SENSITIVITY OF SCIRTSS TO WEIGHTED HEURISTIC FUNCTIONS

by

Thomas Morriss Callaway

A Thesis Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

1 9 7 5

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED Thomas M. Callaway

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Fredrick J. Hill — 6 June 1975

Fredrick J. Hill
Professor of Electrical Engineering

Date

ACKNOWLEDGMENTS

I would like to offer my greatest thanks to my thesis and academic advisor, Dr. F. J. Hill, for his instruction, interest and guidance throughout my course of study at The University of Arizona. I would like to express my sincere gratitude to Mr. Wai Wing Ng and Mr. Ben Huey for their assistance at the outset of this project.

I am thankful to the United States Army and The University of Arizona for the opportunity to pursue my studies.

Finally, my wife, Cathie, daughter Heather, son Brian and dog Brandy, can never be repaid for the burden they bore throughout my graduate study.

TABLE OF CONTENTS

		Page
	LIST OF ILLUSTRATIONS	v
	LIST OF TABLES	vi
	ABSTRACT	vii
1.	INTRODUCTION	1
	Analysis of the SCIRTSS Program	1
	Statement of the Problem	2
2.	THE DATA SHUFFLE CIRCUIT	8
	Test Sequence Determination	11
3.	THE NARROW WINDOW CIRCUIT	20
	Test Sequence Determination	25
4.	THE MEMORY-EMBEDDED CIRCUIT	36
	Test Sequence Determination	42
	Search Comparison	53
5.	CONCLUSIONS	58
	Search Sensitivity to Weight Changes	62
	Circuit Dependent Tendencies	63
	Circuit Classifications	64
	Optimization Process	65
	LIST OF REFERENCES	68

LIST OF ILLUSTRATIONS

Figure		Page
1.1	Breadth-first Search	6
2.1	AHPL Description of the Data Shuffle Circuit	9
2.2	Control State Diagram for the Data Shuffle Circuit	10
2.3	Control and Optimal Runs for the Data Shuffle Circuit	13
2.4	Control and Optimal Results for the Data Shuffle Circuit	14
3.1	AHPL Description of NWC	21
3.2	Control State Diagram for the Narrow Window Circuit	23
3.3	Optimal and Control Run Results for the Narrow Window Circuit	29
3.4	Search Results for Optimal and Control Runs on the Narrow Window Circuit	30
4.1	Logic Block Diagram of the Memory-Embedded Circuit	37
4.2	AHPL Description of the Memory-Embedded Circuit	41
4.3	Results of Control and Weighted Runs for the Memory-Embedded Circuit	46
4.4	Results of Control and Weighted Runs on the Memory-Embedded Circuit	47
4.5	Example Printout For Breadth-first Search	55
4.6	Example Printout For Weighted Search	56
4.7	Example Graph Trees	57

LIST OF TABLES

Table		Page
2.1	Interim Results for the Data Shuffle Circuit	15
2.2	Optimized Weights for the Data Shuffle Circuit	18
3.1	Initial Runs for the Narrow Window Circuit	26
3.2	Initial Runs on the Narrow Window Circuit	31
3.3	Results of Changed Priority Class Run	34
4.1	Initial Searches of the Memory-Embedded Circuit	43
4.2	Interim Runs on the Memory-Embedded Circuit	49

ABSTRACT

In recent months considerable work was done toward the successful creation and testing of the SCIRTSS computer program. This program applies heuristic graph searching techniques to the problem of fault detection in large sequential circuits.

Initially the evaluation of SCIRTSS was done by comparing the results of test sequences generated by SCIRTSS with a random test sequence. This was done by applying both sets of inputs to a clock mode simulator included in the SCIRTSS program. Comparisons remained to be made with the results of another method of searching for efficient test sequences. That comparison is the purpose of this paper. In addition, classification of circuits according to SCIRTSS performance was attempted.

Three previously tested circuits are subjected to both SCIRTSS heuristically weighted searches and to breadth-first searches. The results of these runs are analyzed and compared. It is then shown that SCIRTSS heuristically weighted searches are superior in both efficiency and economy to breadth-first searches.

CHAPTER 1

INTRODUCTION

With the advent of large-scale integrated circuits (LSI) came a need for the ability to test such circuits. By the very nature of their design and implementation, LSI circuits have a small ratio of input and output pins relative to internal elements which precludes element by element or partitioned testing or fault detection. With the problem of the general combinational logic circuit solved by Roth (1966), the problem of developing effective test sequences for clock mode LSI sequential circuits was undertaken by Belt (1973) and Carter (1973). Their result was the Sequential Circuit Test Search System, or SCIRTSS, which Carter (1973) computerized and refined. SCIRTSS is, at the time of this writing, in its second version.

All work to establish SCIRTSS has had in common the use of the heuristic graph search as given by Nilsson (1971). This technique of searching the control state graph of LSI circuits is applied in both search modes of the SCIRTSS program. The theoretical basis of its application is very clearly explained by Belt (1973).

Analysis of the SCIRTSS Program

In order to meaningfully discuss the analysis of the SCIRTSS computer program, the basic assumptions and limitations of the system must be understood. Inherent in the system are the following:

1. Any circuit considered must operate in the clock mode.
2. The system will provide fault detection for only stuck-at-one (SA1) and stuck-at-zero (SA0) faults. Intermittent faults are not considered.
3. While the simulation portion of the program uses parallel fault simulation, the search portion of the program searches for goal nodes related to only one fault at a time.
4. Each circuit considered must possess an initial state.
5. All circuits considered must be describable in AHPL (a Hardware Programming Language) by Hill and Peterson (1973).

Initially, SCIRTSS was to be considered successful, if it performed well in comparison to random input sequences. This was shown to be true by Carter (1973) for circuits with less than 150 elements. Subsequently, significantly larger circuits were tested and compared against random input sequences by Ng (1974) and Van Helsland (1974). These comparisons were made based on the extensive employment of user input information. This analysis included user written subroutines which could be attached to the SCIRTSS computer program and extensive user-generated heuristic vector tables which offered manual guidance to the searches. Both approaches proved quite effective in the cases of individual circuits.

Statement of the Problem

It is the purpose of this paper to continue analysis of the effectiveness of the SCIRTSS system. It is desired to determine how

the SCIRTSS computer program, guided only by its automatically generated heuristic values, compares against a breadth-first search (direct maximal expansion of the control state graph tree) of the same control state graph tree. This concept will be further explained later in this section.

An effort is made to determine whether circuits can be classified so that general input search parameters correspond to specific circuit classifications. This comparison and analysis is to be made without providing the SCIRTSS computer program with manual assistance in the form of heuristic subroutines or extensive user created heuristic vector tables.

In this way the analysis will center on the theoretical application of the heuristic search procedure to control state graphs and may add additional automation to the present version of the SCIRTSS computer program. It will compare the advantages and disadvantages of two types of searches but will not reconsider the advantages of a search over a random input sequence.

The basis for the optimally determined search in all cases is the following two equations as given by Hill and Huey (1975):

for sensitization search:

$$H_n = G_n + \omega H_d\left(1-W_1\left(1-\frac{H_n^P}{H}\right) - W_2F_b - W_3F_f - W_6F_{cd}\right)$$

for propagation search:

$$H_n = G_n + \omega H_d\left(1-W_4\left(1-\frac{H_n^P}{H_d}\right) - W_5F_b - W_6F_{cd}\right)$$

These equations are the ones used to calculate the heuristic weight assigned to each node as the search mechanism moves through the graph tree in each respective search mode. In them, H_n is the heuristic value assigned to the node in question. G_n is the distance from the starting node. ω is a constant which specifies the relative amount of direction to be used by the search. H_d is the default heuristic value for the node. W_1 is the predecessor node adjustment. H_n^p is the heuristic value for the predecessor node. W_2 is the Boolean distance adjustment for the propagation mode. F_b is a function value which varies inversely with Boolean distance from the goal node. W_3 is the fault proliferation adjustment and F_f is a function value proportional to the number of flip-flops in which the value in the faulty network differs from that in the good network. W_6 is the control state disparity adjustment and F_{cd} is the function value which is inversely proportional to the probability of reaching a particular control state. W_4 is the predecessor node adjustment for the sensitization mode, and W_5 is the Boolean distance adjustment for that mode.

