BRIDGING FAULT ANALYSIS IN DIGITAL CIRCUITS

by

René Manuel Haas

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREES OF
BACHELOR OF SCIENCE
and
MASTER OF SCIENCE
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 1974

Signature of Author ..................................................
Department of Electrical Engineering, January 30, 1974

Certified by .......................................................
Thesis Supervisor

Accepted by ..........................................................
Chairman, Departmental Committee on Graduate Students

Archives

[Stamp]

[Stamp]
BRIDGING FAULT ANALYSIS IN DIGITAL CIRCUITS

by

René Manuel Haas

Submitted to the Department of Electrical Engineering on January 30, 1974 in partial fulfillment of the require-
ments for the Degrees of Bachelor of Science and Master of Science.

ABSTRACT

This thesis is concerned with the detection and diagnosis of bridging (short-circuit) faults on newly manu-
factured digital circuit board assemblies. It is analytically determined that certain bridging faults are always detected by tests designed to detect stuck-at-0, stuck-at-1 failures. In addition, some expected bounds for detection of bridging faults by stuck tests are given.

Since the analytical results apply only to combinational circuits, the detection and diagnosis of bridging faults is investigated experimentally on large sequential circuits with the aid of a fault simulator. The electrical behavior of the shorted transistor-transistor logic elements is studied and a logical model is presented for this behavior.

The bridging fault model and considerations of the modular structure of digital circuits are used to design and implement the Bridging and Stuck-at Digital Fault Simulator. This simulator is used to quantitatively deter-
mine the detection of bridging faults formed by shorts between the adjacent pins of integrated circuit packages. Experiments comparing the detection of bridging and stuck-at faults for 22 digital circuits and test sequences indicate that the test sequences designed to detect stuck-at type failures are also usually effective in detecting most bridging failures. Finally, the stuck-at model was experi-
mentally evaluated for its effectiveness in locating bridging faults. It is found that the stuck-at fault model is inadequate for location of bridging faults, and that the bridging fault model must be used to obtain accurate diagnostic data for such faults.

THESIS SUPERVISOR: Samuel J. Mason
TITLE: Professor of Electrical Engineering

-2-
ACKNOWLEDGMENT

I would like to express my gratitude to the many people who made this thesis possible. In particular, thanks are due to my thesis advisor, Professor Samuel Mason, who encouraged me in this work.

I especially thank Dr. Lutz Henckels for his generous assistance and guidance in preparing the thesis.

I am also grateful to my friends, David Schneider, Ken Brown and Mark Swanson for their help in collecting the data and reading the manuscript; and to Norma Robinson who typed the final copy.

Finally, I would like to thank Sharon for her infinite love and patience in the last four weeks.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter 1</th>
<th>Introduction</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Statement of the Problem</td>
<td>9</td>
</tr>
<tr>
<td>1.2</td>
<td>Definitions and Background</td>
<td>12</td>
</tr>
<tr>
<td>1.3</td>
<td>Outline of this Thesis</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2</th>
<th>The Stuck-at-0, Stuck-at-1 Model</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Description of the Model</td>
<td>24</td>
</tr>
<tr>
<td>2.2</td>
<td>Use of the Stuck-at Model for</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Circuit Testing</td>
<td>28</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 3</th>
<th>The Bridging Fault Model</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Bridging Mechanisms on Electrical Circuits</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>Electrical Characteristics of Shorted Transistor-Transistor Logic Devices</td>
<td>36</td>
</tr>
<tr>
<td>3.4</td>
<td>Logic Representation of the Bridging Fault Model</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>Analytical Results Relating the Bridging and Stuck-at Fault Models</th>
<th>54</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>54</td>
</tr>
<tr>
<td>4.2</td>
<td>Detection of Bridging Faults in Combinational Logic Networks</td>
<td>55</td>
</tr>
<tr>
<td>4.3</td>
<td>Discussion</td>
<td>69</td>
</tr>
<tr>
<td>4.4</td>
<td>Some Expected Bounds for Bridging Fault Detection</td>
<td>70</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>The Bridging and Stuck-at Fault</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Simulation Requirements</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>Fault Simulator Design</td>
<td></td>
</tr>
<tr>
<td>5.3.1</td>
<td>Table Driven Simulation</td>
<td></td>
</tr>
<tr>
<td>5.3.2</td>
<td>Three State Simulation</td>
<td></td>
</tr>
<tr>
<td>5.3.3</td>
<td>Parallel Simulation</td>
<td></td>
</tr>
<tr>
<td>5.3.4</td>
<td>The Delay Model</td>
<td></td>
</tr>
<tr>
<td>5.3.5</td>
<td>Selective Trace Algorithm</td>
<td></td>
</tr>
<tr>
<td>5.3.6</td>
<td>Functional Level Simulation</td>
<td></td>
</tr>
<tr>
<td>5.3.7</td>
<td>Fault Simulation</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>Tester Interface to the Simulator and Date Collection</td>
<td></td>
</tr>
<tr>
<td>5.4.1</td>
<td>Tester Interface</td>
<td></td>
</tr>
<tr>
<td>5.4.2</td>
<td>Data Collection</td>
<td></td>
</tr>
</tbody>
</table>

Chapter 6 | Experimental Results and Conclusions | 100 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Detection of Bridging Faults</td>
<td>100</td>
</tr>
<tr>
<td>6.2</td>
<td>Verification of the Bridging Fault Model Implementation</td>
<td>105</td>
</tr>
<tr>
<td>6.3</td>
<td>Diagnosis of Bridging Faults</td>
<td>110</td>
</tr>
</tbody>
</table>
# List of Tables and Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A Gate</td>
<td>14</td>
</tr>
<tr>
<td>1.2</td>
<td>A Logic Network</td>
<td>17</td>
</tr>
<tr>
<td>2.1</td>
<td>A TTL NAND Gate</td>
<td>27</td>
</tr>
<tr>
<td>2.2</td>
<td>Module with internal fanout</td>
<td>29</td>
</tr>
<tr>
<td>2.3</td>
<td>Module with no internal fanout</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>Typical TTL circuit</td>
<td>37</td>
</tr>
<tr>
<td>3.2</td>
<td>Two outputs shorted together</td>
<td>39</td>
</tr>
<tr>
<td>3.3</td>
<td>Static Model of two shorted TTL outputs</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>Test circuit for two shorted outputs</td>
<td>45</td>
</tr>
<tr>
<td>3.5</td>
<td>Short from an input to the output of NAND/NOR GATE</td>
<td>47</td>
</tr>
<tr>
<td>3.6</td>
<td>Test circuit for shorts between the output and an input of a gate</td>
<td>47</td>
</tr>
<tr>
<td>3.7</td>
<td>Oscilloscope waveforms for shorted outputs</td>
<td>48</td>
</tr>
<tr>
<td>3.8</td>
<td>Function of AND Bridge</td>
<td>51</td>
</tr>
<tr>
<td>3.9</td>
<td>Function of NAND switch</td>
<td>51</td>
</tr>
<tr>
<td>3.10</td>
<td>Function of NOR switch</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Reconvergent Fanout</td>
<td>58</td>
</tr>
<tr>
<td>4.2</td>
<td>Undetectable IBF</td>
<td>58</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.3</td>
<td>Sensitizing the Wired Function</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>NAND Switch</td>
<td>65</td>
</tr>
<tr>
<td>4.5</td>
<td>An IFBF</td>
<td>67</td>
</tr>
<tr>
<td>4.6</td>
<td>Two NAND Network</td>
<td>71</td>
</tr>
<tr>
<td>4.7</td>
<td>Two gates shorted together</td>
<td>74</td>
</tr>
<tr>
<td>4.8</td>
<td>AND Bridge/NAND Switch Transformation</td>
<td>78</td>
</tr>
<tr>
<td>4.9</td>
<td>NOR Switch Transformation</td>
<td>78</td>
</tr>
<tr>
<td>5.1</td>
<td>Simulation Table</td>
<td>90</td>
</tr>
<tr>
<td>5.2</td>
<td>Generating a Pulse</td>
<td>92</td>
</tr>
<tr>
<td>6.1</td>
<td>Detection Data</td>
<td>104</td>
</tr>
<tr>
<td>6.2</td>
<td>Bridging vs. Output Stuck Detection</td>
<td>106</td>
</tr>
<tr>
<td>6.3</td>
<td>Verification Data</td>
<td>108</td>
</tr>
<tr>
<td>6.4</td>
<td>Diagnostic Data - Initial Response</td>
<td>112</td>
</tr>
<tr>
<td>6.5</td>
<td>Diagnostic Date - Entire Response</td>
<td>114</td>
</tr>
</tbody>
</table>
1.1 Statement of the Problem

In the past 5 years, production of digital integrated circuits has more than doubled. Current projections\(^1\) indicate that this trend will continue for at least the next five years. At the same time, the complexity of typical circuit board subassemblies has been rapidly increasing, due to both a greater number of IC packages per board and the heavy use of MSI and LSI microcircuits. These factors, coupled with the rapidly rising cost of highly skilled technical labor, have combined so that now the cost of testing and diagnosing digital circuit boards and components may often represent 20-25\% of a company's entire board-production costs.\(^2\) These high costs have focused attention on the need for efficient automated testing and accurate diagnosis of digital circuits. As a result, industry has strongly responded with the introduction of many automatic test systems. Academic research, however, has only indirectly attacked the problem with investigations mostly geared towards fault tolerant computing.\(^3\) The research as it relates to automated testing of digital circuit boards has been primarily directed toward finding
methods for generating test patterns for abstract logic networks, both to detect the failures likely to occur, and to distinguish between them. To represent the faults which occurred in physical circuits, several models were developed, among them the single gate failure model due to Spann and Hornbuckle, and the "line stuck at 0 or stuck at 1" model. In the single gate failure model, a fault is represented as the transformation of the good function of a particular gate into some other function of the gate input variables, whereas the "line stuck at 0 or stuck at 1" model represents a failure as causing a particular lead in the circuit to become permanently fixed at either the logical 0 or logical 1 level. Using these models, and computer programs to simulate the behavior of logic networks in the presence of these faults, it became possible to evaluate any given test sequence for its capability of detecting and distinguishing these faults on a circuit. In addition, faulty physical circuits could be accurately diagnosed if their output behavior matched that of one of the faults considered in the fault model.

Due to the increase in number of IC packages per board and the fairly high percentage of bad IC's, many manufacturers of products using digital circuit board assemblies have resorted to complete initial IC testing. This pre-testing has, in addition to lowering the initial fail rate of circuit
board assemblies at production test, changed the character of the typical faults encountered. Thus, 90% of the defects on newly manufactured circuit boards are not attributed to IC failures, but rather to some other error in the manufacturing process. Some of these failures are covered by the stuck-at-0, stuck-at-1 fault model, and some other unusual types of faults, such as intermittent faults, have been studied. However, in over 50% of these cases, the malfunction is traced to an unintended connection, or short, between two leads on the circuit. Because these typical bridging failure mechanisms have received very little attention in the literature, and so little is known about the problems in this area, this thesis will consider the detection and diagnosis of such failures on digital circuits. In particular, three main questions are addressed:

1. What is the behavior of typical bridging failure mechanisms and how can they be modeled.
2. Which specific bridging failures are covered by the stuck-at-zero, stuck-at-one fault model.
3. How effective is a test sequence designed to detect and diagnose stuck-at-0, stuck-at-1 failures in detecting and diagnosing bridging failures on a typical digital circuit board. Such a "typical" board may be a
sequential circuit realized with 50 integrated circuits, 50% of which are of MSI complexity.

