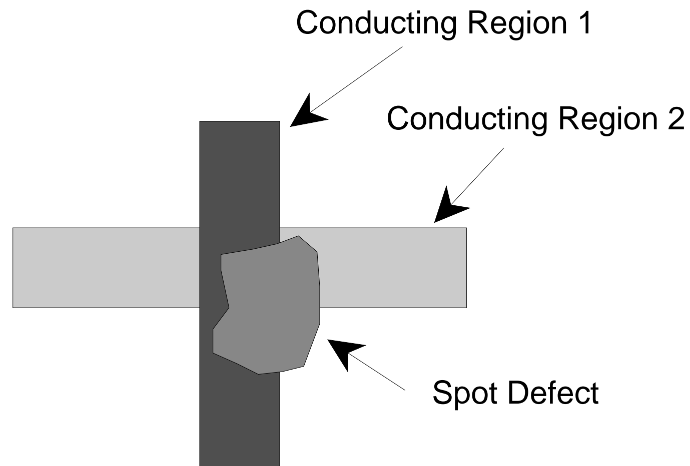

Figure 3.2: Bridge fault type 2.

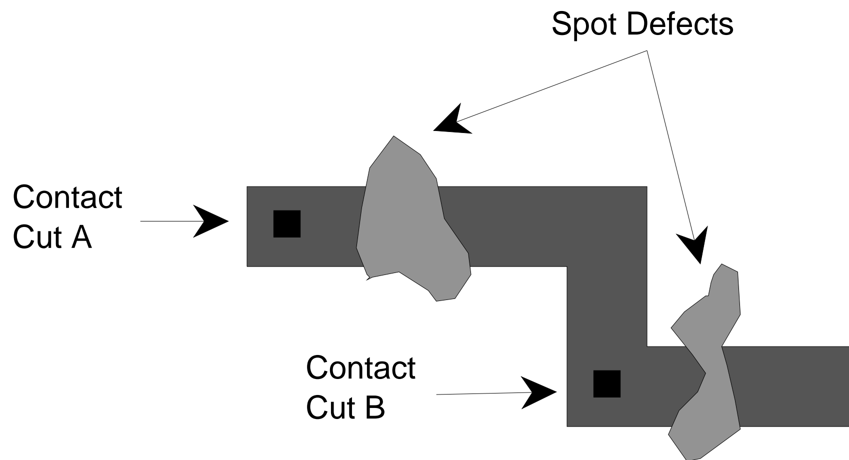

Figure 3.3: Break fault type 1 and an inconsequential break fault.

3.2.2 Break Faults

As with the bridge faults, there are two types of break faults. The first type of break fault causes the separation of an electrical node into two or more nodes on the same layer. The break fault on the left in Figure 3.3 shows a break of this type between contact cut A and contact cut B.

The second break type occurs at the junction of two dissimilar layers of conducting material. This is more commonly called an *open contact*. A single spot defect can cause these break faults by preventing the insulating oxide from being etched away where the contact holes are supposed to be. This results in one part of the electrical node being separated from the other part of the node. There is a chance that a break fault could occur and not partition the netlist of the circuit—perhaps the break fault only isolates a small corner or piece of a larger region of material in a way that does not affect the circuit. The break fault on the right in Figure 3.3 shows an example of a break fault that does not affect the path from contact cut A to contact cut B. Redundancy can also prevent the division of an electrical node as a result of a single break fault by providing multiple conducting paths. Only the break faults that alter the circuit level description are reported by Carafe.

Determining the possible breaks of the second type is relatively simple. The only places where the second type of break fault can occur are at the contacts or vias. A simple search for all of the contacts between the different layers of material yields a list of all possible break faults of the second type.

The first type of break fault is much more difficult to derive from the circuit's layout. The break faults in a given region of conducting material are dependent on the directions of current in the region. If the orientations of the current within a region of conducting material are known, the break faults can be easily found. Unfortunately, the orientations of current are not always readily determined.

3.3 Reducing the Number of Reported Faults

Many defects cause identical faults. These faults are identical in the sense that hey have the same effect on the circuit's electrical behavior. More than one bridge fault bridging the same two nodes together is a simple example. All but one of those bridge faults can be deleted from the fault list.