The analysis of the above stated equations against the breadth-first search is the heart of this paper and will proceed in the following manner. First, the circuit will be run on the SCIRTSS computer program with weights (W_i) from the two equations all set to zero. The effect of this is that the equations will be reduced to effectively:

$$H_n = G_n + \omega H_d$$

where $\omega = 0$. What this means is that every new node expanded at a given level in the search will have the same weight attached to it. This is the value of G_n at the expansion level of the node. In figure 1.1 a graphic demonstration of this concept is shown. From the figure it can be observed that all nodes at a given level have the same heuristic value. At the first level all nodes have a value of one. At the second level all have a value of two, etc. So, the search will branch out from every node it reaches along every possible path in search of the goal node it is seeking. This is what is meant by a direct maximal expansion of the graph tree. This type of run will be made several times for each circuit with the random clock uncontrolled, and the most promising of these breadth-first runs will be picked as a basis of comparison against which the performance of the heuristically weighted searches will be analyzed. This will be the control run. At this point the random clock setting will be noted for the control run and weighted searches will begin from a common point and will be run with the same random clock setting. The common point will be referred to as $\alpha.\beta$ percent. This means the point in the control run at which $\alpha.\beta$ percent of the active detectable faults have been found. It must be remembered that slightly more efficient searches might be gained by leaving the random clock uncontrolled and letting all searches start from a zero percent point, but that would diminish the effectiveness of a control search and detract from comparisons of one type of weighting used against another.

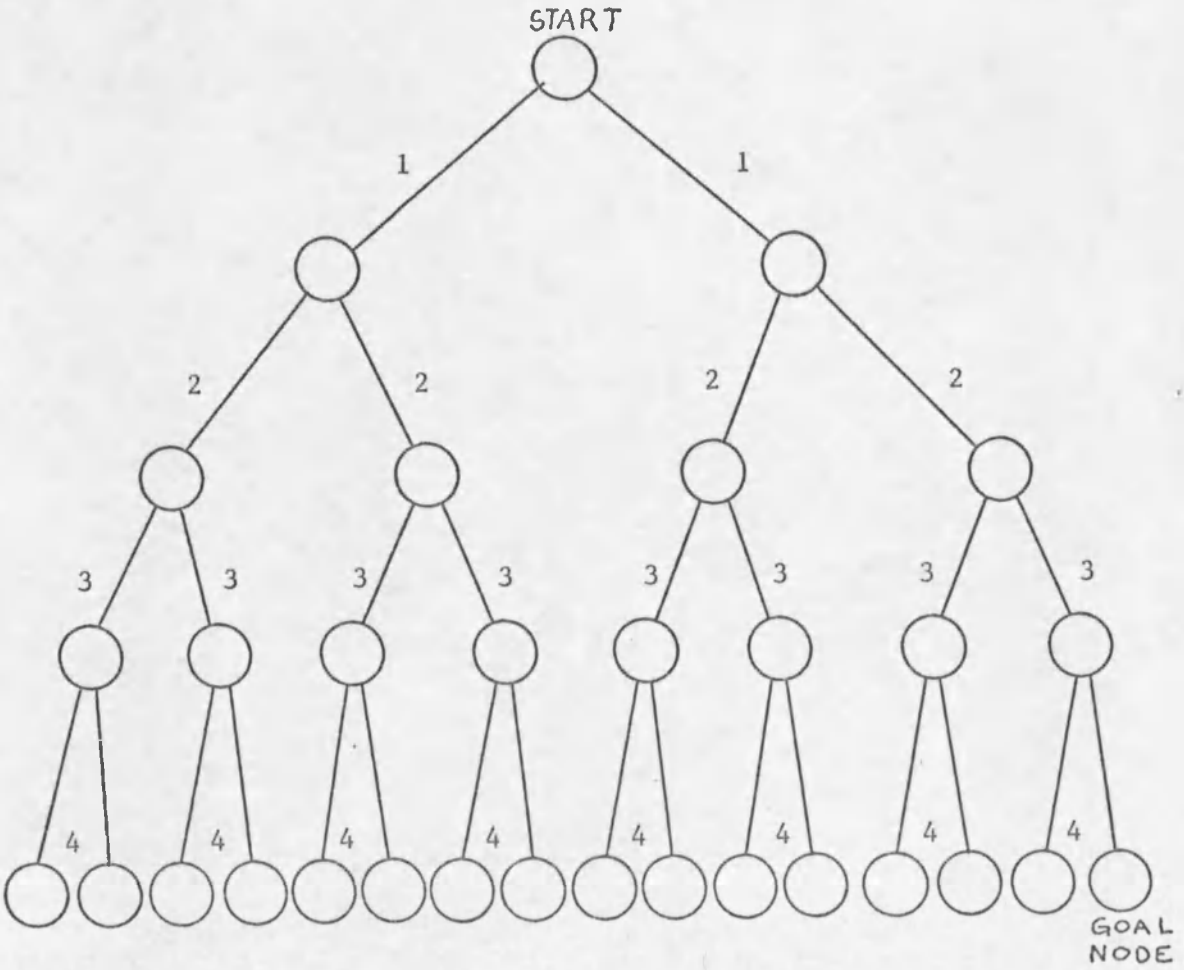


Figure 1.1 Breadth-first Search

An example of one actual search will be given in Chapter 4. It will also contain a small example of how the given equations direct the search.

CHAPTER 2

THE DATA SHUFFLE CIRCUIT

The first circuit presented will be the Data Shuffle Circuit. This has been a very closely studied circuit throughout the entire development of the SCIRTSS program. It was first presented by Belt (1973) as an example in his work. It was further tested by Carter (1973) in his work wherein he made extensive comparisons of version I SCIRTSS runs against random input sequences. As expected, results confirmed the increased efficiency of SCIRTSS over random input sequences. One feature of the circuit is that it has only one output and two input lines. Also, inputs can be applied to the circuit only in control state two as shown in the AHPL description of the circuit in figure 2.1. The circuit is further characterized by the relative simplicity of its internal data transfer operations. In addition, it has an inherently simple control sequence associated with its transfer operations. The control sequence is not based on any elaborate counting or anti-random biasing mechanisms.

There are several interesting features of the Data Shuffle Circuit which may help in further use for classification. The first of these is its control state diagram. As is shown in figure 2.2 the control state diagram is a diverging then converging tree. This, I believe is why the results given by Carter (1973) show the SCIRTSS results to be better than random results, but not exceptionally better.

REGISTERS: $\underline{A}(4)$, $\underline{B}(4)$

INPUTS: $\underline{X}(2)$

OUTPUT: B_2

1. $\alpha^2/\underline{A} \leftarrow \omega^2/\underline{A}$; $\omega^2/\underline{A} \leftarrow \alpha^2/\underline{A}$
   $\alpha^2/\underline{B} \leftarrow \omega^2/\underline{B}$; $\omega^2/\underline{B} \leftarrow \omega^2/\underline{B}$
   $\rightarrow (2 \times X_1 + 5 \times \overline{X}_1)$
2. $A_0 \leftarrow X_1 + X_2$
   $\rightarrow (3 \times B_1 + B_2 + 4 \times \overline{B_1 + B_2})$
3. $\underline{A} \leftarrow \downarrow\underline{A}$; $\rightarrow (8)$
4. $\underline{A} \leftarrow \uparrow\underline{A}$
   $\rightarrow (9 \times X_1 + 10 \times \overline{X}_1)$
5. $B_0 \leftarrow A_3$
   $\rightarrow (6 \times A_0 + 7 \times \overline{A}_0)$
6. $\underline{B} \leftarrow \downarrow\underline{B}$
   $\rightarrow (9 \times X_1 + 10 \times \overline{X}_1)$
7. $\underline{B} \leftarrow \uparrow\underline{B}$; $\rightarrow (8)$
8. $\underline{A} \leftarrow \overline{\underline{A}}$; $\rightarrow (1)$
9. $A_2 \leftarrow A_1$; $\rightarrow (1)$
10. $B_3 \leftarrow B_1$; $\rightarrow (1)$