In order to clarify a discussion of the problem, we will define some terms used throughout this thesis.

1.2 Definitions and Background

Many of the definitions made here have been drawn from the existing literature; most of them are similar to those made by Diephuis. Two major extensions to these definitions have been added. First, since his definitions apply only to combinational networks comprised of gates, extensions have been made to those definitions in order to facilitate the discussion of modular sequential networks. Second, we wish to address the bridging problem for complex digital circuits consisting of interconnected integrated circuit modules. Since the gate level definition of a logic network does not fully reflect the modular structure of present day digital circuits, we incorporate a two level scheme for dealing with them. This simplifies a discussion of bridging faults at the module (IC) level.

We consider a logic network to consist of a set of modules, primary inputs and primary outputs, interconnected by a set of inter-module leads. Each module, in turn, con-
sists of a set of gates, module input pins and module output pins interconnected by a set of intra-module leads. We will often refer to inter-module leads as e-leads, and intra-module leads as a-leads for convenience in the discussion.

Each gate has a set of inputs, each of which must be connected to a single intra-module lead, and an output, which must be connected to one or more intra-module leads. Associated with each gate is a Boolean function, which specifies for any time t, the Boolean value of the gate output for each possible set of gate-input values at time t-Δ, where Δ is a delay associated with the gate. We express this function as 

\[ G(t) = f(a_1(t-\Delta), a_2(t-\Delta), \ldots, a_n(t-\Delta)) \]

This is illustrated in figure 1.1. A module-input pin must be connected to one or more a-leads and exactly one e-lead. At any time, the value of an input pin, and all a-leads connected to it is equal to the value of its connected e-lead. Similarly, a module output pin connects one or more a-leads with a number of e-leads, and the value of a module output pin and its connected e-leads is equal to the value of the connected a-lead.

In the following definitions, we will assume that the number of gates in a network is \( p \), the number of primary inputs is \( n \), and the number of primary outputs is \( m \).
Figure 1.1 A Gate
A primary input (external input) must be connected to one or more inter-module leads. At any time \( t \), a single Boolean value may be assigned to a primary input; the values of all e-leads connected to the primary input are equal to this value. The primary inputs to a logic network will be denoted by the vector \( (x_1, x_2, x_3, \ldots, x_n) (t) \), where \( x_i(t) \) is the value of the \( i^{th} \) primary input at time \( t \) for \( 1 \leq i \leq n \). We will require that \( x_i(t + \Delta) = x_i(t) \) for all \( i = 1 \) to \( n \) whenever \( G_j(t + \Delta) \neq G_j(t) \) for any \( j \), \( j = 1 \) to \( p \). This requirement simply states that the input vector will not change until the network is in a stable state. It will be assumed that no critical races or oscillations result from the application of primary input values to a fault-free network at any time.

A primary output (external output) is connected to exactly one inter-module lead. The value of a primary output at any time is equal to the value of its connected inter-module lead and is always observable. The primary outputs of a network will be denoted by a vector \( (y_1, y_2, \ldots, y_m)(t) \) or \( \overline{Y}(t) \), where \( y_i(t), 1 \leq i \leq m \) is the value of the \( i^{th} \) primary output at time \( t \).

An intra-module lead must be connected between a gate input and a gate output, between a module input pin and a gate input, or between a gate output and a module output pin. Similarly, an inter-module lead must connect a module input pin to a module output pin, a primary input to a module input
pin, or a module output pin to a primary output. These definitions are illustrated by the simple logic network shown in figure 1.2.

The following additional definitions will be used throughout most of this thesis; other definitions relating to specific topics will be made as needed.

**Input vector:** an n-tuple of constant Boolean values $(v_1, \ldots, v_n)(t) = \overline{v}(t)$ at time $t$. Applying this vector $\overline{v}(t)$ to a network means making the primary inputs $x_i$ take on the value $v_i$ for $1 \leq i \leq n$ at time $t$.

**Input sequence:** A finite, ordered set of input vectors $(\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_q)$. Applying the input sequence $(\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_q)$ to a network means successively applying input vectors $\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_q$ subject to the previously mentioned constraint that a new input vector is not applied until the network is stable.

**Path:** an ordered set of e-leads and a-leads $(w_1, \ldots, w_r) = w$ such that if $w_{i+1}$ is connected to the output of a gate, $w_i$ is connected to an input of that gate for all $i$, $1 \leq i \leq r$. When $w_i$ is an e-lead connected to a module input pin, then $w_{i+1}$ is an a-lead connected to the same module input pin. Similarly, when $w_i$ is an a-lead connected to a module output pin, $w_{i+1}$ is an e-lead connected to the same module output pin. A path $(w_1, \ldots, w_r)$ is said to be from a gate (or module
Figure 1.2. A Logic Network
input or output pin, or primary input) \( g_i \) to a gate (or module input or output pin, or primary output) \( g_j \) if \( w_1 \) is connected to an output of \( g_i \) and \( w_r \) is connected to an output \( g_j \). A gate (or module input or output pin) \( g \) is said to be on a path \((w_1, \ldots, w_r)\) if there exists \( i, 1 \leq i \leq r \) such that \( w_i \) is an input or an output at gate \( g \). A module is said to be on a path if any of its input or output pins are on that path.

**Feedback path:** a path \((w_1, \ldots, w_r)\) where \( r \neq 1 \) such that either:

1. \( w_1 \) is connected to the output of a gate and \( w_r \) is connected to one of its inputs, or
2. \( w_1 \) and \( w_r \) are connected to the same module input or output pin.

**Combinational network:** a logic network with no feedback paths.

**Fault:** any change in a logic network \( N \) which causes the logical function realized by any part or parts of that network to change. A fault is a **combinational fault** if, in the logic network which results after the fault occurs, there is no feedback path which was not present in the original network. A fault will be denoted by the symbol \( f \). \( N(s) \) will
denote the sequence of output vectors given the input sequence \( \bar{s} \), while \( N(f, \bar{s}) \) will denote the sequence at output vectors given the input sequence \( \bar{s} \) and the occurrence of the fault \( f \).

**Bridging fault:** a fault in a logic network which forces the values of two gate outputs to be equal; this value is a function of the inputs to both these gates.

**Stuck-at fault:** a fault in a logic network which causes some lead in the network to permanently assume the logical 0 or 1 value. Unless explicitly stated otherwise, whenever we use the term bridging or stuck-at fault we refer to a single occurrence of either fault.

A test sequence for a fault \( f \), is a sequence of input vectors, \( \bar{s} \), such that \( N(\bar{s}) \neq N(f, \bar{s}) \); in other words, the output sequence differs from the one in the fault free network. Thus, the fault \( f \) is said to be detected by this test sequence. Finally, we say that a test sequence distinguishes \( f_1 \) and \( f_2 \) if \( N(f_1, \bar{s}) \neq N(f_2, \bar{s}) \).

In order to analyze the failure mechanisms of digital circuits, we must limit the possible faults which we consider. The faults which are under consideration at any time are collected together into a fault set for a network denoted \( F(N) \). The types of faults which are considered possible and are therefore used to determine the fault set for a given
network, will be called the fault model.

Most research in fault detection has used the single stuck-at fault model as the basis for analysis, and some studies have also considered a multiple stuck model. In this thesis, the type of fault we will be mainly concerned with is an unwanted connection on a digital circuit board. As will be shown in chapters 3 and 4, this phenomenon will require a new fault model.

1.3 Outline of this Thesis

The work done in this thesis has been divided into four major tasks:

1. Establish an accurate model for shorting behavior in digital circuits through experimentation with several types of integrated circuit gates.
2. Using this model, determine which types of shorts will be detected by a test for stuck-at-0, stuck-at-1 failures.
3. Design and implement a computer program to simulate the effect of shorts and stuck-at faults on large asynchronous sequential circuits (a bridging and stuck-at fault simulator).
4. Using several sample networks, quantitatively
determine the relative percentage detection of stuck faults and bridging faults. Also, for these networks, verify the accuracy of the model by matching the response patterns of the physical faulty circuit with the response patterns N(f, s) of the modeled faulty network. Finally, present quantitative results indicating to what extent the stuck-at-0, stuck-at-1 fault model is useful in diagnosing shorts on these digital circuits.

Chapters 2 and 3 are concerned with establishing an accurate model for bridging failures. In chapter 2, the single stuck-at-0, stuck-at-1 fault model is described in some detail. This is done for several reasons:

1. The stuck-at-0, stuck-at-1 fault model is by far the most widely used in the literature, and thus forms a basis for comparison with other models.
2. There are many faults which may occur on newly assembled physical circuits which are exactly modeled by the stuck-at-0, stuck-at-1 fault model. These are presented in a list of possible electrical failures which are modeled by the stuck-at-0, stuck-at-1 model. Chapter 3 establishes the basis for the bridging fault model by presenting an experimental study of several
TTL Integrated Circuit gates under shorting conditions. The limitations of the fault model are also discussed.

In chapter 4, some analytical results are presented concerning the detection of bridging faults by tests designed solely to detect stuck-at-0, stuck-at-1 faults. These results are mostly applicable to combinational logic networks, but give some insight into the problems associated with analyzing asynchronous sequential networks. Arguments are also presented which give some expected upper and lower bounds to the percentage detection of bridging faults.

Chapters 5 and 6 deal with the design and implementation of the bridging and stuck-at fault simulator, the procedures used to verify the accuracy of the bridging fault model, and quantitative experimental results relating detection and diagnosis using the bridging and stuck-at fault model. The problems of simulating large sequential circuits composed of interconnected MSI and LSI modules will be considered, and the methods employed in the solution of these problems will be presented. The bridging fault set will be limited by considering only the effects of shorts between adjacent integrated circuit pins, but it will be shown that any likely bridging failure between intermodule leads can also be simulated.
Finally, the thesis will be ended with a summary of results and conclusions.
CHAPTER 2
THE STUCK-AT-0, STUCK-AT-1 MODEL

2.1 Description of the Model

Published research in the field of fault detection and diagnosis has almost universally used stuck-at-0, stuck-at-1 fault models in determining the set of faults to be considered for analysis. In such fault models, electrical faults which occur on physical networks are assumed to cause some lead in the circuit to become permanently fixed at the logical 0 or logical 1 level. Thus the logical function of a faulty physical circuit corresponds to the function of logic network with one or more of its leads stuck. The "stuck-at" fault models offer several advantages for analysis, namely, that stuck faults cannot add feedback to a circuit, and in general cannot affect the topology of the network. Thus, combinatorial circuits remain combinatorial and sequential circuits cannot increase in number of states with the insertion of stuck faults.