Identical break faults have one of two forms. In the first form, there is a cycle or more than one path from one point of the node to another. In this case a single break will not

isolate those two points from one another and thus all of the break faults that are found in the cycle can be removed since they will not affect the electrical behavior of the circuit (a single break in this case may affect the parametric behavior of the circuit, but such faults are not considered in Carafe.) In the second form, there is more than one break fault in succession. In this form, each break fault has the exact same effect on the circuit, namely isolating point A from point B. These redundant faults are dealt with in the same way as redundant bridge faults. Inconsequential breaks that do not change the netlist of the circuit can also be deleted from the fault list.

By removing inconsequential faults and all but one fault from each class of identical faults, Carafe reduces the time for fault simulation and test pattern generation. Since fault simulation with the COSMOS fault simulator is orders of magnitude more expensive than fault extraction with Carafe, any reduction in the number of faults that need to be simulated will greatly improve the overall computation time.

3.4 Modeling the Faults for Simulation

Determining the possible faults is only a small part of the testing picture. Once the realistic faults are identified, fault simulation can be used to determine the effect of each fault on the circuit. Fault simulation may show whether the fault actually affects the behavior of the circuit, and whether the fault can be detected by any specific test set.

The first step in finding all of the possible faults in the circuit is the extraction of the transistor-level representation of the circuit. This is done using the standard region merging algorithms. Next, for each fault that Carafe finds, a transistor is inserted into the original netlist to model the effect of that fault. For bridge faults, a transistor is inserted across the two nodes of the bridge fault. Turning the transistor on causes a short to appear across the two nodes. Break faults are modeled with the insertion of transistors between the two points to be broken. Turning the transistor off isolates the two points of the break fault from each other. Examples of these transistors are shown in Figure 3.4.

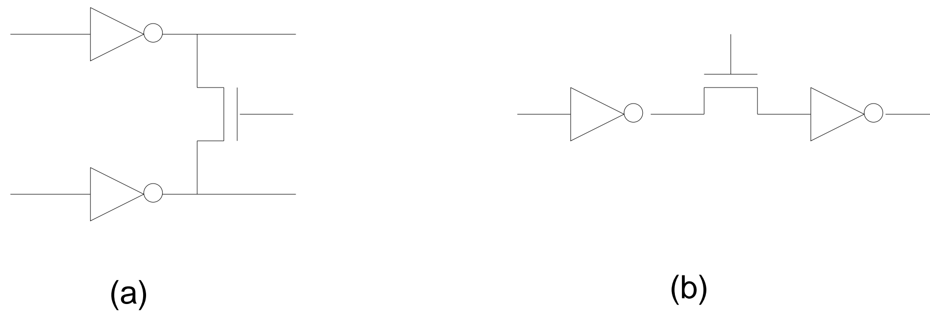

Figure 3.4: Modeling bridge and break faults as extra transistors.

If all of the bridge transistors are turned off and all of the break transistors are turned on, the circuit will behave as if there were no faults in it at all.

Once the fault transistors are in place in the netlist, the bridge and break faults can then be simulated as stuck-at faults at the gate nodes of the fault transistors. In doing so, we can now simulate bridge and break faults using a switch level, stuck-at fault simulator to determine the effect of these faults and determine the bridge and break fault coverage of any given test set. Carafe generates a simulation command file for use with the switch-level fault simulator COSMOS.

3.5 Ranking the Faults

Carafe uses the concept of sensitive area to determine the likelihood of each fault. The *sensitive area* is defined as the area in which the center of a defect of a given diameter must fall in order to cause the fault. An example of the sensitive area for a break fault in a rectangular region of conducting material is shown in Figure 3.5. The sensitive area estimates the likelihood of the given fault occurring. Larger sensitive areas imply more likely to occur faults.

The sensitive areas calculated for the faults are dependent on the layer in which the regions of material lie. For most fabrication processes, defects due to spot defects are more likely to occur during the fabrication of the upper layers, such as the metal layers, than in the lower layers, such as the polysilicon layers. For this reason, Carafe has provisions