Figure 2.1 AHPL Description of the Data Shuffle Circuit

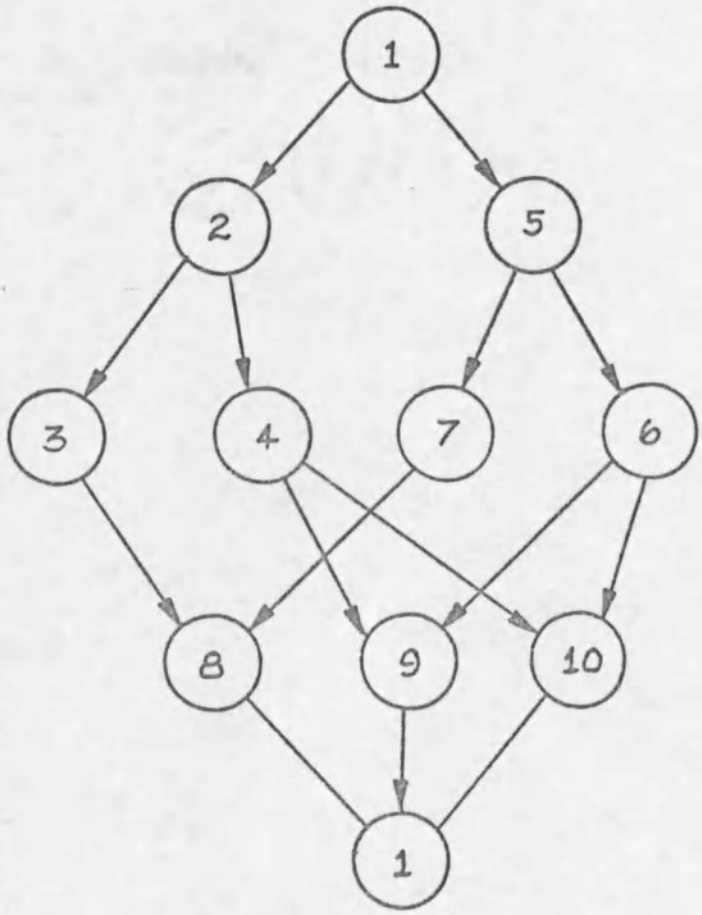


Figure 2.2 Control State Diagram for the Data Shuffle Circuit

Secondly, from figure 2.1 it is apparent that inputs are available only in state two. From figure 2.2 it can be seen that the circuit has at least a 0.5 probability of traversing that state as it goes through every cycle of its state diagram. In this way the narrow input channel is less restrictive. Outputs offer even less of a problem since changes can occur in 4 of the 10 states and are available for monitoring in every state. With this in mind let us turn to the problem of determining the best test sequence under the given control circumstances.

Test Sequence Determination

As in all of the circuits tested, a breadth-first run was established as a control run. Four randomly clocked breadth-first runs were made with all weights in the heuristic equations set to zero. Of these runs the most promising in terms of percent faults detected per test sequence iteration was chosen to be used as the control. At 114 iterations of test sequence this run had found 90.5 percent of the detectable faults. This would be the starting point for all subsequent runs. Next, the control run was reconstituted from this point with a random clock setting of (09.39.15) and allowed to search until it had detected 99.7 percent of all detectable faults. One fault remained undetected, but it was determined that the search would have to be allowed to continue beyond the alloted time limit in order to detect that fault. In addition, the NSIM subroutine of the SCIRTSS

program allows for uniform graph expansion speed. The significance of this fact is that runs held to a given time limit have considerable uniformity in the number of expanded nodes. This makes comparisons of percent faults detected and number of test sequence iterations considerably more meaningful.

At this point with the same random clock setting from the same 90.5 percent point the heuristically weighted searches were begun. A time limit of 40 seconds was set on every run in order to hold to comparable values the number of nodes expanded in each search. This was further facilitated by the fact that this particular circuit did not require extensive use of the D-algorithm. Applications of the D-algorithm consumed significant quanities of computer time in several other circuits and made it difficult to control the number of nodes expanded. The results of the control run, which succeeded in detecting 99.7 percent of the active detectable faults, are shown along with the optimal results in figures 2.3 and 2.4. In addition, table 2.1 contains the results of the weighted runs which were less efficient than the optional run. It can be seen that, although the best run came from a weighted search, some weighting actually detracted from the effectiveness of the searches and caused a loss of efficiency in the run.

Below is a discription of the analysis undertaken to determine exactly the difference in the runs and why one weight or set of weights performed better than another. This in effect was the optimization process used.