Certain faults, which are indistinguishable when observed at the primary outputs, are said to be equivalent. Thus we partition a fault set into a set of fault equivalence classes. As an example, the input of a NAND gate stuck at 0 is equivalent to its output stuck at 1.
In the discussions relating stuck faults with bridging faults in chapters 4, 5, and 6, we will usually want to distinguish between an input stuck and an output stuck. An input stuck affects only the (one) a-lead connected to an input of the gate in question, whereas an output stuck affects all a-leads connected to the output of a gate. Since we will always deal with classes of equivalent faults rather than individual faults, we will use the term "output stuck" whenever referring to a fault class which contains at least one output stuck fault. This is useful because the behavior of shorts can be related to that of output stucks, as will be shown in chapter 4. The stuck-at-0, stuck-at-1 fault model accounts for the function of many electrical faults on a physical circuit. Examples of faults which are accurately modeled for physical circuit boards made up of standard Transistor Transistor Logic (TTL) elements are described below. In this description, as well as throughout this thesis, we use the positive logic convention, i.e. a logical 0 represents the lower of the two normal output voltages, and a logical 1 represents the higher level. Some of the faults accurately modeled are:
1. An open connection between the IC solder pads and the pins of an IC package, or between the pins and the external circuitry, modeled as one or more leads stuck-at-1.

2. A broken wire, modeled by one or more leads stuck-at-1.

3. An external short to $V_{cc}$ or Ground, modeled as the connected output pin stuck-at-1 or 0.

4. In figure 2.1 showing the circuit of a TTL gate, a collector-emitter short on $Q_1$ or $Q_2$ would be modeled by the output lead stuck-at-1 or stuck-at-0.

5. An open circuit between collector and emitter of $Q_2$, modeled by the output lead stuck-at-1.

6. A base to emitter short on $Q_2$, modeled by the output stuck-at-1.

7. A base to emitter open on the input transistor $Q_3$, corresponding to an input lead stuck-at-1 or 0.

Of these different failure mechanisms, the one most likely to be present in an initial test of a newly manufactured circuit board are the first three. In particular, each of these three mechanisms involve a mechanical defect caused by the manufacturing and assembly process. The other types of failures are those which can either be detected
Figure 2.1. A TTL NAND Gate
when an IC is tested individually, or later when the circuit is tested over an extended period of time.

The stuck-at faults which are most common, then, correspond to an input or output pin of an IC stuck-at-0 or stuck-at-1. The next section discusses which stuck-at faults should be considered in a logic network when designing test sequences for digital circuit boards. In particular, we relate the gate-level stuck model to the modular network structure we have defined in chapter 1, and show that single failures at the module level do not always correspond to single stuck-at faults at the gate level.

2.2 Use of the Stuck-at Model for Circuit Testing

Figure 2.2 shows a module and part of a logic network. This module, which determines the AND, NAND, OR, and NOR functions of A and B, will be used to illustrate the application of the stuck-at fault model to circuits which are composed at interconnected modules. The notation used is as follows:

1. ➔ indicates connection to a primary output or the input pin of some module.
2. ➔ indicates the source of a signal from a primary input, or output pin of some other module in the network.
Figure 2.2. Module with internal fanout
3. A and B are module input pins, while C, D, E and F are the module output pins.

Note that in an electrical network where the module shown corresponds to a physical IC, an open connection at input pin A cannot be modeled as a single gate input or output stuck. Logically, this fault corresponds to both inputs m and n stuck. Similarly, an open connection at output pin C corresponds to leads p, q, and r stuck at 1, without affecting the output of gate X. In order to model these failures accurately in the fault simulator implementation, and yet use only the single stuck fault model, we have constructed the notion of logical module input pins and logical module output pins (see chapter 1). In the discussion of the fault simulator in chapter 5, the only faults which are considered occur at module input and output pins. Thus we define a module input pin stuck as causing all intra-module leads connected to the module input pin to become fixed in value.

For the purposes of this thesis, it is useful to take into account the electrical properties of DTL and TTL integrated circuits by distinguishing between the stuck-at-0 and stuck-at-1 case for module output pins as follows:
A module output pin stuck at 1 causes all inter-module leads connected to the module output pin to become fixed at a logical 1, whereas a module output pin stuck at 0 causes the value of the intra-module lead connected to a gate output inside the module to become fixed at a logical 0. For example, module output pin C stuck-at-0 would cause the output of X to become fixed at 0, which would, in turn, cause all inter-module leads from C to be stuck-at-0 and all inter-module leads from D to be stuck-at-1. On the other hand, module output pin C stuck at 1 would affect only p, q, and r, and not D. This different treatment of the stuck-at-0 and stuck-at-1 cases is useful because, physically, an open connection at an IC output pin will cause all the IC input pins connected to it to become fixed at the logical 1 level. At the same time, this fault has no effect internal to the IC. However, a short to ground anywhere on a signal path will cause the fixed 0 logic level to feedback into the IC and possibly affect other outputs of that IC. Note that the distinction made between the stuck-at-0 and stuck-at-1 cases vanishes when there is no fanout from a gate connected to a module output pin. Thus, in figure 2.3, output pin C stuck-at-0 and stuck-at-1 would be treated identically.

The single stuck-at-0, stuck-at-1 model has been used extensively in evaluating the merit of test sequences for
Figure 2.3. Module with no Internal Fanout
detecting likely failures in digital circuits. In order to detect all stuck-at faults, it is necessary for every possible faulty node in the circuit to assume both the logical 0 and 1 state at some time during the application of the test sequence; thus, it has been reasoned that the logic is well exercised, and most multiple stuck faults, as well as other faults not modeled, will be detected as well.\textsuperscript{14} In chapter 4, we will discuss this argument in relation to bridging faults for the case of combinatorial circuits. While no definitive conclusions are drawn for sequential networks, arguments are presented to indicate how well test sequences designed for stuck-at-0, stuck-at-1 faults might detect arbitrary bridging faults, and in addition, experimental results are given in chapter 6 which do indicate a relation between the detection of stuck and bridging failures.

The next chapter will discuss the electrical behavior observed when the outputs of various integrated circuit gates are shorted together. Based on this behavior, a bridging fault model will be presented, which will be used in the bridging fault simulator.
CHAPTER 3
THE BRIDGING FAULT MODEL

3.1 Introduction

In this chapter we consider the basis for the Bridging Fault Model, in terms of the static electrical behavior of various integrated circuits currently used in logic design. In particular, we discuss the currently popular TTL technology with reference to its behavior under shorting conditions. This technology was chosen for study for two major reasons. First, the current extensive use of TTL integrated circuits on many digital circuit boards makes these circuits readily available for experimentation and the verification of our models, and allows wide application of the results. Also, shorting two TTL outputs together is not an accepted standard design procedure, as it is in RTL, DTL and ECL technologies. Thus, the result of shorting these devices is open to some question.

Since the circuit parameters for most integrated circuits are not published, and the operation of the circuit is only specified within certain guaranteed limits, the approach taken in this study was primarily experimental, except where the nature of the circuit clearly indicated the behavior to be expected.
3.2 Bridging Mechanisms on Electrical Circuits

There are many possible ways in which a short may occur on a digital circuit board, and most of them are the direct result of errors in the manufacturing or assembly process. The most common types of shorts are:

1. Shorts between adjacent pins of the same IC package. These may occur within the IC itself, as bridges across the tiny solder pads on the chip. Most often, however, (and these are the only cases possible if the chip has been pre-tested) the shorts occur external to the IC. On boards in which IC's or IC sockets are soldered onto the printed circuit, solder often bridges the gap between two pins. In the cases where wire wrapping is used, a frequent source of shorts is tiny chips of bare wire, and the easily bent wire wrap pins themselves.

2. Shorts between closely spaced printed circuit tracks, again caused by the wave soldering process, or mechanical mishandling, or bare wire fragments.

3. Two wires shorted together due to either worn out or burned insulation.
In any case, no matter which two points are shorted together, the problem can be viewed as two outputs tied together, since every lead must be connected to the output of some gate.

In the next section we will present discussions of behavior for the TTL logic circuit family. We will conclude the chapter with a summary of the model for TTL circuits, which will be used throughout the remainder of the thesis, particularly in the implementation of the bridging fault simulator.

3.3 Electrical Characteristics of Shorted Transistor-Transistor-Logic Devices

A basic TTL circuit is shown in figure 3.1. This circuit is usually characterized by a single multi-emitter transistor input and an active pullup transistor output which gives current gain drive for switching in both directions. Previously, the TTL family of logic circuits has been excluded from a consideration of wired logic properties because of the active pullup and pulldown characteristics of this typical totem pole output stage. Even though TTL gates with open collector output stages are available and commonly used to realize wire logic functions, analysis of these outputs is essentially identical to the passive
Figure 3.1. Typical TTL Circuit
pullup output used in DTL circuitry, which is commonly used to implement the wired AND function. Thus the open collector gate will be ignored in this discussion. In figure 3.2 we consider the circuit which results when two TTL outputs are shorted together. For the conditions where the separate output values of each gate are both 0 or both 1, we see that the behavior of the circuit is the same whether or not the short exists, independent of loading conditions. The output circuits, being identical, will each stabilize with an output voltage in the low (0 - .4 v) or high (2.4 - 3.8 v) range.

The problem with modeling the shorted behavior of TTL circuits arises when separately the logical outputs have different values. Clearly, because of symmetry, it is irrelevant in this analysis which gate asserts the high or low value. Thus in figure 3.2 we consider G₁ to be in the logical 1 state and G₂ to be in the logical 0 state. This means that transistor Q₁ is saturated or operating as an emitter follower, while Q₂ is cutoff. Conversely, in G₂, Q₃ is cutoff and Q₄ is saturated. In this case, the resultant voltage level is a function of several parameters:
Figure 3.2. Two Outputs Shorted Together
1. the current limiting resistor $R_l$,
2. the saturation resistance of transistor $Q_4$ and $Q_4$,
   and
3. the total number of TTL circuits driven by both outputs.

Using the following values, which have been published as typical for standard TTL,\textsuperscript{19,20} we use the simplified circuit of figure 3.3 to analyze the situation.

\[ R_{Q4} = R_{Q4} = 10\Omega \]
\[ I_{\text{sink}} = 20 \text{ loads} \times 1.6 \text{ mA max/load} = 32 \text{ mA} \]
\[ R_l = 130\Omega \]
\[ V_{cc} = 5.0 \text{ v} \]

By specifying the current from the loads, it has been assumed that the result will be a low voltage, which is consistent with the results. Thus the saturation current $I_s$ is:

\[ I_s = \frac{V_{cc} - 0.6 - I_{\text{sink}}}{R_{Q4}} \]

\[ I_s = \frac{R_l + R_{Q4}}{R_{Q4}} \]
Figure 3.3. Static Model of Two Shorted Outputs
Substituting values we get:

\[ I_s = (5 - .6 - .32)/(150) \]

\[ I_s = 27 \text{ mA} \]

Thus, the resultant output voltage \( V_o \) is:

\[ V_o = (I_{sink} + I_s)R_{Q4} = .59 \text{ volt} \]

This result is less than the maximum level which specifies a logical 0 in TTL logic \( V_{IL} \) (max) = .8v\(^5\) so we say that the current sinking capability of the low level gate will dominate over the current driving of the high level, or that the function realized by shorting the two outputs is the logical 'AND' of the two separate values.