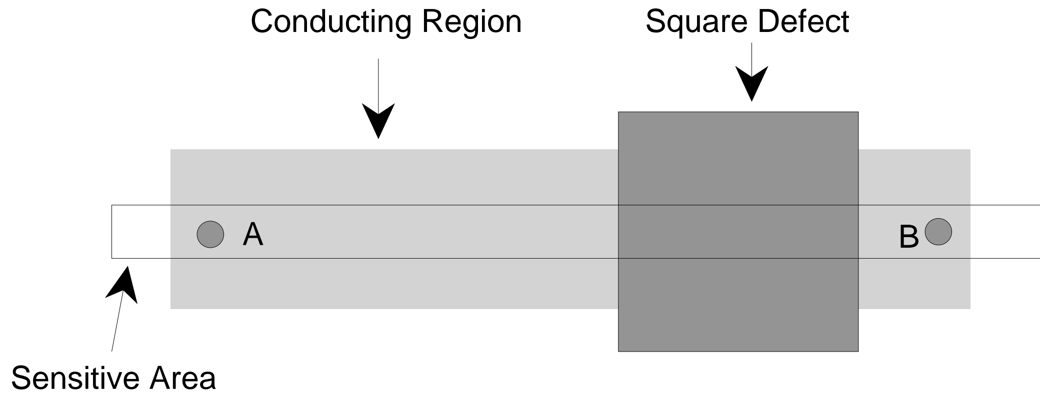

Figure 3.5: Sensitive area for a break fault between point A and B.

for including any defect distributions that occur in a given fabrication line organized by layer and by defect radii. This allows maximum flexibility in using Carafe with real defect distributions to achieve more realistic fault results and likelihoods of occurrence.

3.6 Implementation Details

This section provides some of the implementation details of the Carafe program. This section is by no means a complete and thorough account of the complete program, but it describes the major data structures and algorithms used.

Carafe is written in the "C" programming language under the UNIX operating system. The code was written with portability in mind to increase the usefulness and availability of the program. The original code was written and debugged on an IBM PC and ported to Sun workstations with no recoding.

3.6.1 Circuit Representation

Carafe uses the layout of circuits to determine the realistically possible bridge and break faults. Carafe can read electronic representations of circuit layout in either the Calma GDS format or the Berkeley Magic format.

The internal representation of a layout is kept in a data structure similar to the data structure of the Berkeley Magic CAD tool [OHM+84]. The layouts are represented in a series of planes which contain corner-stitched tiles representing the regions of material used in the fabrication process. The structure of the tiles allows efficient area searches and enumerations, which are used frequently in the circuit and fault extraction algorithms.

Because the tiles are rectangular, only Manhattan geometries can be represented in the tile planes. Layouts containing non-Manhattan geometries are represented by a series of small rectangles to approximate the original polygon.

Each tile plane may represent more than one fabrication layer. The most common example of this is the plane that contains the transistors. In this case, both forms of diffusion (n-diffusion and p-diffusion) reside on the same tile plane as polysilicon, even though diffusion and polysilicon are entirely different fabrication layers. This was done so that the extraction of the circuit's transistors would not involve computationally expensive geometric operations. A quick check of the tile's contents can reveal the locations of the transistors. For more information refer to the Carafe User's Manual [Jee90].

3.6.2 Circuit Extraction

The circuit extraction done by Carafe is similar to the circuit extraction done by Magic [SO86]. The extractor groups the tiles into electrical nodes by a recursive algorithm. Two touching tiles are considered to be in the same electrical node if they are the same type or they are specified to be electrically connected in the technology file. Carafe's circuit extractor is much simpler than the magic circuit extractor, since Carafe does not extract node resistance or parasitic capacitances. Carafe only extracts the transistors of a circuit and their connectivity.

Before the nodes are extracted, the transistor regions are determined using the same process described above except that only the tiles representing transistors, as specified in the technology file, are considered. The length and width of each transistor is computed for use in simulation. The length and width of rectangular transistors are simple to compute,

but non-rectangular transistor channels are more difficult to calculate. The length of the transistor is computed by averaging the lengths of the perimeter of the transistor that are not adjacent to diffusion. The width of the channel is determined by averaging the lengths of the perimeter that are adjacent to diffusion.

3.6.3 Bridge Fault Extraction

To determine the possible bridge faults we take every tile that belongs to an electrical node and enumerate all of the tiles that are within a specified distance from the original tile. For each tile found in the search, if the tile belongs to a different node than the original, a potential bridge has been found. A list of all bridge faults found is created, and checked each time a new bridge fault is found to prevent duplicate bridge faults from appearing.

The distance that is searched is the largest defect size that the user is interested in. This distance is used only to determine the possible bridge faults and does not affect the calculation of sensitive areas for the faults.

3.6.4 Break Fault Extraction

The break fault extractor is currently not implemented, but is being developed and should be ready in the near future.

3.6.5 Sensitive Area Calculation

There are several different types of sensitive areas that are found by Carafe. This section describes each type and how they are calculated.

Break-Fault Sensitive Area : the following is a description of the different types of possible break-fault sensitive areas and the method used to calculate them.