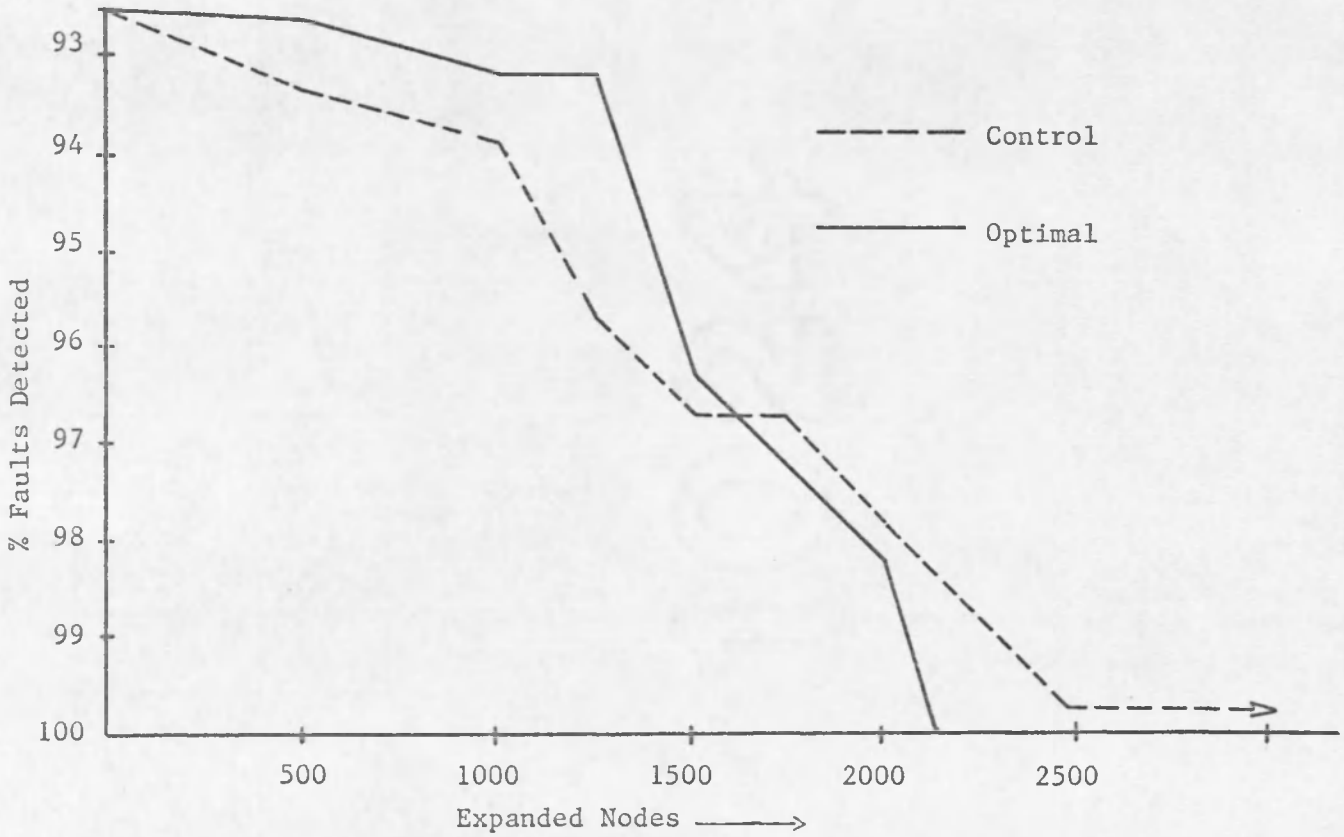


Figure 2.3 Control and Optimal Runs for the Data Shuffle Circuit

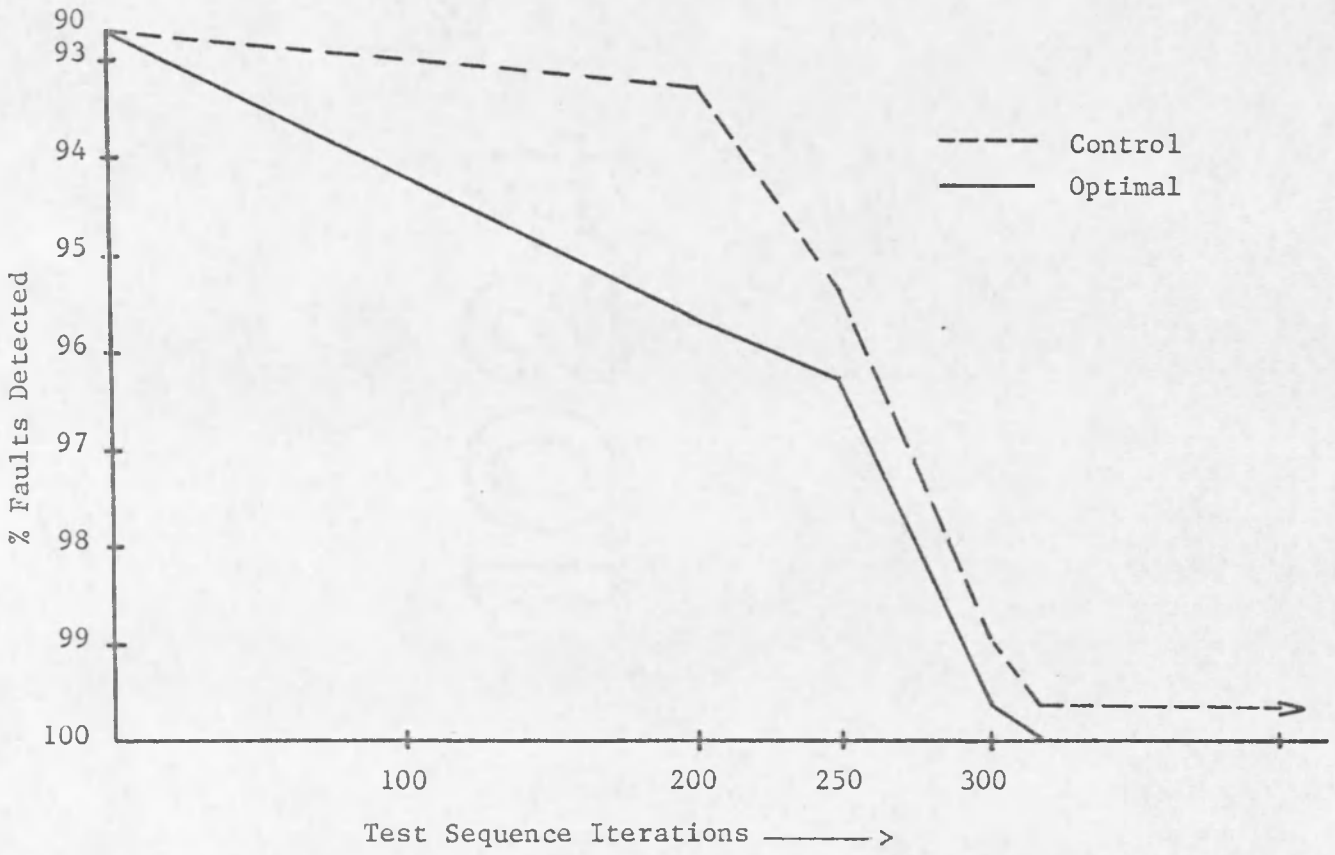


Figure 2.4 Control and Optimal Results for the Data Shuffle Circuit

TABLE 2.1

INTERIM RESULTS FOR THE DATA SHUFFLE CIRCUIT

Run No.	iterations at: 90%	93%	97%	99%	time limit
1.	114	157	300	371	412 (99.3)
2.	114	155	247	315	359 (99.7)
3.	114	174	255	331	371 (99.7)
4.	114	163	271	351	415 (99.7)
5.	114	155	251	331	391 (99.7)
6.	114	155	233	315	331 (100)
7.	114	155	245	317	343 (99.7)
8.	114	157	275	348	367 (99.7)
*9.	114	155	227	295	317 (100)
**10.	114	171	275	320	331 (99.7)

	W_1	W_2	W_3	W_4	W_5	W_6
1.	.5	.5	.5	.5	.5	.5
2.	.1	.1	.9	.8	.8	.5
3.	.2	.2	.2	.2	.2	2
4.	.0	.0	.9	.9	.9	0.0

TABLE 2.1 (continued)

Run No.	W_1	W_2	W_3	W_4	W_5	W_6
5.	.2	.2	.0	.0	.0	0.9
6.	.2	.2	.2	.9	.9	.5
7.	.5	.5	.9	.0	.0	.5
8.	.2	.3	.5	.5	.5	.2
*9.	.2	.2	.9	.8	.8	.5
**10.	.0	.0	.0	.0	.0	.0

* optimal run
** control run

1. It was initially speculated that a high setting of the fault proliferation weighting in the fault propagation mode would be of great value in order to facilitate shorter propagation mode searches. Initial runs holding other weights to zero and starting the fault proliferation adjustment at 0.9 bore this out.
2. After initial runs it became apparent that some difficulty was being encountered in the later fault sensitization mode searches. This problem was corrected by setting the Boolean distance adjustment and the fault predecessor adjustment in the sensitization mode to first 0.9 and later to 0.8. This helped particularly in the detection of the last 5 faults.
3. It was also discovered that a 0.5 setting for the control state disparity weighting was best used in combination with those previously discussed. This observation is undoubtedly connected to the configuration of the control state graph. In only 5 out of the 10 control states does the circuit have the opportunity to vary from a given path, and in these instances, that variance can be only a maximum of 50 percent. This would insure that the probabilities of not traversing any given state in the circuit for any large number of cycles would be extremely small.

With the optimized weights as shown in table 2.2, the SCIRTSS program was found to perform more efficiently from the given starting

TABLE 2.2

OPTIMIZED WEIGHTS FOR THE DATA SHUFFLE CIRCUIT

Heuristic Weights	W_i
Predecessor Node Adjustment (FPM):	0.2
Boolean Distance Adjustment (FPM):	0.2
Fault Proliferation Adjustment (FPM):	0.9
Predecessor Node Adjustment (SM):	0.8
Boolean Distance Adjustment (SM):	0.8
Control State Disparity Index Weighting:	0.5

point (114 iterations). At this point, it seems pertinent to modify this statement with two additional observations. First, in both the optimal search and the control search a similar number of nodes were expanded. This was primarily due to the operation of SCIRTSS at clock mode and a given amount of computational time (40 seconds) allotted for each search. The significant fact was that for a similar sized expansion the optimal run was clearly more efficient. The second point is that the circuit was placed in a rigidly controlled state which may not have allowed as efficient total searching as otherwise possible. In this way the behavior of the search was more closely monitored. Results of the control and optimal searches are shown and compared as percent of faults detected versus first nodes expanded in figure 2.3 and secondly versus test sequence iterations in figure 2.4.

CHAPTER 3

THE NARROW WINDOW CIRCUIT

The next circuit undertaken for study will be the Narrow Window Circuit. Before any consideration of its nature or characteristics can be meaningful, an understanding of the AHPL description of the circuit as shown in figure 3.1 is essential. The control state diagram is shown in figure 3.2. The Narrow Window Circuit was originally studied by Ng (1974) to test the concept of a narrow window or passage between two groupings of states in the control state diagram. The only way to get to the lower group of control states from the upper group is to branch from control states four or seven to state eight. In order to accomplish this branch, the two most significant bits of the CNT register (CNT_3 and CNT_2) must be set to one, a state which can be achieved only by counting above twelve (binary 1100). The result is that there are only four out of sixteen possible states of the CNT register for which the branch in question can be made, which creates one direction or one half of the narrow window.

The second portion of the narrow window is involved in the transfer from the lower group of states back to the upper group. The only way this can occur is from control state eleven to control state one. For this transfer to take place the A register must be set to all zeros, otherwise the circuit will transfer back to control state

REGISTERS: A(3), B(3), CNT(4), Y(1)

INPUTS: X(3), I_1, I_2

OUTPUTS: Z, B_1, C

1. $\underline{A} \leftarrow \underline{X}$; $Y \leftarrow I_1$; $\underline{CNT} \leftarrow INC(\underline{CNT})$

   $\rightarrow (\overline{I_2} \times 2 + I_2 \times 5)$

2. $\underline{B} \leftarrow \omega^3/ADD(\underline{A}, \underline{B})$; $C = \alpha^1/ADD(\underline{A}, \underline{B})$

   $\rightarrow (\overline{Y} \times 3 + Y \times 4)$

3. $\underline{B} \leftarrow \omega^3/ADD(\underline{A}, \underline{B})$; $C = \alpha^1/ADD(\underline{A}, \underline{B})$

   $\rightarrow (4)$

4. $\underline{CNT} \leftarrow (INC(\underline{CNT})) * I_1$

   $\rightarrow ((CNT_1 \wedge CNT_2) \times 8 + (\overline{CNT_1 \wedge CNT_2}) \times 1$