It is worthwhile to note here, that manufacturers of TTL integrated circuits caution against wiring the outputs of two gates together as a standard design procedure. There are several reasons for this:

1. Very high power dissipation results from this connection. One of the gates may eventually be destroyed by high current damage.
2. Even though the result of wiring two outputs together is predictable, adding a third or fourth output will certainly cause non-logical (threshold) behavior.

3. Since the resultant voltage level is highly dependent on $R_1$ and $R_{Q4}$, large variations in these values may cause non-logical behavior.

We have not made a worst case analysis. Specifications published indicate that the short circuit (to ground) current at a gate in the high state may be as great as 55 mA.\textsuperscript{19} This implies that if $V_o$ is tied to ground in figure 3.3, $I_s = 55$ mA and $(R_1+R_{Q1}) = (V_{cc} - .6)/.055$, or $R_1+R_{Q1} = 80\Omega$. At the same time, a TTL gate is only guaranteed to sink 16 mA at .4 volts in the low state, which indicates that $R_{Q4}$ may be as high as 25\Omega. Using these new values for $(R_1+R_{Q1})$ and $R_{Q4}$ we have

$$I_s = \frac{(V_{cc} - .6 - I_{sink}R_{Q4})}{(R_1+R_{Q1}+R_{Q4})}$$

or

$$I_s = \frac{(5 - .6 - .8)}{105}$$

$\therefore I_s = 34$ mA

Thus $V_o = (I_{sink}+I_s) \cdot R_{Q4} = 1.65$ v
This value is in the middle of the threshold region for TTL. However, at this level, several assumptions that we have made, notably that 1.6 mA sink current and the linear resistance characteristics of \( Q_4 \), are incorrect. The point here is that we are considering worst case conditions on all possible gates connected (i.e. worst case sink current from a maximum number of inputs, worst case (max) high output driving capability on \( G_1 \), and worst case low level current sinking ability in \( G_2 \)). These conditions are very rarely met in practice, especially under normal room temperature circuit testing conditions, and do not justify a more complex model. In particular, the worst case specifications generally apply at the extremes of operating temperature ranges.\(^5\) The dynamic characteristics of this shorting connection have not been considered since it is assumed that new inputs are applied at a rate slow enough to be essentially static.

To verify these results, standard TTL NAND gates were used to implement the circuit of figure 3.4, and for all gates tested, the output voltage \( V_o \) was always less than .8 volts. Even though these results apply to the large majority of shorted gates in a typical circuit, there are several special cases which deserve further attention. Specifically, it is necessary to separately examine the
Figure 3.4. Test Circuit for Two Shorted Outputs
behavior which occurs when one of the two outputs shorted together was also the input to one of the gates. This type of situation is a common occurrence on small scale integration (SSI) IC's, where an input and the output of the same gate are frequently adjacent pins and thus likely to be shorted.

For the case of the NAND and NOR gates, this situation is shown in figure 3.5. As long as the output of X is 0, we have determined that the resultant value at m is 0. But if X = 1 and A = 1 for the NAND gate, 11 \rightarrow 0. However, a 0 at m forces X = 0 and 01 \rightarrow 1. Thus it would seem that this fault might cause the output to oscillate. A similar situation exists with the NOR gate when X = 1 and B = 0. It is not feasible to specify that a test sequence avoid this problem by not allowing X = 1 and A = 1 at the same time, since the 11 input combination to the NAND gate is necessary to detect its output stuck-at-1.

Figure 3.6 shows the circuits which were employed for this experiment, which was carried out for four different types of two input gates, namely the NAND, AND, NOR, and OR gates. The circuits employed were identical except for the type of gate. In the experiment, point A was shorted to point C, and all four input combinations were repetitively applied at X and Y. Figure 3.7 shows the results obtained
Figure 3.5. Short from an Input to the Output of a NAND/NOR Gate
Figure 3.4. Test Circuit for Shorts between the Input and Output of a Gate
Figure 3.7. Oscilloscope Waveforms for Shorted Outputs
as a sketch of oscilloscope waveforms. In this figure A and B are the logical input values applied by gates m and n (Figure 3.5) to shorted gate s. C (or $\overline{C}$) is a sketch of the waveform at the shorted connection. D (or $\overline{D}$) is a sketch of the waveform at the output of gate p. D is important because it indicates how the result of the short is "read" by a TTL gate. The following interesting points should be noted about these results:

1. The NAND gate and NOR gate do not usually oscillate when the shorted input is sensitized and $A = 1$. Even though the actual voltage level at the shorted output is in the threshold region, the experimental results indicate that this level is consistently "read" as a logical 1 by a TTL gate. Some reservations must be made, however, when the shorted leads are connected as clocking or data inputs to sequential modules, in which case noise spikes on the marginal level may cause false switching.

2. The AND or OR gates exhibit "lockout". That is, once the shorted connection is set into the low state by the output of the shorted gate, the A input cannot set it high. In the case of the OR gate, the cycle is reset by $B = 1$, but once the AND gate is "locked out" it
remains so until power is turned off. In this case, the normal 'AND' function of two shorted signal lines applies, and correctly accounts for the observed behavior. In the case of a short between the input and output of an AND gate, the positive (non-inverting) feedback of the low value prevents the "driving" gate (m of figure 3.6), from asserting a logical 1. This keeps the circuit in a trapped state. The OR gate is similar except that assertion of the logical 1 on the unshorted input breaks the positive feedback path and allows the result to be 1 for A and B = 1.

3.4 Logic Representation of the Bridging Fault Model

Using the previous results of experimentation, we present here the logical model used to represent bridging faults in TTL circuits. This representation will be used in the next chapter in discussing the detection of bridging results. In addition, it forms the basis of the bridging fault simulator design and implementation presented in chapter 5. The different types of faults which can occur are as follows:

1. Normal 'AND' Bridge

This model is used in all cases for which the two leads shorted together are not connected to the input and output of the same NAND or NOR gate. For TTL circuits, the
Figure 3.8. Function of AND Bridge

Figure 3.9. Function of NAND Switch

Figure 3.10. Function of Nor Switch
functional result of a short is that the leads contain a logical signal which is the 'AND' of the two separate signals. This is shown in figure 3.8.

2. **Short Between the Output and an Input of a NAND Gate**

   This fault causes the input value to be propagated through the gate without regard to the values on the other inputs, as shown in figure 3.9.

3. **Short Between the Output and an Input of a NOR Gate**

   In this case, the shorted signal line has the value

   \[ x = A \cdot A_2 + \ldots + A_n \]

   where A is the value of the lead connected to the output of the other gate, and \( A_2, \ldots, A_n \) are the values of the unshorted input leads. This is shown in figure 3.10.

   The first of these three models, by far the most often used, will be referred to as the 'AND' bridge or, more simply, the **AND fault**. The second will be called a **NAND switch**, and the third a **NOR switch** fault.

   In the discussion of chapter 4 it will be assumed that, for the purposes of modeling the behavior of an electrical short, the appropriate model can be chosen by the bridging fault simulator. The implementation of the bridging fault
simulator will be discussed in detail in chapter 5. The next chapter and chapter 6 will consider the relation between the bridging and stuck fault models from the viewpoint of detection and diagnosis. In particular, we are interested in determining to what extent bridging faults are detected or diagnosed by a test program generated using the single stuck-at-0 - stuck-at-1 fault model. This will give insight concerning the usefulness of the bridging fault model in generating test sequences for logic circuits as well as its importance in automatic diagnosis of such circuits.
CHAPTER 4

ANALYTICAL RESULTS RELATING THE BRIDGING
AND STUCK-AT FAULT MODELS

4.1 Introduction

Most research in the automatic generation of tests for
digital circuits has used the single stuck-at-0, stuck-at-1
fault model as the basis for the fault set. Bridging faults
are in general more difficult to analyze than stuck-at faults,
primarily because they alter the topology of the network, and
may make the network more complex by the introduction of
additional states.

This chapter will consider the problem of bridging
fault detection from two points of view. First, in section
2, we will discuss certain bridging failures in combinational
networks and show that these will always be detected by a
single stuck fault detection test set. Some of the theorems
in this section are similar to those stated elsewhere, but
are proven here because the assumptions, which needed to be
made, were not clearly pointed out. In section 3 we will
consider a small network and the applicability of our
theoretical results to this network. Unfortunately, no
exact quantitative relationship has been found for the
detection of single bridging faults vs. stuck-at faults.
In order to gain some insight into this problem, however, a different point of view is adopted in section 4. Here we present an upper and lower bound on detection of arbitrary shorts that can be expected given certain assumptions about the network. Finally, section 5 outlines simple network transformations for which a bridging fault in the original network corresponds to a stuck-at fault in the transformed network. Such a transformation has the advantage that it is an extension to previous research in fault detection and diagnosis, and allows methods developed for generating stuck fault detection tests to be used in generating tests of for bridging faults.

4.2 Detection of Bridging Faults in Combinational Logic Networks

In this section we examine certain bridging failures which are always detected by stuck-at test sets. We adopt some of the definitions and notation used by Mei,¹² and for completeness have included some theorems stated by him. These theorems have been marked with a reference.

Even though the bridging fault usually performs the AND function, as discussed earlier, this analysis is equally applicable to fault models in which the result of a short is the OR function. By duality, the only necessary modifications would be certain changes in polarity.
We make the following definitions:

A Boolean function is **elementary in input** \( x \) if the function can be expressed in the form \( x \cdot F_1 \) or \( x + F_2 \) where \( F_1 \) and \( F_2 \) are vacuous in \( x \).

A function is elementary if it is elementary in all of its input variables.

An **elementary gate** is a gate whose function is elementary.

An **input bridging fault** (IBF) is created by a short across two input leads of an elementary gate.

As an example, the function \( a(b+c) \) is elementary in \( a \) only. Of the gates usually employed in logic design, most (NAND, AND, OR, NOR) are elementary while XOR is not. A bridging fault between two leads \( x_1 \) and \( x_2 \) will be denoted \( * (x_1 / x_2) \), except in the cases where the fault bridges the input and output of a NAND or NOR gate. In those cases we will use the notation \( A(x_1/x_2) \) for NAND switch and \( 0(x_1/x_2) \) for NOR switch, where \( x_1 \) is connected to an input and \( x_2 \) is connected to the output of the gate. We define several additional terms:
Fanout occurs when the output of one gate is connected to more than one input lead.

Fanout from an input $x$ occurs when the output connected to $x$ has fanout. To define reconvergent fanout from an input, we use the diagram of figure 4.1. We say that reconvergent fanout from input $x$ occurs if there exists a path from the output of gate B to an input of a gate C and there exists another path to C from A which does not include lead y.

In the case of an IBF $(x_1/x_2)$ on a NAND or AND gate with no fanout from either input, the fault is undetectable since the fault performs the same function as the gate itself, and the operation of $(x_1/x_2)$ is idempotent. An example of this is shown in figure 4.2.