- **Intraplanar** : these break faults occur on a single layer of conducting material. Figure 3.6 shows an example of this type of break fault. The equation for this type of sensitive area is

$$SA = L(2R - W) \tag{3.1}$$

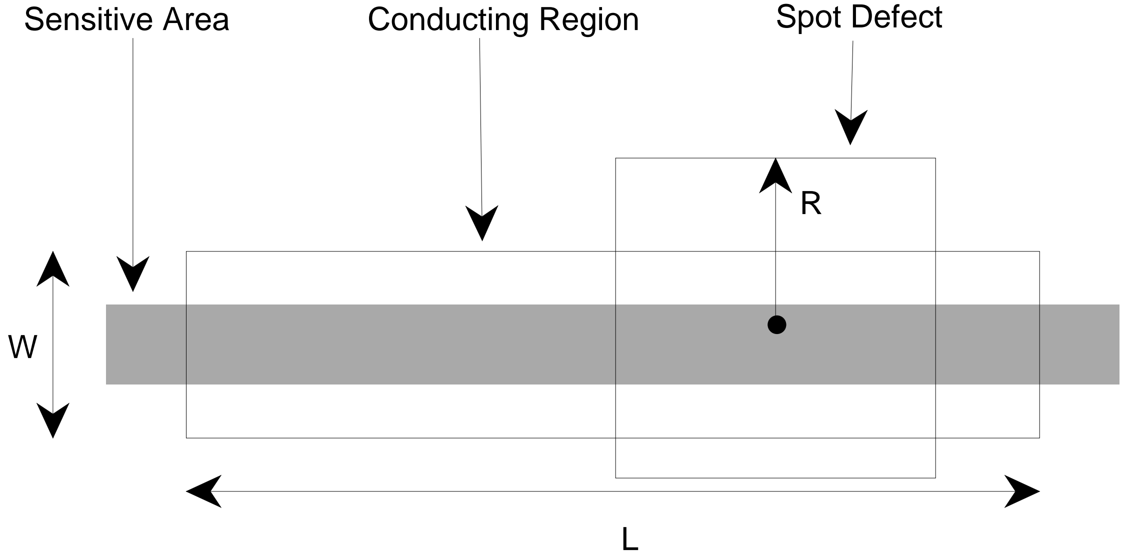

Figure 3.6: Sensitive area for a break fault with a horizontal length.

where L is the direction of the current, W is the width of the conducting material and R is the radius of the defect currently being used.

Interplanar : these break faults occur in the connections between two different layers of conducting material. These connections, known as contact cut or vias, can fail, separating the node into two pieces. The sensitive area for this type of break fault is given by

$$SA = (2R - W)(2R - L) \tag{3.2}$$

where W and L are the length and width of the contact cut and R is the current defect radius.

Bridge-Fault Sensitive Area : the following is a description of the different types of possible bridge faults and the equations used to find their sensitive areas.

Intraplanar : for bridge faults that occur between two regions of the same type of conducting material, there are three possible orientations for length of the sensitive area: horizontal, where the regions are above and below each another;

vertical, the regions are to the left and right of each other; or diagonal, the regions are not directly above, below, to the left, or to the right of each other. The horizontal and vertical sensitive areas are computed similar to the intraplanar break faults using the equation

$$SA = L(2R - W) \tag{3.3}$$

where L is the length of the area between the regions, W is the distance between the two regions, and R is the current defect radius. For diagonal bridge faults, the sensitive area is given by

$$SA = 2R\sqrt{W^2 + L^2} \tag{3.4}$$

where W and L are the Manhattan distances between the two nearest corners of the two regions.

Interplanar : these bridge faults occur when the insulating material between two layers becomes defective allowing the two layers to become connected. Carafe assumes that these faults can only occur where two regions of different layers directly overlap each other. The equation used to compute this type of sensitive area is

$$SA = (2R + L)(2R + W) \tag{3.5}$$

where L and W are the dimensions of the area of overlap and R is the current defect radius.

Each sensitive area is multiplied by the defect density found in the fabrication statistics file for the layers of the sensitive area for the given defect radius. The fabrication statistics file allows any defect radius distribution to be incorporated into the sensitive areas calculated by Carafe. The final sensitive area for each fault is the sum of all the sensitive areas found for every defect radius, multiplied by the defect density (DD) added together. The equation of the final sensitive area for a specific circuit fault is given in Equation 3.6.

$$FinalSA = \sum_{r \in \text{Defect radii}} \sum_{i \in \text{\# of SA}} SA_i(radius[r])DD(layers_i, r) \tag{3.6}$$

4. Results

Carafe was used to analyze the set of standard ISCAS'85 benchmark circuits [BF85] implemented using a CMOS standard cells by the Mircoelectronics Center of North Carolina. Since the MCNC implementations of the benchmark circuits were in a hierarchical, standard cell form, they were flattened into one level of hierarchy before extracting the faults.