5. $\underline{A} \leftarrow \uparrow(\underline{A} \wedge \underline{B})$

   $\rightarrow (I_2 \times 6 + I_2 \times 7)$

6. $\underline{B} \leftarrow \overline{\varepsilon(3)}$; $Z = 1$

   $\rightarrow (7)$

7. $\underline{CNT} \leftarrow (INC(\underline{CNT})) * Y$

   $\rightarrow ((CNT_1 \wedge CNT_2) \times 8 + (\overline{CNT_1 \wedge CNT_2}) \times 1)$

8. $(\underline{A} \leftarrow \underline{B}) * I_2$; $(\underline{B} \leftarrow \underline{A}) * I_2$

   $\rightarrow (\overline{I_1} \times 9 + I_1 \times 11)$

Figure 3.1 AHPL Description of NWC

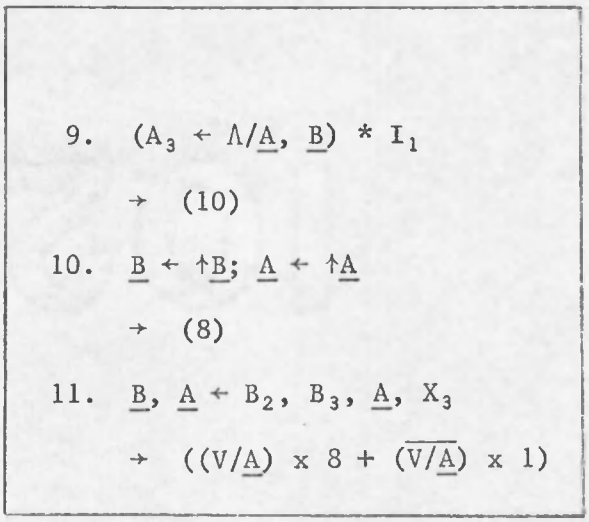


Figure 3.1 (continued)

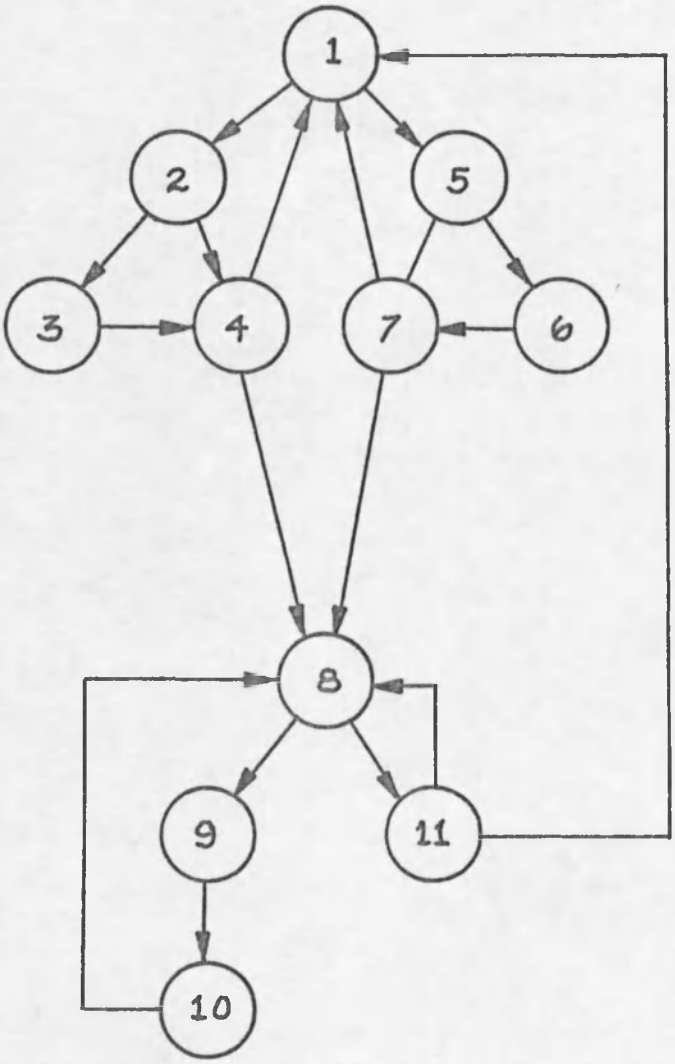


Figure 3.2 Control State Diagram for the Narrow Window Circuit

eight. In this way the narrow window is created. Because of the narrow window between the two groups of states, the circuit will have a tendency to remain in one group or the other for several clock periods, a feature which normally guides a SCIRTSS user to apply a fairly large (over 0.5) weight to the control state disparity adjustment (W_6).

Another feature of the circuit is that the contents of the CNT register are never transferred to any other registers or outputs. The only function of this register is to control the branching from control states four and seven into control state eight. However, faults located in the CNT register can be detected in the same manner as those in any control section. That is, errors in the CNT register will cause erroneous state transfers which will be detectable as faults at the outputs.

In his fault detection experiment, Ng (1974) applied several randomly generated test sequences to the Narrow Window Circuit. None of these were significantly more than 90 percent effective after applications of approximately 300 iterations. SCIRTSS was then used in attempts to better the random results. By using an extensive, manually generated heuristic vector table and then by augmenting SCIRTSS with a computer subroutine, the same results were obtained in 150 iterations. SCIRTSS was twice as efficient as a random input sequence.

With this information as background let us turn to the defined problem at hand, that is, the comparison of heuristically weighted

searches against a breadth-first control run. Although comparisons of other types may be valid, such as the results shown by Ng (1974) versus those realized through the use of heuristically weighted searches in this work, they are put aside for the present and all comparisons are held against the control run. Determination of the control run is then the next item for consideration.

Test Sequence Determination

As in all of the circuits tested, the first priority in the test procedure was to determine the control run. This was accomplished in the following manner.

With the computer-initiated random clock uncontrolled several breadth-first runs were made in order that the run most successful in finding the first 90 percent of all faults could be picked as the control run. The results of these runs are listed in table 3.1. Also run were two weighted searches whose results are also tabulated in table 3.1. Note from these runs that run number four was considerably better than both the other weighted run and all of the breadth-first runs. This was the only circuit in which this occurred. Analyzing the possibilities for this result leads to the conclusion that for the first 90 percent of the active faults the control state disparity index weighting performed as expected and helped route the search through a variety of the control states in the graph tree. For the remaining 10 percent of the faults this proved invalid as will be seen.

In the best breadth-first run the search detected approximately 90.3 percent of the active faults after 103 iterations of test sequence.

TABLE 3.1

INITIAL RUNS FOR THE NARROW WINDOW CIRCUIT

Run No.	For ___% Faults Detected: 20% iterations	20% nodes expanded	40% iterations	40% nodes expanded	75% iterations	75% nodes expanded	90% iterations	90% nodes expanded
1.	5	20	13	45	38	134	103	2016
2.	3	17	10	48	40	133	115	2375
3.	5	20	15	52	42	136	112	2138
4.	5	20	13	45	38	134	65	206
5.	5	20	13	45	42	149	120	1744
6.	5	20	11	52	45	161	103	593

TABLE 3.1 (continued)

Run No.	W_1	W_2	W_3	W_4	W_5	W_6
1.	0	0	0	0	0	0
2.	0	0	0	0	0	0
3.	0	0	0	0	0	0
4.	0	0	0	0	0	.9
5.	.5	.5	.5	.5	.5	.5
6.	0	0	0	0	0	0

It was from this point that all comparisons were made since it was felt that the remaining 10 percent of the faults would form a more challenging task for any search mechanism. From this iteration the breadth-first search was reinitiated with the same setting of the random clock. The results of this control run are plotted first as percent faults detected versus test sequence iterations in figure 3.3 and secondly, as percent faults detected versus expanded nodes in figure 3.4. Also plotted in these two figures are the results of the optimally weighted run. Discussion will turn now to the determination of this optimal run and particularly to the values of the weights (W_i) which led to it.