We say that a function $f(x_1, \ldots, x_n)$ is sensitized to an input $x_i$ $1 \leq i \leq n$ when $f(c_1, \ldots, c_i, \ldots, c_n) \neq f(c_1, \ldots, \overline{c_i}, \ldots, c_n)$ for some set of values $c_j$, $1 \leq j \leq n$, $j \neq i$. A sensitized path for an input vector $\overline{v}$ is a path for which the function of each gate on the path is sensitized to an input lead on that path. We now state the conditions for detection of an IBF:

**Theorem 4.1.** An IBF $(x_1/x_2)$ with no fanout from $x_1$ or $x_2$ can be detected by an input vector $\overline{v}$ if and only if, for
Figure 4.1. Reconvergent Fanout

Figure 4.2 Undetectable IBF
that input vector, \( x_1 \neq x_2 \) and the output of the gate is sensitized to the input which is 1.

Proof: If \( x_1 \neq x_2 \) then \( x_1 \cdot x_2 = 0 \) which is the function performed by the fault. Now if \( x_1 = 1 \) then the gate is sensitized to \( x_1 \) (the output depends on the value of \( x_1 \)). Since the bridging fault changes the effective value of \( x_1 \), the output of the gate is the opposite of the fault free case, and can therefore be detected. A similar argument holds for \( x_2 = 1 \). The converse (only if) is proven in the following: Since there is no fanout from either \( x_1 \) or \( x_2 \), the bridging fault can only be detected through the output of the gate. Clearly, \( *(x_1/x_2) \) cannot be detected when \( x_1 = x_2 \), since the fault has no effect under these conditions. Now, for \( x_1 \neq x_2 \), if the function is not sensitized to either input, then the fault cannot be detected, since a change in any one input value will not affect the function. Finally, if the function is sensitized to the input which is 0, the fault is not detected since the bridge fault never modifies a 0 value. This leaves only the case specified in the theorem.

Q.E.D.
Figure 4.3 shows an example where the conditions are met. If \( x_3 \) were 1, though, the output would not be sensitized to the AND bridge and the fault could not be detected. The condition for bridging fault detection given in the above theorem suggests a strong relation to stuck-at fault detection, and this is given in the following theorem:

**Theorem 4.2:** Every detectable IBF \( \ast(x_1/x_2) \) with no fanout from \( x_1 \) or \( x_2 \) is detected by an input stuck test on \( x_1 \) or \( x_2 \).

**Proof:** There are only six elementary functions of two variables for which an IBF with no fanout is detectable, namely \( x_1+x_2, x_1\overline{x}_2, x_1\overline{x}_2 \) and the complements of these. Clearly, the complementation has no effect, so we consider the three cases mentioned.

1. \( x_1+x_2 \)

In this case \( x_1 = 1, x_2 = 0 \) is the test for input \( x_1 \) stuck at 0. Since it fulfills the requirements of Thm.4.1, it is also a test for \( \ast(x_1/x_2) \).

2. \( x_1\overline{x}_2 \)

Here \( x_1 = 1, x_2 = 0 \) is the test for input \( x_1 \) stuck-at-0 and is a test for \( \ast(x_1/x_2) \).
Figure 4.3. Sensitizing the Wired Function
3. $x_1 + \overline{x_2}$

Here $x_1 = 0, x_2 = 1$ is the test for input $x_2$ stuck-at-1 and is a test for *(x$_1$/x$_2$)

Q.E.D.

So far we have only mentioned IBF's which have no fanout on either input. We can make similar statements about IBF's with fanout, but must take into account the possibility of reconvergent fanout which might interfere with detection, as pointed out by Shertz. One such example of a circuit in which an input bridging fault is not detected by a stuck-at test set has been given by Friedman. The following theorems precisely define the conditions for detection of input bridging faults with fanout.

**Theorem 4.3:** An IBF *(x$_1$/x$_2$) on an OR/NOR gate with fanout on one of the shorted inputs is always detected by an input stuck test.

**Proof:** For an OR/NOR gate, we have shown that inputs $x_1 = 0, x_2 = 1$ and $x_1 = 1, x_2 = 0$ are necessary to detect $x_2$ stuck-at-0.
and \( x_1 \) stuck-at-0 respectively. Since there is fanout from only one input lead, there exists an input stuck test for which the fanout lead is not affected; i.e. the value of the fanout lead is 0. This test, which detects the other input stuck-at-0, also detects the IBF because the output is sensitized to the wired function and the fault cannot propagate from the fanout point.

Q.E.D.

Finally, we present a theorem for the case where there is fanout on both inputs that are shorted.

**Theorem 4.4:** An IBF \( *(x_1/x_2) \) on an OR/NOR gate with fanout on both the shorted inputs will be detected by an input stuck test if the fanout from at least one of the shorted inputs is non-reconvergent.

**Proof:** By the definition of non-reconvergent fanout, a faulty signal on the non-reconvergent fanout lead cannot affect the signal which propagates from the output of the IBF shorted gate. This fanout can thus be disregarded and since we are left with one fanout input, Theorem 4.3 applies.

Q.E.D.
The application of the above theorems to the problem of generating tests to detect faults is obvious: whenever the IBF's mentioned above are encountered, they may be ignored for the purpose of test generation, as long as all input stuck faults are detected.

As we discussed in chapter 3, another common type of short may exist between the input and output of a NAND or NOR gate. Using the model presented in that chapter, we find that the detection of these faults is particularly simple.

**Theorem 4.5:** A NAND switch fault $A(x_1/x_2)$ is detected by a test for input $x_1$ stuck-at-1 and a NOR switch fault is similarly detected by a test for input $x_1$ stuck-at-0.

**Proof:** The test for a NAND input stuck-at-1 involves setting all other inputs to 1 and setting the input to be tested to 0. In this case the correct output is 1, but the output in the presence of $A(x_1/x_2)$ is 0. Figure 4.4 illustrates this. The argument for a NOR switch is similar.

Q.E.D.

Finally, there is one more class of bridging faults which is always detected by a single stuck test set. These
Figure 4.4. NAND Switch
faults are formed by shorting two leads in a fanout free
subnetwork, with the following restrictions:

1. One of the leads shorted must be on the path from
the other shorted lead to the primary output.
2. There must be an odd number of inversions (>1) in
the path from one shorted lead to another.

This type of fault, called an Inverting Feedback Bridging
Fault (IFBF)\textsuperscript{12} is illustrated in figure 4.5. In this
example, the detection scheme for the IFBF is to sensitize
the output A to the input $x_1$ of B and set $x_1=0$. This
causes the output A to detect $*(x_1/x_2)$ since the good value
of 1 will not appear. But this test is the same as the
test necessary for input $x_1$ stuck-at-1. This forms the
basis for the final theorem:

\textbf{Theorem 4.6:}\textsuperscript{12} \textbf{ALL IFBF's in a network are detected by a}
\textbf{single stuck-at test set.}

\textbf{Proof:} Consider the shorted input of the gate farthest from
the primary output (this would be $x_1$ in figure 4.5) and call
it $x_1$. Now, the test for $x_1$ stuck-at-1 must set $x_1=0$ and
Figure 4.5. An IFBF
sensitize the path to the primary output. This sensitization, and the odd number of inversions guarantee that the output, \( x_2 \), of the other shorted gate would normally be 1 but is now 0 due to \( \neg(x_1 \land x_2) \). Since the output is sensitized to the path from \( x_2 \), the faulty value will propagate to the output and thus the fault will be detected.

Q.E.D.

Paraphrased, this theorem states that any short on the same path of a fanout free subnetwork, with an odd number of inversions, will be detected by a stuck at test set.

Having presented these results, it is worthwhile to summarize the types of detectable bridging faults for which detection is not guaranteed, a priori, when using any single stuck fault test set:

1. Shorts which introduce non-inverting feedback, or any feedback shorts in non-fanout free subnetworks.
2. Shorts across inputs of NAND/AND gates which have fanout on one or both of the shorted inputs.
3. Shorts across different levels of logic, or across entirely different paths or subnetworks.
4. Any shorts in asynchronous sequential circuits.
Even though it has been shown possible to rearrange the tests in a single fault test set to detect some faults in the first category by "sequential sensitization" most of the faults mentioned appear to be quite unstructured and may require, in general, extra tests to detect. In fact, on a typical digital circuit, the likely shorts about which we have no a priori knowledge of detection are in the vast majority over any others. However, as with many other faults which can occur in a system, a large number of bridging faults are "accidentally" detected by a single stuck test. The bridging fault simulator was designed and implemented in order to quantitatively determine how effective this "accidental" detection actually is for a given network, and experimental results were obtained relating the detection of these two different types of failures.

4.3 Discussion

In the previous section we have seen that certain bridging faults must be detected by a single stuck test set. This section considers a small (2 gate) network in an effort to gain insight into the applicability of these results and give some examples of faults which are not detected.
Figure 4.6 shows one of the simplest possible two gate networks, namely two parallel NAND gates, and two possible sequences which may be used to detect all stuck faults in the network. There are 6 nodes in this network and thus $6\times5/2 = 15$ possible single bridging faults. These faults are tabulated in table 4.1 along with their detection by either sequence. This network is of interest because in physical circuits employing SSI technology, the leads for different gates are on adjacent pins of the device, and a parallel structure of functions is very common. We see that one of the test sequences detects only 8 of the possible shorts while the other detects 10. In fact, it can be shown that no matter what permutations of those separate test sequences for each gate are applied it is not possible to detect more than 10 of the 13 detectable shorts in this network. This indicates that for a given network there may be no minimal stuck at test sequence which detects all single bridging faults and that, consequently, bridging faults must be separately considered when generating test sequences for logic networks.

4.4 Some Expected Bounds for Bridging Fault Detection

Section 1 of this chapter has considered a precise relation between the detection of certain bridging failures
Figure 4.6. Two NAND Network

<table>
<thead>
<tr>
<th>Test Sequence A</th>
<th>Faults detected (d) or not detected (n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>test</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1. Detection of Bridging Faults
in a logic network, and the detection of stuck-at faults. In this section we seek to gain some understanding of the extent to which bridging failures may be detected in large networks, and for test sets which are not complete, i.e. do not detect 100% of the single stuck failures. This discussion is limited to combinational networks; however, the empirical results of simulation data presented in chapter 6 indicate that similar arguments may be developed for sequential circuits. For the following discussion we consider two leads \( x_1 \) and \( x_2 \) to be logically independent for an input vector \( \overline{v}_t \) (or logically independent at time \( t \)) if, for \( \overline{v} \), there is no sensitized path from \( x_1 \) to \( x_2 \) and vice versa. Clearly \( x_1 \) and \( x_2 \) are always logically independent if there is no path from \( x_1 \) to \( x_2 \) and no path from \( x_2 \) to \( x_1 \). For a large network consisting of complex integrated circuits, and on which shorts may only occur at the circuit board level, it may be assumed that two points shorted together are logically independent for a given input vector. For this case, we present an argument which leads to a lower bound on the expected detection of bridging faults by a test for all output stuck-at faults.