The defect radius used in these experiments was 4 microns and the defects were assumed to equally likely on all layers of material. The experiments were all run on Sun SPARCstation 1+ workstations.

Table 4.1 summarizes the performance of Carafe and the number of faults that were found. The first column shows the number of nodes that the circuit contained. Since Carafe works at the transistor level, not the gate level, all nodes, including nodes internal to logic gates, are reported. The second column shows the area of the circuit in square microns. The next column shows the number of bridge faults that were found. The last column in the table shows the time in CPU seconds that Carafe needed to perform the circuit extraction, fault extraction, fault likelihood computations, and the generation of the fault simulation command files.

Bridge faults to power and ground will probably behave as stuck faults. Nodes that are internal to gates bridged to power or ground may not act as a stuck-at fault. The

Circuit	# Nodes	Area (μ^2)	Bridge Faults	Time (s)
C17	23	217.8x10^2	77	0.98
C432	429	6.419x10^5	2941	56.88
C499	832	10.88x10^5	5778	153.89
C880	667	11.66x10^5	5624	145.80
C1355	963	19.32x10^5	7396	243.53
C1908	1146	17.74x10^5	8819	335.19
C2670	1697	40.17x10^5	19891	1196.12
C3540	2177	54.35x10^5	24628	1944.44
C5315	3623	111.11x10^5	55290	9194.24
C6288	4352	88.86x10^5	35631	4704.37
C7552	4771	143.42x10^5	72999	15310.80

Table 4.1: Bridge faults for ISCAS'85 benchmark circuits.

ISCAS Circuit	Bridge Faults	Bridges to Power	Bridges to Ground
C17	77	25	25
C432	2941	501	473
C499	5778	904	912
C880	5624	815	835
C1355	7396	1127	1127
C1908	8815	1294	1362
C2670	19891	2220	2402
C3540	24628	2989	2944
C5315	55290	5489	5881
C6288	35631	5476	5412
C7552	72999	6868	6980

Table 4.2: Bridge faults to power and to ground

next section will address this issue. Table 4.2 shows the number of bridge faults that were bridged to power or to ground.

4.1 MCNC Standard Cells

Carafe was used to analyze some of the standard cell gates from MCNC. Each bridge fault found in the standard cells were simulated with all possible input vectors using SPICE with the bridge fault modeled as a 0.1Ω resistor. The inputs to the cells were applied through single-drive inverters from the same standard cell library. Applying the inputs in this manner provides a realistic instance of the faulty cell, if the bridge fault involved any of the input nodes. A more detailed analysis of the cell library would use other cells to drive the inputs also, since the drive strength of the other cells could affect the circuit's behavior when the bridge fault involves the faulty cell's inputs. The logic gates that were analyzed were 2-input versions of the AND, NAND, OR, NOR, and XOR logic gates. The maximum defect size was set to 4 microns and the defects were assumed to be equally likely on all layers of material. The results of this experiment are shown in Table 4.3. The first column indicates the gate that was analyzed with the number of bridge faults found for each gate in the second column.

Many of the cells have feed-through lines in the metal-2 layer that allow signals to

Gate	Bridges	Feed-thru	Stuck-at	No Effect	Other
AND	19	26%	53%	5%	16%
INV	5	0	80%	20%	0
NAND	11	0	73%	9%	18%
NOR	11	0	73%	0	27%
OR	19	26%	42%	11%	21%
XOR	31	32%	29%	13%	26%

Table 4.3: Bridge faults for logic gates broken down by effect.

extend from one wiring channel to an adjacent channel separated by a row of cells. The third column in the table shows the percentage of the faults that bridged a node of the gate to one of these feed-through lines. In this experiment, bridge faults to these feed-throughs have no effect on the function of the gate since the feed-throughs were not connected to any other gates. In a real circuit, these feed-throughs could be driven by other gates. In this case, the logic function of both the gate driving the feed-through and the faulty gate would be affected.

The fourth column of the table shows the percentage of the bridge faults that behaved as stuck-at faults. The SPICE simulations for these faults produced outputs that were the same as the outputs of the gate in the presence of a stuck-at fault. These bridge faults would be detected by any test set with complete stuck-at coverage.