Again the random clock setting of the control run was noted (11.26.58) and applied to all subsequent runs. Starting from the point where 90.3 percent of the faults had been detected by 101 iterations of test sequence, several runs were made in an attempt to optimize the heuristic search. The results of these runs are tabulated in table 3.2. From these runs several facts emerged. The first was that the optimal run had a control state disparity adjustment of 0.0. This was contrary to what had been previously expected. The reason for this observation seemed to be in the CNT register. In order to make the proper control state transitions the CNT register was impervious to the control state disparity adjustment which was attempting to drive the search into infrequently traversed states. This was in contrast to the Data Shuffle Circuit, for instance, where the control state

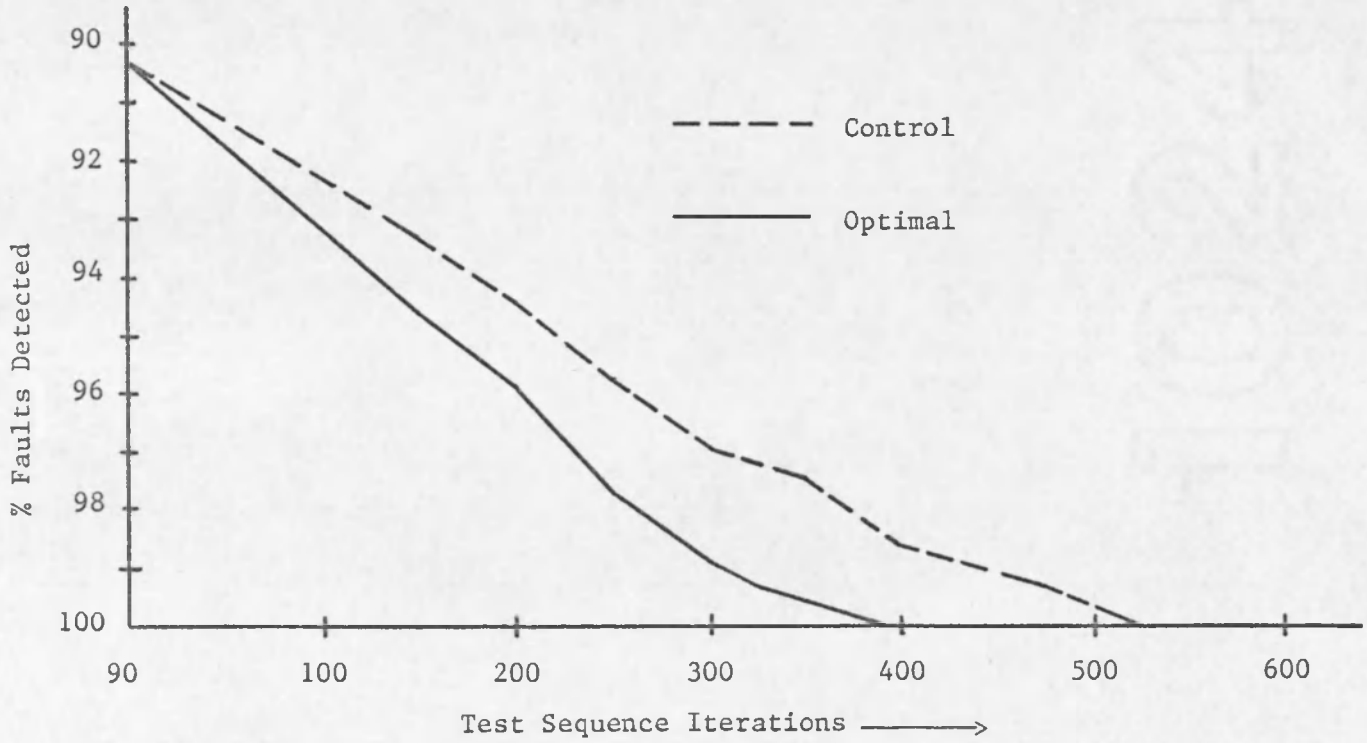


Figure 3.3 Optimal and Control Run Results for the Narrow Window Circuit

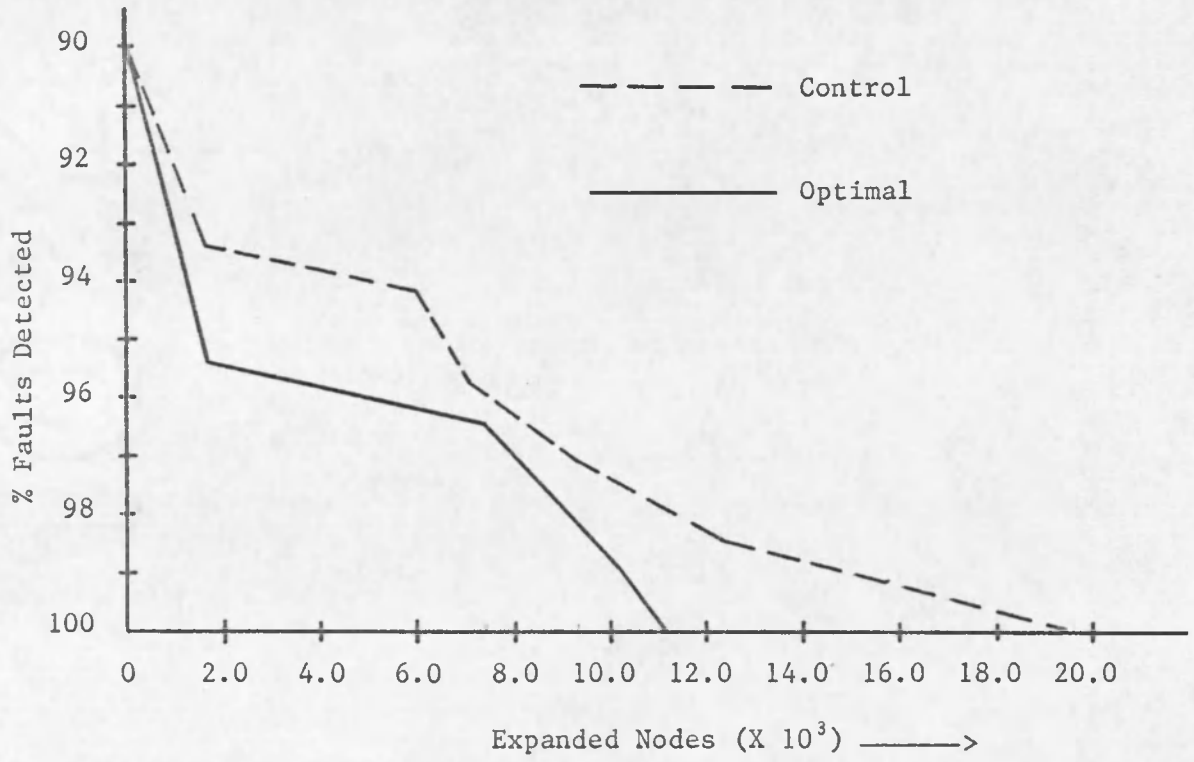


Figure 3.4 Search Results for the Optimal and Control Runs on the Narrow Window Circuit

TABLE 3.2

INITIAL RUNS ON THE NARROW WINDOW CIRCUIT

Run No.	For ___% Faults Detected: 90% iterations	90% nodes expanded	93% iterations	93% nodes expanded	97% iterations	97% nodes expanded	99% iterations	99% nodes expanded	100% it.	100% nodes exp.
1.	101		131	1450	208	7496	300	10062	385	11043
2.	101		124	1113	261	5673	414	11294	———	———
3.	101		131	1450	202	1886	396	10600	454	10806
4.	101		131	1450	266	8489	———	———	———	———
5.	101		145	928	236	2597	433	10910	519	12998
6.	101		130	525	361	7486	———	———	———	———

TABLE 3.2 (continued)

Run No.	W_1	W_2	W_3	W_4	W_5	W_6
1.	.6	.5	.9	.6	.5	.0
2.	.8	.5	.9	.8	.5	.0
3.	.6	.5	.0	.6	.5	.0
4.	.7	.5	.9	.7	.5	.0
5.	.4	.5	.9	.4	.5	.0
6.	.0	.5	.9	.0	.5	.0

disparity adjustment offered considerable assistance in guiding the circuit through all parts of its control state diagram.

Another factor noted was that the runs performed noticeably better where the predecessor node adjustment was the same value in both the sensitization mode searches and the propagation mode searches. This was also true for the Boolean distance adjustment.

For the fault proliferation weighting, an initial setting of 0.9 was used and found to be optimal. Any lowering of this value caused less efficient results no matter what values were assigned to the other weights.

In addition to the above observations of the optimization process, there was another approach taken with this particular circuit. In the optimization process for the Narrow Window Circuit it was noted that in the latter portions of the heuristically weighted runs SCIRTSS was depending heavily upon sensitization mode searches. In an attempt to better the optimized run the flip-flop priority classes were changed so that SCIRTSS could enter propagation mode as soon as a fault had been sensitized to any flip-flop in the circuit. Table 3.3 shows the results of this run which bettered the control run but fell short of the optimal run. In fact, the program entered the propagation mode more often, but there were four instances of propagation search failures as compared to zero in the optimal search.