We now consider the conditions which are necessary for an AND bridging fault to be detected. Say \( g_1 \) and \( g_2 \) are two output pins shorted together in a network \( N \), as
shown in figure 4.7. The first condition that must be fulfilled to make detection possible is that the output values of the two pins are not equal. For a large network it is assumed that the probability of any lead in a network being a 1 or 0 at any time is equal. Assuming this, the probability that \( x_1 \neq x_2 \) at any given step in the application of the input sequence is .5. Now suppose that input vector \( \bar{v}_{t_1} \) is a test for the output of \( g_1 \) stuck-at-0. This implies that \( x_1(t_1) = 1 \). If \( g_2 \) is logically independent of \( g_1 \) at \( t_2 \), and \( g_1 \) stuck-at-0 is detected on a primary output logically independent of \( g_2 \) (at \( t_1 \)), then the probability \( P(\text{bridge } x_1) \) of detecting \( *(x_1/x_2) \) (figure 4.7) at test \( t_1 \) is .5. However, the bridging fault \( *(x_1/x_2) \) can be detected not only be a path from \( g_1 \), but also from \( g_2 \). Thus, if test \( t_2 \) detects \( g_2 \) stuck-at-0 then the probability \( P(\text{bridge } x_2) \) of detecting \( *(x_1/x_2) \) at test \( t_2 \) is also .5. This implies that the total probability \( P(\text{bridge}) \) for detecting \( *(x_1/x_2) \) at either \( t_1 \) or \( t_2 \) is

\[
P(\text{bridge}) = P(\text{bridge } x_1) + P(\text{bridge } x_2) - P(\text{bridge } x_1)P(\text{bridge } x_2)
\]

or

\[
P(\text{bridge}) = .75
\]
Figure 4.7. Two Gates Shorted Together
Thus, if each of the output stuck-at-0 faults is detected exactly once by the test sequence, we can expect to detect 75% of the shorts. Now consider the case where each of the stuck-at-0 outputs are detected by a large number of tests. In this case the probability that the short will be detected through a path from either $g_1$ or $g_2$ approaches unity, and the combined probability of detection must likewise approach unity. Thus, depending on the number of times each of the output stuck-at-0 faults is detected, we expect to detect at least 75% and up to $\sim 100\%$ of the shorts for which our assumptions hold. Intuitively, this seems to indicate that the stuck-at fault model is very good for detecting most shorts. Now let us suppose that not all of the output stuck-at-0 faults are detected. The probability of detecting a given stuck-at-0 output fault is then the percentage of faults detected divided by 100. We denote the probability that $x_1$ or $x_2$ stuck-at-0 (in figure 4.7.) is detected by $P(s-a-0)$; and the probability that the bridging fault is detected by $P(\text{bridge})$. Using the same reasoning as above, we see that:

\[
P(\text{bridge}) = .5P(s-a-0) + .5P(s-a-0) - (.5P(s-a-0))^2
\]

\[P(\text{bridge}) = P(s-a-0) - 1/4(P(s-a-0))^2\]
for the case in which each of the stuck faults which are detected, are detected only once. In the more realistic case where we assume that each detected output stuck-at-0 is detected by many input vectors, we approach the limit.

\[ P(\text{bridge}) = 2P(s-a-0) - P(s-a-0)^2 \]

for which \( P(\text{bridge}) \geq P(s-a-0) \)

Thus one would expect the percentage detection of bridging faults to be greater than that for output stuck faults.

4.5 **Network Transformations for Modeling Bridging Faults With the Stuck-at Model**

Most efforts in designing algorithms to test and diagnose digital networks have concentrated on methods to detect stuck-at-0, stuck-at-1 failures. Two networks transformations are presented here which allow these methods to be used in generating test sequences for bridging faults. The only exceptions to this are algorithms which work only for combinational logic networks. When the bridging fault causes the network to become sequential, the feedback loop needs to be identified and the fault must be treated separately.
The first transformation is used for the normal AND bridge and also the NAND switch, while the second concerns the NOR switch. The procedures for using the transformations are as follows:

1. Determine which type of bridging fault is to be modeled. A NAND or NOR switch may be ignored for purposes of detection in combinational networks.

2. Change the network model as shown in figures 4.8 and 4.9. The first transformation, for AND bridges and NAND switches, may be described as follows:
   a) Connect all leads which were connected to the outputs of \( g_1 \) and \( g_2 \) to \( B_1 \) and \( B_2 \), respectively.
   b) Connect \( g_1 \) and \( g_2 \) to \( A_1 \) and \( A_2 \), respectively.
   c) Add primary input \( a \) to the network. This lead controls the presence of the fault.

The NOR switch is made more complex by the fact that, in general, the result of the short may depend on the other inputs to the shorted NOR gate. This transformation (shown in figure 4.9) is described as follows:

   a) connect all leads which were connected to the outputs of \( g_1 \) and \( g_2 \) to \( B_1 \) and \( B_2 \), respectively.
NAND Bridge: \( F = A_1 \cdot A_2 \)
AND Switch: \( F = A_1 \)

Figure 4.8. AND Bridge/NAND Switch Transformation

Figure 4.9. NOR Switch Transformation
b) connect the output of \( q_1 \) to \( A_1 \)
c) connect input leads \( A_2, \ldots, A_n \) of \( q_2 \) to \( A'_2, \ldots, A'_n \) of the transformation network.
d) add primary input \( \alpha \).

In both the above transformations, the network performs its normal function when \( \alpha = 0 \), and performs the bridging function when \( \alpha = 1 \). The normal network corresponds to the new transformed network with primary input \( \alpha \) always set to 0. Thus, the bridging fault in the old network corresponds to the new network with \( \alpha \) stuck-at-1.

These transformations are implemented in the Bridging and stuck-at-fault simulator, which is described in the next chapter.
CHAPTER 5

THE BRIDGING AND STUCK-AT FAULT SIMULATOR

5.1 Introduction

The Bridging and Stuck-at Fault Simulator is a program designed to evaluate the effectiveness of test sequences in detecting and diagnosing stuck-at and bridging faults on large asynchronous sequential networks. The program is an extension of a stuck fault simulator written by the author as part of a mini-computer based logic test system.\textsuperscript{16,17}

In this chapter, we will discuss the design and implementation of the Bridging and Stuck-at Fault Simulator in detail, and also briefly explain some elements of the logic test system which are necessary for the discussion. In addition, we will discuss the implementation of the bridging fault model, and those aspects of the simulator which reflect the modular structure of digital circuit boards. The simulator presented here is of interest because it is, to the author's knowledge, the first program which has quantitatively determined the detection of likely bridging faults on digital circuit boards. No prior published results specifically addressing this problem have been found. The next section outlines some of the design requirements which are necessary for accurate simulation of
complex digital systems. Section 3 discusses in detail the design of the simulator with references to these requirements, and the implementation of the bridging fault model. Finally, in section 4, we discuss the tester interface with the simulator, and the methods which were used to collect the data presented in chapter 6.

5.2 Simulation Requirements

One of the major tasks of this thesis has been to quantitatively determine the detection of bridging failures on digital circuit boards in actual use today. In chapter 3, we studied the behavior of shorting conditions on particular types of integrated circuits and defined a model for their behavior. However, in order for the fault model to be useful in determining which of the corresponding physical failures are detected by a test sequence, it is necessary to simulate the behavior of these faults in a model of the circuit. If the results of simulation are to be correct, the fault simulator must accurately take into account many features of actual circuit operation. As opposed to abstract combinational logic networks in which the function realized is a simple boolean function of input values, a typical digital circuit board is a complex sequential circuit which frequently depends on crucial
timing information in order to carry out its intended functions. The addition of a bridging fault model compounds this issue, since these faults may cause asynchronisms not present in the original network.

Another important consideration in the design of the bridging fault simulator is its feasibility; that is, can the simulation of all faults be done in a reasonable amount of time. The large number of possible bridging and stuck-at failures which are considered rule out the possibility of re-analyzing the network structure for each fault to be modeled. Thus, even though a bridging fault may introduce a new memory element into a digital circuit, we must avoid performing a loop analysis to re-determine all the network state variables for this particular fault. Behavior of a bridging fault should be implicitly accounted for in the structure of the network model, without special handling for individual faults. A third requirement for the bridging fault simulator is that it correctly model the modular structure of digital circuit boards. In particular, an MSI IC module may be represented by perhaps fifty or more gates. The signal path from one input to an output may include as many as 10 gates. However, the manufacturer's specification for that module might call for a propagation
delay equal to that for two normal gates. Therefore, it is necessary to handle the modular timing conditions correctly and efficiently.

Finally, in order to verify the accuracy of our fault model and thereby evaluate its usefulness as a tool for detection and diagnosis, it is necessary to compare the results of simulation with the actual responses of faulted digital circuits. This aspect of the test system is used to obtain the diagnostic data comparing the bridging and stuck-at fault model for their diagnostic capabilities.

The design of the fault simulator may be described by its basic properties which are explained in detail below. In particular, the simulator is a table driven, three-state, parallel, delay model, selective trace, functional level fault simulator.

5.3 Fault Simulator Design

The Bridging and Stuck-at Fault Simulator is an assembly language program written for a PDP 8/E mini-computer with a minimum of 8k of core storage and a high speed random access disk storage device. Implementation of the simulator on this computer allows on-line comparison
of the simulation outputs with the response of actual
digital circuits through the use of a logic test system,
which is based on the same mini-computer. This approach
has the major advantage that the accuracy of the fault
model can be ascertained rapidly and automatically. In
addition, it will be shown that the combination of a logic
tester and fault simulator is a powerful tool for the
automated diagnosis of digital circuits.

5.3.1 Table Driven Simulation

The description of a network which is input to the
fault simulator can be interpreted in two ways. First,
it can be compiled into a set of computer instructions
and subroutines designed to perform the function of the
specified network. This approach is known as compiler
driven simulation. The Bridging and stuck-at fault
simulator uses a different approach. In this approach,
the description of a network is translated into a table,
or data file, which is accessed by a simulator control
routine. This data file contains all the information
which characterizes a particular network, including the
interconnections and types of all its constituent elements.
In particular, this table of data consists of a set of
devices which are core resident throughout the simulation. These devices, in turn, are composed of a chain record, one or more input records and output records, a possible hidden state record, and a device type record. These are explained below.

Output record:

At any given time this record contains an output value of a particular device. In addition, it contains a pointer to the first of a chain of connected input records. Finally, space is reserved for a flag to indicate whether this output is faulted in a stuck-at condition.

Input record:

The value of a device input, a chain pointer to other inputs connected, and a stuck flag are kept in this record.

Device type record:

This record simply contains the device type information used by the control routine in determining how to simulate the device.

Chain record:

This record indicates whether a given device is to be simulated during any given time step. The exact function of the record will be explained in the description of the
selective trace algorithm employed by the simulator.

Hidden state record:

The hidden state record contains internal state information about a particular sequential device. The use of the hidden state record allows efficient simulation of devices at the module level. Since we do not consider shorts internal to any modules in a logic network, it is not necessary to recompute state variables of any devices for any bridging faults.

Figure 5.1 illustrates the way devices are connected together. Note that an external input has only an output and a type record, and an external output has only an input and a type record.