The "No Effect" column shows the percentage of the bridge faults that had no effect on the logical function of the gate. The bridge faults in this category usually created output voltages other than GND or Vdd, but were within 1.5 volts of GND or Vdd and thus were treated as valid logic values. These faults may be detected as an increased propagation delay or as increased quiescent power supply current. The last column of the table shows the percentage of the bridge faults that either changed the function of the gate or produced invalid logic values at the output of the gate, but are not represented in the other columns.

The data in Table 4.3 is of value for automatic test pattern generation. Assuming complete stuck-at coverage, the number of faults that need to be considered for detecting bridge faults within the cell is at most the sum of the faults that fall into the "Feed-Thru" and "Other" categories.

Gate	Bridges	Feed-thru	Stuck-at	No Effect	Other
AND	100%	17%	52%	12%	19%
INV	100%	0%	78%	22%	0%
NAND	100%	0%	62%	22%	16%
NOR	100%	0%	60%	0%	40%
OR	100%	21%	43%	7%	29%
XOR	100%	18%	33%	12%	37%

Table 4.4: Percentage of defects causing bridge faults broken down by effect.

The data in Table 4.4 is more useful for evaluating the defect coverage levels of test sets. It shows the percentage of spot defects that caused bridge faults arranged by the effect on the gate. The first column shows the name of the gate analyzed. The fourth column in the table shows the percentage of spot defects that caused a stuck-at bridge fault. Since a complete stuck-at test set is certain to detect all bridge faults in this category, it will have at least this percentage of bridge-fault-inducing spot defects. Those in the sixth column may cause a logical fault that is not detected by a complete stuck at test set.

4.2 Defect Coverages

The sensitive areas for all the bridge faults for the MCNC ISCAS'85 circuits were sorted and plotted against the number of faults. An example plot for the C7552 circuit is shown in Figure 4.1. The C7552 contains 1474 logic gates and 4741 nodes (including nodes internal to logic gates). The maximum defect size chosen for this experiment was 4 microns and the defects were assumed to be equally likely on all layers of material.

Figure 4.1 clearly shows that some of the bridge faults are much more likely to occur than others. A cumulative density function plot of the same data is shown in Figure 4.2. From the figure, we can see that the most likely 30% of the bridge faults account for over 57% of the spot defects that cause bridge faults. If realistic defect distribution densities were used, the percentage of defects represented by the most likely 30% of the faults would probably be greater.

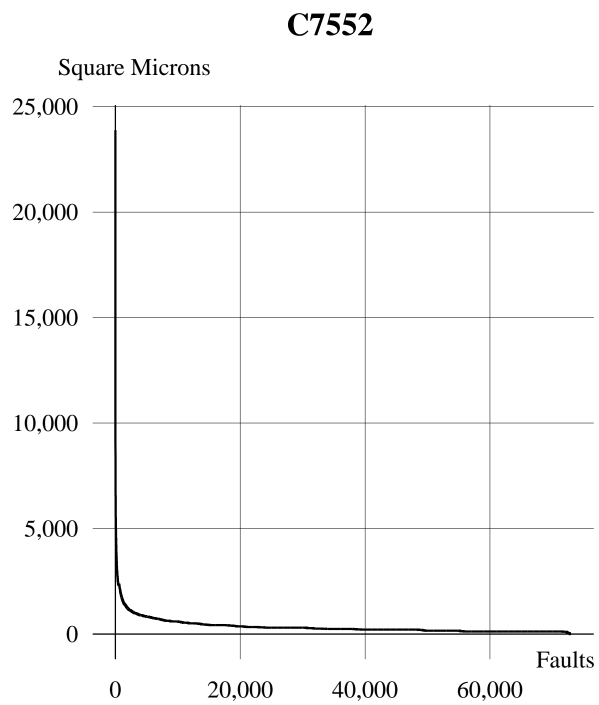

Figure 4.1: Plot of the bridge fault sensitive areas for the C7552 ISCAS'85 circuit sorted from highest to lowest.