Now that the optimal run had been determined, it could be meaningfully compared with the control run. In figure 3.3 the control and

TABLE 3.3

RESULTS OF CHANGED PRIORITY CLASS RUN

W_1	W_2	W_3	W_4	W_5	W_6
0.6	0.5	0.0	0.6	0.5	0.0

Iterations (expanded nodes) at

90%	93%	97%	99%	Completion
101	149(767)	243(2843)	364(12449)	432(13331)

NOTE: This may be compared with Run No. 3 on Table 3.2 which used the same heuristic weights. Iterations were improved in this run, but expanded nodes were not as efficient.

optimal runs are compared in terms of percentage of faults detected versus test sequence iterations. In figure 3.4 they are compared in terms of percentage of faults detected versus expanded nodes.

Several facts emerged from an analysis of this comparison. The first was that a least squares analysis of points on the two curves in each figure showed the slopes to be greater for the optimal search in both (.75 vs .56 and .76 vs .57). The significance of this fact is that the heuristically weighted run is approaching completion at a noticeably more rapid rate than in the control run. It was also noted that even though there was a significant disparity in results between the control and optimal searches in terms of test sequence length, the most noticeable difference between the two was in terms of nodes expanded. The control search expanded 77.57 percent more nodes in order to obtain the same results as the optimal search. Both searches found all but one of the detectable faults. These facts clearly indicated greatly improved efficiency of heuristic search over the control search in terms of determining optimal path through control state graph expansions. These results also validate the heuristic equations given by Hill and Huey (1975) which, as given previously, employ the weights to determine optimal path.

CHAPTER 4

THE MEMORY-EMBEDDED CIRCUIT

The last circuit tested was called the Memory-Embedded Circuit because of its internal configuration of an eight-bit register which modeled a random access memory unit. The design of the circuit was taken from a random access memory circuit having eight three bit words. The basic design was then modified such that it would have fewer outputs, and each of the three bit words was changed to become a one bit word or register. In this way the circuit could access only one bit at a time. Because of this change in word size and decrease in the number of outputs it remained similarly difficult for faults to be driven out for detection. The significance of this result becomes apparent in the logic description of the circuit shown in figure 4.1. The AHPL description of the circuit is given in figure 4.2.

A brief analysis of the logic description of the circuit and the AHPL description first revealed that, due to the physical configuration of the circuit, faults would be rather easily sensitized to registers or flip-flops but more difficult to drive to output. Secondly, the fact that the circuit had only a mildly complicated control sequence meant that the search would probably be successful at detecting faults in it in the early stages of the search. These also seem to be observations which may add to an ability to classify circuits