5.3.2 Three State Simulation

Three values of logic (0, 1 and X (unknown)) are employed in the simulator, to determine when the good network has become initialized and thus when outputs should be compared for fault detection. Since most sequential logic is initially in an unknown, random state, it is necessary to apply a resetting or initializing sequence to the logic in order to home it into a set state for testing. A three state simulator determines when
initialization of the circuit is complete: thus, it indicates when a meaningful comparison can be made between the outputs of good and faulty networks. In addition, the three states of logic enable the simulator to identify failures which do not allow the network to become initialized. Detection of these faults by a test sequence, even though highly probable, cannot be guaranteed because the physical network may start in the correct initial state even in the presence of the fault. An example of such a fault is a stuck fault on, or a bridge fault to, the reset pin of a flip-flop. In practice, the occurrences of such a fault is almost always detected, so that, in determining the fault coverage of a particular test sequence these faults are indicated as detected.

5.3.3. Parallel Simulation

To allow the rapid simulation of a large number of faults, a parallel simulation scheme was implemented in which a number of different faulty networks are simulated simultaneously. Since the bit encoding of the network "states" (for each device) uses two bits for each logic e-lead in a network (3 states => 2 bits) and the PDP 8 is a 12 bit computer, it is possible to simulate six faulty
networks in parallel. We will often refer to the different faulty networks that are being simulated as "machines". This is a reflection of the fact that each different faulty network is a different finite state machine. Parallel simulation takes advantage of the width of a computer's word to perform many common logical operations. For example, "AND" ing two values, can be done as efficiently for 12 bits as it can for 2 bits. The greater the word length of the particular computer, the greater the overall savings in simulation time will be. The resultant code which must keep separate track of the detection of each machine by masking out others is more complex, but this additional complexity is fully justified by the savings in total simulation time. The simulation of several bridging faults in parallel presents a special problem since, in general, six different network topologies must be maintained at once. This will be explained in detail in the next section.

5.3.4 The Delay Model

Associated with each device in the simulation table is a unit delay. This design feature allows the simulator to implicitly take into account the operator of feedback
inputs to C are 10 and the pulse is not created! Had we chosen to simulate the devices in order ACBD, the inputs to C would have been 11 (after simulation of A) and the pulse would have been generated. This order dependency of the results cannot be tolerated, and therefore we must adopt an approach which always gives the correct results. This is done by having two values in each input record, one for the present state of the input, and another for the next network time instant. Any device is always simulated using the 'present' values of the input, and the output values computed are propagated to the 'future' values of any inputs. After simulation of a given network instant, the future values become present values and the cycle is repeated.

Using this scheme, asynchronisms in a digital circuit are properly reflected in the network model, and we may expect a perfect match between the behavior of the actual circuit, and the network modeled in the computer.

5.3.5 Selective Trace Algorithm

In the previous section we mentioned that each device is simulated for every given network instant. Substantial savings in simulator execution time are realized if we
Figure 5.1. Simulation Table
loops in a circuit. In order to clarify the discussion of this model, we must distinguish between two different kinds of time. This is because it is impossible to simulate the behavior of all different parts of a network simultaneously, even though in the physical circuit counterpart of the model, many transitions may be occurring at the same time. Thus, we refer to a simulator instant or simulator time as corresponding to the execution of an instruction in the computer, whereas a network instant corresponds to an instance of the real time which is being simulated.

Consider the example of figure 5.2. We see that for an input sequence 01 on primary input A, a pulse is generated at device D. Suppose, now, that we have this network in the simulation table and simulate each device at every network instant. Also assume that we start with the stable network configuration shown for $t_n = 1$. Now let us consider what happens when we change the value of A to 1. Assume that we simulate the devices in order A, B, C, D. First, when we simulate A, we propagate the new value of A to the inputs of B and C. Next, we simulate device B (a NOT gate). Using the information in the table that B is a NOT gate, we compute the result which is 0, and propagate this to the other input of C. But now the
Figure 5.2. Generating a Pulse
only simulate those devices in the network whose input configuration has changed. Thus, instead of having a list of pointers to all the devices, the simulator employs a chaining method whereby only those devices which are active are selected for simulation in any given network time instant. This is done through the use of the aforementioned chain records for each device. Specifically, after the outputs of any particular device are computed, it is first determined whether they have changed from their old values. For any outputs that are changed, the connected inputs are updated and the devices containing these inputs are chained to the "future chain", which signals it for simulation in the next network time instant. Thus, each chain record contains two pointers, one for the next device in the chain to be simulated in this network time instant, and the other has the same function for the next time instant. The base of the chain is located in a known location, so that new devices may be easily added. When any particular network time instant has been completely simulated (signified by a null pointer in the present chain record) then the sense of the two chains is reversed and the future chains becomes the present chain. The network is stable (i.e. simulation is complete for a given input vector \( \vec{v} \)) when no
devices are placed on the future chain. This happens when all devices simulated for a given network time instant do not change the value of their outputs.

5.3.6 **Functional Level Simulation**

The basic elements of simulation in many fault simulators are gates. The advantage of this approach, which is called gate level simulation, is its simplicity. However, there are two problems. First, it is impractical to represent many complex MSI and LSI devices in this manner. For example, a 1024 bit shift register would require about 6,000 gates for its implementation. The storage and computation time requirements to simulate one time step for 6,000 gates are orders of magnitude larger than those required to compute the actual function performed by this device. Second, as mentioned earlier, it is difficult to maintain complex circuit timing relationships in the gate model representation. Thus, we adopt a functional level simulation approach in which the basic devices in the simulation table may represent complex modules. Basically, each device type is simulated by a different subroutine. Because of the sequential nature of most circuit modules, the input variables for these subroutines consist not only
of device input and output values, but also internal state information. This data is kept in the hidden state records for these types of devices. In the example of the 1024 bit shift register, the hidden state record requires only 1024 words for the register's storage, plus a few words of control information. Using the input values and hidden state information, a simple subroutine can easily compute the new output values.

Since this module is represented by a single device, much the same as an AND gate, the propagation delay associated with it is identical to that of the gate. Thus, the delay across any device is always one time unit. For standard TTL logic one time unit may be equated with 10 nano-seconds. If the actual propagation delay associated with a shift register is say, 20 nano-seconds then we implement a module as more than one device. In particular, it consists of the shift register device plus single input AND gates, called buffers, connected to each of the shift register outputs.

5.3.7 Fault Simulation

A stuck-at-0, stuck-at-1 fault model is simply recognized by a set of flags in the appropriate input or output
record of a device. If the flag is set for a particular machine, the value of that module input or output pin will remain fixed for that machine throughout the simulation. The pins to be faulted are selected prior to simulation by a special network analysis program. This program selects the stuck failures to be simulated, collapses them into equivalent fault classes, and provides an appropriate list of faults to the fault simulator. For the implementation of the bridging fault model, this program has been extended to select the proper bridging fault mechanism for each short to be modeled, and provide pointers to the output records of the shorted devices. The fault simulator uses these pointers and the information of which type of bridge to model in order to effect the transformations presented in chapter 4. In particular, the bridging fault model is implemented by a dynamic reconnection of the appropriate e-leads of the network for each fault that is simulated. Since we simulate six bridging faults in parallel, six different bridging faults must be present in the network simultaneously. Therefore, six transformations are inserted into the good network at once. If \( a = 0 \) for all six transformations then the resultant network is functionally equivalent to the fault-free network. To
effect a single bridging failure, in one machine, it is only necessary to set the corresponding \( a \) for that machine stuck-at-1. Thus, it is possible to consider six different bridging failures independently in a single network.

5.4 Tester Interface to the Simulator and Data Collection

As stated earlier, the fault simulator is part of a logic test system whose detailed operation is described in two manuals.\textsuperscript{16,17} Here we explain only those parts of the system directly relevant to the collection of data for detection and diagnosis.

5.4.1 Tester Interface

In order to detect faults on a digital circuit board or Unit Under Test (UUT), the tester applies a test sequence to the UUT, and compares the circuit responses at the primary outputs with the expected responses of a fault free network. The test sequence is typically created by a test engineer, and the expected responses of the good circuit are generated by simulating the fault-free network. The simulator also evaluates the percentage of stuck and/or bridging failures which are detected by that test sequence.
In order to diagnose faults on the UUT, the tester automatically invokes an on-line fault simulation process which operates as follows:

First, a lookup is made into a fault signature table generated prior to testing. This table contains, for each modeled fault, the primary output values at the first failing test (referred to as the initial faulty response). The lookup compares the UUT response at the first failing test with the entries in the table and selects the corresponding modeled faults. These faults are those which match the UUT responses up to and including the first failing test. Second, the faults selected are simulated and the outputs are compared with the failing UUT response for all tests (referred to as the entire faulty response). Faults which match the behavior of the bad UUT exactly are considered diagnosed to the best diagnostic resolution of the test program.

This technique of comparing simulated faulty behavior to physical faulty behavior made it possible to establish and verify the high accuracy of the bridging fault model implementation.
5.4.2 Data Collection

To compare the bridging fault model with the stuck-at fault model for detection and diagnosis, two types of data were collected for the digital circuits discussed in chapter 6.

1. The detection data was obtained from computer printouts which listed the percentage of detection for two different fault models, namely the stuck outputs and shorts between adjacent pins of IC's.

2. The diagnostic data was generated by inserting shorts between adjacent IC pins into the good UUT. The fault model used for diagnosis was the output stuck-at fault model. The information collected were the faults, if any, that were found in the fault table, and the faults, if any, that matched the faulted UUT's entire faulty response. Similar data was also collected using the bridging fault model, to verify its accuracy in the diagnosis of such faults.
CHAPTER 6
EXPERIMENTAL RESULTS AND CONCLUSIONS

Introduction

The objective of this thesis was to determine the effectiveness of the stuck-at fault model in detecting and diagnosing bridging faults. In chapter 4, we discussed the results obtained by analytical investigations. However, as pointed out earlier these results apply only to specific bridging faults in restricted classes of combinational networks, and therefore have only very limited usefulness. In order to obtain significant results for more typical types of circuits, we chose to compare the bridging and stuck-at fault models via fault simulation.

In this chapter, two types of results are presented. First, we quantitatively establish the effectiveness of the stuck-at fault model in detecting bridging failures for typical digital circuit boards. Second, we quantitatively determine the usefulness of the stuck-at model in locating bridging faults on such circuits.

6.1 Detection of Bridging Faults

To obtain the experimental results presented in this
section, 22 different digital circuits in actual use today were selected at random for simulation. In particular, the test sequences normally used for production testing of these circuits were used to perform the fault simulation of these circuits with two different fault models, namely, the output stuck-at model and the bridging fault model. The output stuck-at model was chosen because this type of fault affects an entire signal line (an output and all its connected inputs) in the same way that a bridging fault does. Intuitively, a bridging fault may be viewed as an output pin stuck for a part of the applied test sequence in which the two shorted output values disagree. Thus, a short is most closely related in its behavior with an output pin stuck.

The bridging faults modeled for these circuits, namely, the adjacent IC pins shorted together, were chosen for several reasons. First, these types of bridging faults are among the most common defects introduced in the manufacturing process, and therefore represent an important set of faults to consider. Second, consideration of adjacent IC module pins shorted allows automatic insertion of these faults by the fault simulator. Finally, the bridging faults modeled in this manner, represent a fairly random sample of all
the likely single bridging faults which may occur on a circuit. The results for detection of these faults are thus a reasonable indication of how well all possible bridging faults may be detected by a particular test sequence.