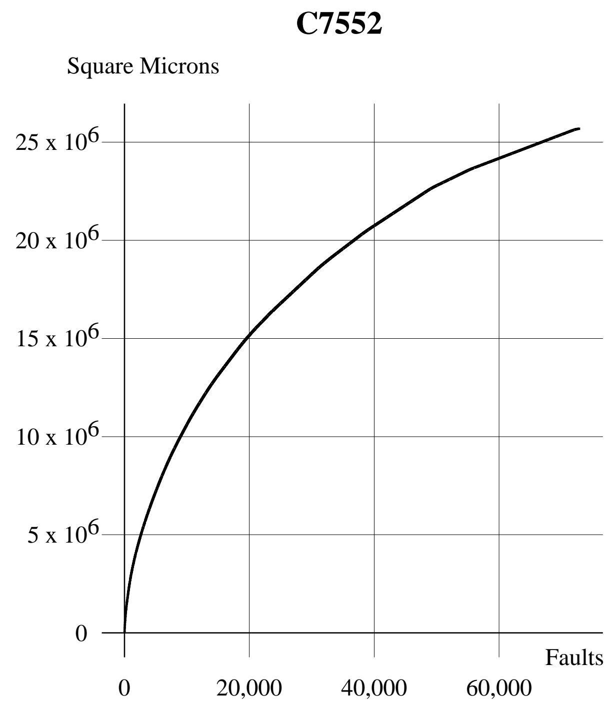

Figure 4.2: Plot of the cumulative sorted bridge fault sensitive areas for the C7552 ISCAS'85 circuit.

5. Current Limitations of Carafe

In this chapter, we will discuss a few aspects of extracting break faults including a plan to implement them and a discussion of some some potential pitfalls. Then, we will show how to use Carafe to analyze large circuits.

5.1 Break Faults

Break faults have always been more difficult to extract from the layout of circuits than bridge faults. The difficulty lies in determining the exact path of current between transistors. Once the path is found, the break faults can be easily found.

To find the direction of current, we find the topology of each node to determine the relationship of each transistor to the other transistors in the node. Once this is determined, the direction of current and thus the break faults can be found. We are currently using arbitrary polygons to represent the nodes and are investigating various algorithms that produce a skeleton of the polygon, which should reveal the topology of the node.

Multiple contact cuts connecting the same regions together make it more difficult to find the break faults since all of the contact cuts must be broken in order to create a break fault. Odd-shaped transistors, such as "U" shaped, present another difficulty in determining the exact topology of a node since these transistors may not be represented as a single point of contact in the node skeleton.

5.2 Analyzing Large Circuits

From the data in Table 4.1, it would seem that using Carafe to analyze large circuits is impractical because of the CPU time required. We have found that the fault simulation of these circuits with switch-level simulators is much more time consuming than extracting the faults. This problem may be solved by using a logic-level fault simulation and partitioning the circuit in a hierarchical fashion.

In a hierarchical design, the faults for the logic gates need only be extracted once since the gate does not change. All that is necessary after finding the faults for each gate is finding the faults that occur in the interconnect wiring. Carafe has already been modified to find the bridge faults in the interconnect wires only. Finding the faults for a circuit this way would decrease the time needed to find all of the faults in the circuit. Finding the faults in this manner, however, will miss the faults that occur in the interaction of two adjacent logic gates in the circuit. These faults will need to be studied in more detail to determine their importance.

6. Applications of Carafe

The sensitive areas can be used for a variety of applications. A few of them are discussed in this chapter.

6.1 Grading Test Sets for Defect Coverage

One important aspect of the sensitive area for a fault is that it provides a way to convert fault coverage to defect coverage. It has been shown that quality level is related to defect coverage of the test set used. Since sensitive area allows us to relate fault coverage to defect coverage, we can now relate fault coverage to quality level.

The ratio of a fault's sensitive area to the circuit's total sensitive area provides an estimate on the percentage of defects that would be detected given that the fault is detected. By summing all the sensitive areas of the faults detected by a given test set, we can determine the defect coverage of the test set.

6.2 Physical Design for Testability

The testability of an IC is usually not considered during the IC design process. IC designers are reluctant to modify an IC design to accommodate the testing process once the design is finished. Carafe can be used to analyze the design in the early stages to flag potential testing problems.

The list of faults generated by Carafe can help guide the layout of circuits. Since it is very difficult to design and fabricate a circuit with no defects at all, it would be beneficial to design the circuit so that any defects that do occur manifest themselves as easy-to-detect faults. Some difficult-to-detect faults are those that introduce feedback into the circuit and those that create invalid logic values. We would want the likelihood of these kinds of faults to be much lower than the likelihood of more easily detected faults such as stuck-at faults. The circuit designer can take the list of likelihoods of the faults from Carafe as a guide to modify the circuit to be easier to test.

6.3 Design for Manufacturability

Carafe can be used to estimate the relative manufacturability or yield of a given circuit layout. The sum of the sensitive areas for all of the faults for a given circuit can be compared to the sum of sensitive areas for the same circuit with a different layout. Since the sensitive areas are related to the number of spot defects that cause faults, the design with a lower sensitive area sum will be more resistant to spot defects and thus have a higher yield. By the same token, the sensitive areas may be used to refine spacing rules for a given fabrication process to improve yields.

7. Conclusions and Further Research

Carafe is a program for analyzing CMOS circuits for realistic bridge and break faults caused by spot defects. Using the possible fabrication defects to determine which faults can occur provides a much more realistic picture of how a circuit can fail. Fault simulating the realistic faults provides a measure of test quality that is superior to stuck-at fault coverage. Test sets generated for realistic faults should be more effective and provide a higher quality level of ICs shipped.

Carafe improves on previous implementations of IFA in several ways. More efficient algorithms are used in Carafe so that larger circuits may be analyzed. Carafe creates command scripts for use with the COSMOS fault simulator to aid in fault simulation and test set grading. Carafe was written so that it will work with almost any CMOS fabrication process available. The reduction of the number of faults improves fault simulation time tremendously and fault simulation is currently the bottleneck.

Since using Carafe to analyze an entire chip can generate an unwieldy amount of data, it is possible to use Carafe to characterize small sections of a circuit individually and then combine the together to characterize the realistic faults for a much larger circuit. This works well with standard cell and gate array implementations of circuits. Carafe can be used to analyze each logic gate or primitive individually and to analyze the interconnecting wires for faults separately. Partitioning the work in this manner results in a more palatable method of characterizing large circuits. Work is currently underway to generate bridging fault test patterns for circuits where the bridging faults are assumed to occur only in the interconnecting wires of standard cell implementations of circuits.

Many interesting experiments have yet to be done with Carafe, including:

- Determine the actual effectiveness of various stuck-at test sets in detecting the realistic faults.
- Determine the sensitivity of defect distributions on the sensitive areas. Since defect distributions are generally difficult to obtain, this would be a worthwhile experiment.

- Develop a test pattern generation tool that tests for bridge and break faults directly rather than the usual stuck-at fault.
- An entire standard cell library could be analyzed as was done in Section 4.1.

References

[ABF90] Miron Abramovici, Melvin A. Breuer, and Arthur D. Friedman. *Digital Systems Testing and Testable Design*. Computer Science Press, 1990.

[Ack88] John M. Acken. *Deriving Accurate Fault Models*. PhD thesis, Stanford University, 1988.

[BBB+87] R.E. Bryant, D. Beatty, K. Brace, K. Cho, and T. Sheffler. COSMOS: A compiled simulator for MOS circuits. In *Proceedings of Design Automation Conference*, pages 9–16. ACM and IEEE, 1987.

[BF85] F. Brglez and H. Fujiwara. A neutral netlist of 10 combinational benchmark circuits and a target translator in FORTRAN. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1985.

[Fer87] F. Joel Ferguson. *Inductive Fault Analysis of VLSI Circuits*. PhD thesis, Carnegie Mellon University, Department of Electrical and Computer Engineering, October 1987.

[FS88] F. Joel Ferguson and John P. Shen. A CMOS fault extractor for inductive fault analysis. *IEEE Transactions on Computer-Aided Design*, 7(11):1181–1194, November 1988.

[GCV80] J. Galiay, Y. Crouzet, and M. Vergniault. Physical versus logical fault models in mos lsi circuits: Impact on their testability. *IEEE Transactions on Computers*, C-29(6):283–287, June 1980.

[Jee90] Alvin Jee. Carafe user's manual. Technical Report UCSC-CRL-90-61, University of California at Santa Cruz, 1990.

[MB88] E.J. McCluskey and F. Buelow. IC quality and test transparency. In *Proceedings of International Test Conference*, pages 295–301. IEEE, 1988.

[OHM+84] John K. Ousterhout, Gordon T. Hamachi, Robert N. Mayo, Walter S. Scott, and George S. Taylor. Magic: A VLSI layout system. In *Proceedings of the Design Automation Conference*, pages 152–159, 1984.

[SMF85] J.P. Shen, W. Maly, and F.J. Ferguson. Inductive fault analysis of MOS integrated circuits. *IEEE Design and Test of Computers*, 2(6):13–26, December 1985.

[SO86] Walter S. Scott and John K. Ousterhout. Magic's circuit extractor. *IEEE Design and Test*, pages 24–34, February 1986.

[WB81] Thomas W. Williams and N. C. Brown. Defect level as a function of fault coverage. *IEEE Transactions on Computers*, C-30(12):987–988, December 1981.