Table 6.1 shows the detection results obtained for 22 digital circuit boards considered in this experiment. The columns of this table are organized into three groups:

1. Circuit Description
2. Test sequence length
3. Fault detection

Under the circuit description heading are three simple measures of network complexity. The first column is the total number of gates which would be required for a gate-equivalent description of the circuit. The second column indicates the count of the number of integrated circuit modules contained on each board. The third column shows the depth of memory or sequential levels of the circuit. For example, a 4 bit shift register is considered to have 4 memory levels, as in a 4 bit counter. These first three columns provide a rough measure of a circuit board's
overall complexity, and also give an indication of the degree to which medium and large scale integration are employed on each circuit.

The second section contains one column marked $|\bar{S}|$ which indicates the number of input vectors applied in each test sequence.

Finally, the third section contains the detection data gathered through simulation of the output-stuck faults and the adjacent pin bridging faults. The columns marked "#0" and "#S" indicate the number of faults of each type that were simulated, and the columns marked "%0" and "%S" show the percentage detection of each of these types of failures.

**Evaluation of Data and Conclusions**

The observation that is immediately made is the relatively high percentage of bridging fault detection when compared to the detection of output stuck-at faults. In only 2 of the 22 cases studied here was the percent detection significantly lower for bridging faults than it was for output stuck-at faults. In 17 of the boards, detection of bridging faults was greater than or equal to that for output stucks, while in the remaining 3, the results for the bridging fault model were less than 4
<table>
<thead>
<tr>
<th>circuit complexity</th>
<th>test</th>
<th>detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>gates</td>
<td>ICs</td>
<td>levels</td>
</tr>
<tr>
<td>A 7148</td>
<td>53</td>
<td>100</td>
</tr>
<tr>
<td>B 6900</td>
<td>38</td>
<td>15</td>
</tr>
<tr>
<td>C 3278</td>
<td>93</td>
<td>4</td>
</tr>
<tr>
<td>D 2191</td>
<td>57</td>
<td>8</td>
</tr>
<tr>
<td>E 1950</td>
<td>50</td>
<td>52</td>
</tr>
<tr>
<td>F 1937</td>
<td>54</td>
<td>30</td>
</tr>
<tr>
<td>G 1816</td>
<td>43</td>
<td>8</td>
</tr>
<tr>
<td>H 1100</td>
<td>116</td>
<td>4</td>
</tr>
<tr>
<td>I 900</td>
<td>55</td>
<td>14</td>
</tr>
<tr>
<td>J 854</td>
<td>25</td>
<td>1</td>
</tr>
<tr>
<td>K 814</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>L 645</td>
<td>26</td>
<td>8</td>
</tr>
<tr>
<td>M 580</td>
<td>56</td>
<td>20</td>
</tr>
<tr>
<td>N 446</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>O 418</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>P 390</td>
<td>26</td>
<td>1</td>
</tr>
<tr>
<td>Q 336</td>
<td>24</td>
<td>4</td>
</tr>
<tr>
<td>R 202</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>S 142</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>T 104</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>U 86</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>V 62</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.1 Detection Data
percentage points below those for the output stuck model. This information is summarized in the graph of figure 6.2. These experimental results indicate that the stuck-at-0, stuck-at-1 fault model is highly effective in developing test sequences for bridging faults, as well as those faults which are directly modeled as stuck faults. For most purposes, a test sequence designed solely to detect stuck-at failures may be considered sufficient to detect all likely failures on a circuit board. However, there are a few cases in which detection of bridging faults is significantly lower than that for sticks. Therefore, for those applications in which the higher level of confidence for testing is essential, simulation of the network using the bridging fault model is necessary. In addition to supplying exact information concerning the detection of these faults, the simulator can provide valuable feedback to a test engineer (or computer program, for that matter) in designing tests for bridging faults.

6.2 Verification of the Bridging Fault Model Implementation

In order to assure the accuracy of the quantitative experimental results presented in this thesis, it was necessary to establish the correctness of the bridging
Figure 6.2. Bridging vs. Output Stuck Detection
fault model and simulator implementation. This was done by comparing the output of faulty modeled networks with the corresponding outputs of faulted physical circuits. In particular, shorts were inserted on 4 different circuit boards (UUT's), and the electrical outputs at the first failing test were compared with the primary outputs of faulty modeled networks detected by the same test. The data shown in table 6.3 indicates the results of inserting 180 different adjacent pin shorts on 4 circuits. The letters on the left-most column refer back to the circuits of table 6.1. In the column marked "# inserted" are the numbers of bridging faults which were inserted into each circuit board. The last column contains the number of these faults which were correctly found in the fault table lookup.

Overall, the percentage of modeled faults which correctly matched the UUT responses at the first failing test was 93%. Mismatches between the physical circuit response and that of the fault model occurred for several cases outlined below:
<table>
<thead>
<tr>
<th>Circuit</th>
<th># faults inserted</th>
<th># which did not match model</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>L</td>
<td>35</td>
<td>5</td>
</tr>
<tr>
<td>H</td>
<td>36</td>
<td>3</td>
</tr>
<tr>
<td>M</td>
<td>36</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>147</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 6.3 Verification Data
1. In two cases, a short between the output and an input of an XOR gate caused the circuit to oscillate. This condition effectively results in non-deterministic behavior since, at the time the primary outputs are sensed, any output that is not stable will be arbitrarily read as a 1 or 0.

2. Several shorts were inserted which prevented the circuit from becoming initialized. Here again, the result is non-deterministic, since the circuit may be in a different initial state each time it is tested.

3. Some bridging faults from an input to the output of a NAND gate caused false switching when connected directly to flip-flops. This also would result in non-repeatable behavior.

In summary, then, the bridging fault model is quite accurate in predicting the behavior of shorts on digital circuit boards. The second problem that was noted above is not a problem which is unique to the bridging fault model. In fact, many stuck-at faults may similarly prevent a circuit from becoming initialized. In the author's opinion, the other two problems noted cannot be solved without
adopting a far more complex dynamic (perhaps probabilistic) model of the bridging fault and network behavior. This does not seem practical for any but the simplest networks.

6.3 Diagnosis of Bridging Faults

In this section we seek to gain some insight into the effectiveness of the stuck-at model when used to diagnose bridging failures. In other words, we want to determine whether the bridging fault model is necessary to diagnose digital circuit boards. The intuitive argument made for using the stuck-at-0, stuck-at-1 model for circuit diagnosis is as follows.

First, there are many common faults in newly manufactured circuits which are accurately represented by the stuck-at-0, stuck-at-1 fault model (see chapter 2). Second, as stated earlier, a bridging fault often causes a lead in the network to "appear" to be stuck-at-0 during part of the test sequence application. The primary objective of the experiments in this section was to test the validity of this second argument. Since it is clearly impossible to explicitly diagnose a bridging fault using the stuck-at model, we consider a short to be located by the stuck-at
model if, for any given test sequence, the response of the bridging fault is the same as for either of the shorted output leads stuck-at-0.

Table 6.4 shows the results obtained when the initial response of a physical short was used to find a match with the initial response of one of the shorted output leads stuck-at-0. We see that in almost a fifth of the 256 shorts inserted, the initial response of the bridging fault is different from that of either of the shorted leads stuck. Thus, an automatic diagnostic system based on a stuck-at model may be expected to often give a wrong location for bridging faults. Using only the initial response has the following disadvantages:

1. An incorrect diagnosis may be expected in 1 out of 5 cases.
2. Poor diagnostic resolution is expected since only part of the test sequence is used for the diagnosis, namely, the tests up to and including the first failing test.

This leads to vague and often incorrect diagnostic results which are often useless. In order to obtain the best
<table>
<thead>
<tr>
<th>Circuit</th>
<th># faults inserted</th>
<th># matches in initial responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>36</td>
<td>32</td>
</tr>
<tr>
<td>H</td>
<td>35</td>
<td>24</td>
</tr>
<tr>
<td>F</td>
<td>60</td>
<td>40</td>
</tr>
<tr>
<td>M</td>
<td>44</td>
<td>37</td>
</tr>
<tr>
<td>L</td>
<td>47</td>
<td>31</td>
</tr>
<tr>
<td>I</td>
<td>34</td>
<td>31</td>
</tr>
</tbody>
</table>

| Totals  | 256              | 210                          |

Table 6.4 Diagnostic Data - Initial Response
possible diagnostic resolution from a given test sequence, it is necessary to match the entire response of a faulty physical circuit with some modeled faulty network.

Table 6.5 shows the results of comparing the entire responses of physical bridging faults with the entire responses of modeled output stuck faults on the shorted leads. In only 27 of the 256 faults inserted did the entire response of the bridging fault match that of an output stuck model. This indicates that the stuck-at-0, stuck-at-1 fault model is completely inadequate for the accurate diagnosis of bridging failures.

The use of the bridging fault model is essential if we are to achieve correct diagnosis and maximum diagnostic resolution. There are two reasons for this:

1. By using the bridging fault model it is possible to explicitly diagnose a short which is in the set of bridging faults considered.

2. The best diagnostic resolution of the test program may be achieved by matching the physical response of a faulty board with the simulated responses of possible failures throughout the entire test sequence.
<table>
<thead>
<tr>
<th>Circuit</th>
<th># faults inserted</th>
<th># matches in entire response</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>36</td>
<td>1</td>
</tr>
<tr>
<td>H</td>
<td>35</td>
<td>7</td>
</tr>
<tr>
<td>F</td>
<td>60</td>
<td>8</td>
</tr>
<tr>
<td>M</td>
<td>44</td>
<td>4</td>
</tr>
<tr>
<td>L</td>
<td>47</td>
<td>2</td>
</tr>
<tr>
<td>I</td>
<td>34</td>
<td>5</td>
</tr>
</tbody>
</table>

256                      27

Table 6.5 Diagnostic Data - Entire Response
6. Summary of Results and Conclusions

Several important results concerning bridging faults have been obtained from the studies made in this thesis. These may be divided into four parts which are briefly summarized below.

First, in examining the behavior of bridging faults on TTL gates, we found that this type of fault almost always performs a simple logical function of the shorted output leads. This led to a simple logical model of bridging fault behavior which could easily be implemented on a computer.

Second, a careful study of the effects of bridging faults in combinational networks revealed that certain bridging failures may be disregarded for purposes of generating fault detection tests, since those failures are always detected by a test for stuck-at-0, stuck-at-1 faults. In addition, we presented arguments indicating what percent of bridging faults we would expect to detect with a test sequence designed to detect only stuck-at failures.

Third, we showed that the behavior of bridging faults could be correctly accounted for in a computer program to simulate digital circuits. In particular, the use of a table driven design allowed the topological changes caused
by bridging faults to be easily performed, and the delay model of devices allowed the addition of feedback loops by these faults to be implicitly taken into account.

Finally, the fourth part of this thesis was concerned with obtaining quantitative results relating the detection and diagnosis of bridging failures using the bridging versus the stuck-at fault model. From the results of these experiments we conclude that the stuck-at fault model is an effective tool in generating test sequences which detect bridging failures. On the other hand, we have found that the stuck-at fault model is entirely inadequate for the diagnosis of bridging failures. As a result we conclude that fault dictionaries used in the diagnosis of digital circuit boards should be generated using both the bridging and stuck-at fault models.
REFERENCES