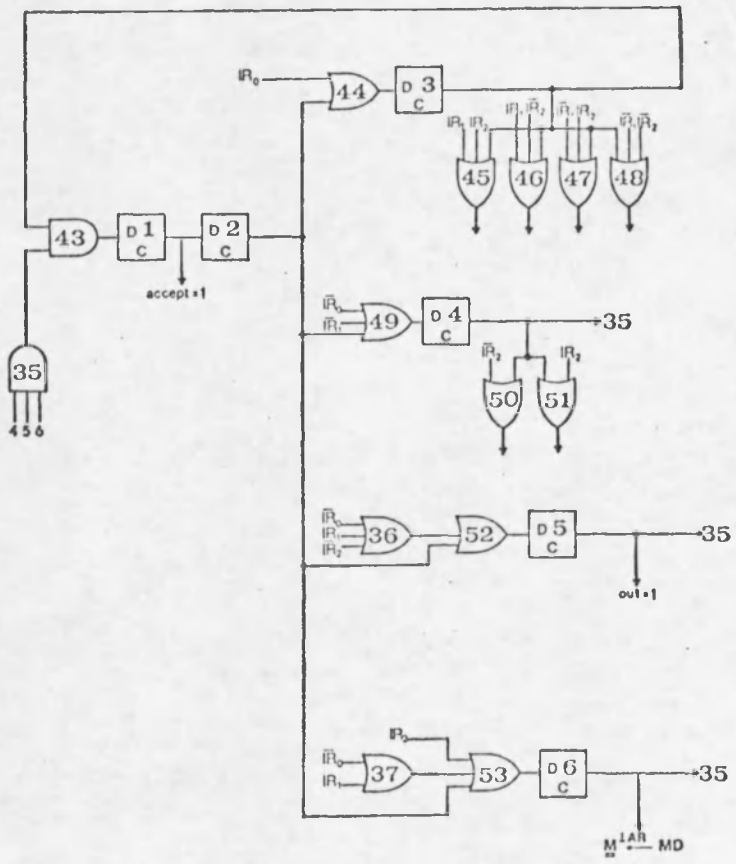


Figure 4.1 Logic Block Diagram of the Memory-Embedded Circuit

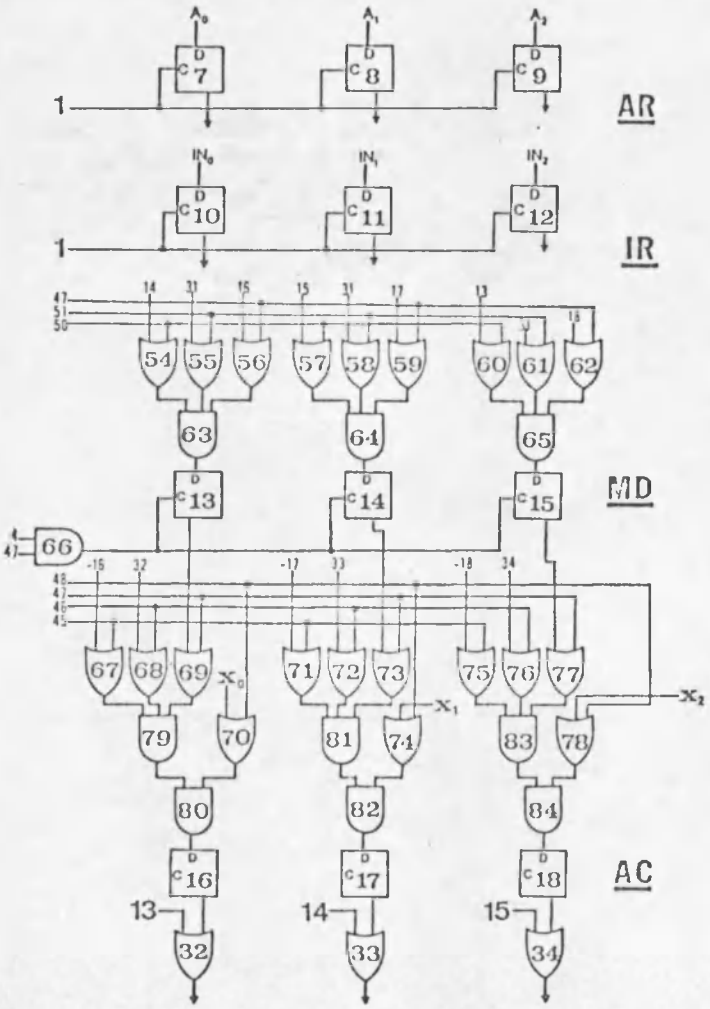


Figure 4.1 (continued) Logic Block Diagram of the Memory-Embedded Circuit

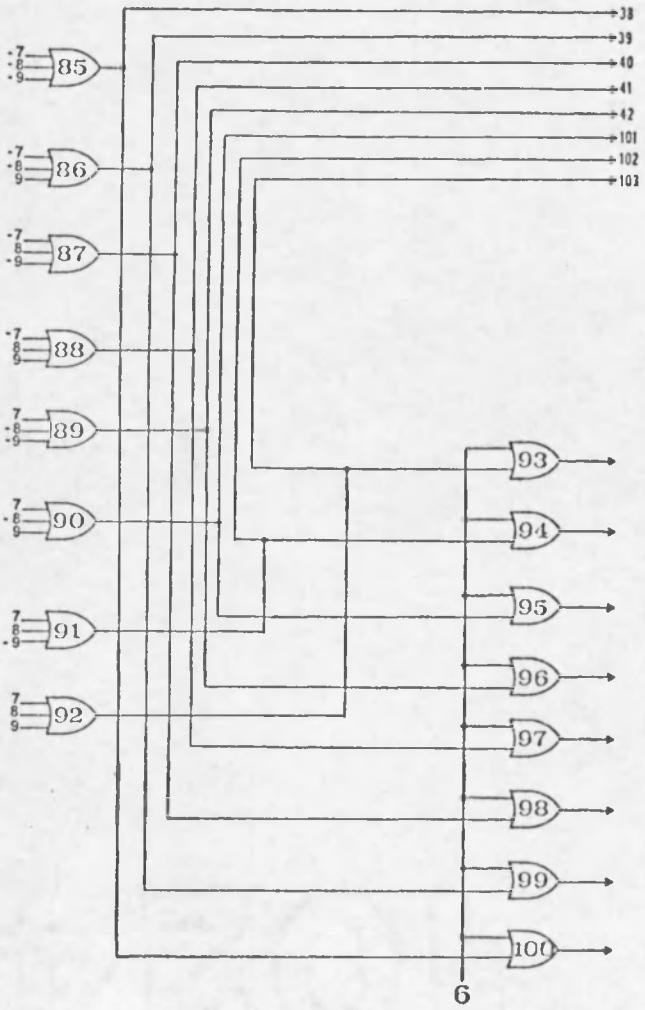


Figure 4.1 (continued) Logic Block Diagram of the Memory-Embedded Circuit

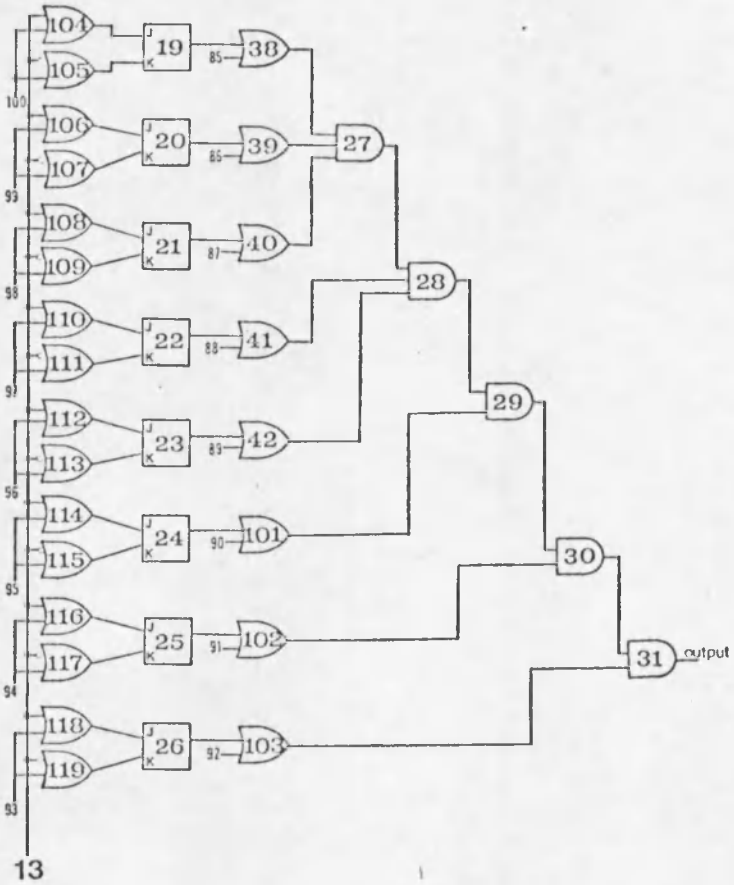


Figure 4.1 (continued) Logic Block Diagram of the Memory-Embedded Circuit

REGISTERS: $M_0(1)...M^7(1)$, MD(3), AC(3), IR(3), AR(3)

INPUTS: IN(3), A(3), X(3)

OUTPUTS: AC(3), out, accept, input

1. $\underline{AR} \leftarrow \underline{A}$; $\underline{IR} \leftarrow \underline{IN}$; accept = 1

   $\rightarrow$ (2)

2. $\rightarrow$ $(IR_0 \times 3 + \overline{IR}_0 \wedge \overline{IR}_1 \times 4 + \overline{IR}_0 \wedge IR_1 \wedge \overline{IR}_2 \times 5 + \overline{IR}_0 \wedge IR_1 \wedge IR_2 \times 6)$

3. $\underline{AC} \leftarrow \overline{\underline{AC}} * (IR_1 \wedge IR_2) \vee (\underline{AC} \wedge \underline{MD}) * (IR_1 \wedge \overline{IR}_2) \vee \underline{MD} * (IR_1 \wedge IR_2) \vee \underline{X} * (\overline{IR}_1 \wedge IR_2)$; $MD \leftarrow AC * (\overline{IR}_1 \wedge IR_2)$; $(input = 1) * (\overline{IR}_1 \wedge IR_2)$

   $\rightarrow$ (1)

4. $\underline{MD} \leftarrow (\uparrow \underline{MD} * \overline{IR}_2) \vee (\underline{\underline{M}}^{\underline{\underline{IAR}}} * IR_2)$

   $\rightarrow$ (1)

5. out = 1

   $\rightarrow$ (1)

6. $\underline{\underline{M}}^{\underline{\underline{IAR}}} \leftarrow MD_0$

   $\rightarrow$ (1)

Figure 4.2 AHPL Description of the Memory-Embedded Circuit

because of a similarity to random access memory circuits. One further point should be that consideration should be given to the possibility that the testing of this circuit might give some indication as to the ability of heuristically weighted SCIRTSS searches and breadth-first searches to cope with conditional AHPL transfers. This is borne out by inspection of figure 4.2. In it, control state number three can be seen to contain fairly formidable conditional transfers. Figure 4.1 reveals the logic structure associated with these transfers. It can also be noted that, other than control state four, these are the only portions of the control sequence having conditional transfers. There are no others with any major conditional transfers.

Test Sequence Determination

As in all of the circuits tested, the first portion of the experiment was to determine a control run against which attempts at optimization would be compared. With an initial time limit of one-hundred seconds imposed on all runs, several runs were made. First, a group of runs was made as breadth-first searches. As before, this meant that all heuristic weights were set to zero, and the searches expanded unaided by the heuristic valves provided by the equations given in Chapter 1. Secondly, an additional group of runs was made with various values assigned to the heuristic weights (W_i). In all of the above mentioned runs the random clock was allowed to run uncontrolled. The results of all of these runs are given in table 4.1. From these results it was determined that the most efficient run in terms of both faults

TABLE 4.1

INITIAL SEARCHES OF THE MEMORY-EMBEDDED CIRCUIT

	For ___% Faults Detected							
	20%		30%		40%		42.2%	
Run No.	iterations	nodes expanded	iterations	nodes expanded	iterations	nodes expanded	iterations	nodes expanded
1.	18	227	39	5068	57	10052	60	10076
2.	18	227	38	5013	—	——	—	——
3.	22	249	40	4934	—	——	—	——
4.	18	226	36	9621	—	——	—	——
5.	20	452	46	8734	—	——	—	——
6.	18	227	43	9152	—	——	—	——

TABLE 4.1 (continued)

Run No.	W_1	W_2	W_3	W_4	W_5	W_6
4.	.9	.9	.9	.9	.9	.9
5.	.2	.2	.2	.2	.2	.2
6.	.0	.5	.5	.0	.5	.0

All other runs Wi = 0 i = 1, 2, 3, 4, 5, 6

detected per test sequence iteration and per expanded node was the breadth-first search which in the one hundred second time limit detected 42.2 percent of the active faults. At this point it was determined that this run would be used for the control. Its random clock setting (14.25.23) was noted so that all subsequent runs could be implemented with it. The control run was then continued for considerable additional search time (2000 seconds) with this setting and the basis for the control run was considered established. The results of this completed control run are plotted as a portion of figure 4.3 in the form of percentage of faults detected versus expanded nodes. In figure 4.4 the results are plotted with the optimal run as percentage of faults detected versus test sequence iterations. It should be noted from this figure that the control run appears to be more efficient than either of the other two runs plotted. The reason for this appearance is that the control run is using a very broad, shallow search while the weighted searches, guided primarily by the Boolean distance adjustment weight, are penetrating quite deeply and narrowly into the graph tree in search of goal nodes. For this particular circuit shallow location of some goal nodes enabled the control run to perform efficiently in terms of test sequence length. This is only half the story, however. In twice the computer execution time and over two times the number of nodes expanded the control run was only slightly more efficient in test sequence length and still only reached a point of 80 percent faults detected.

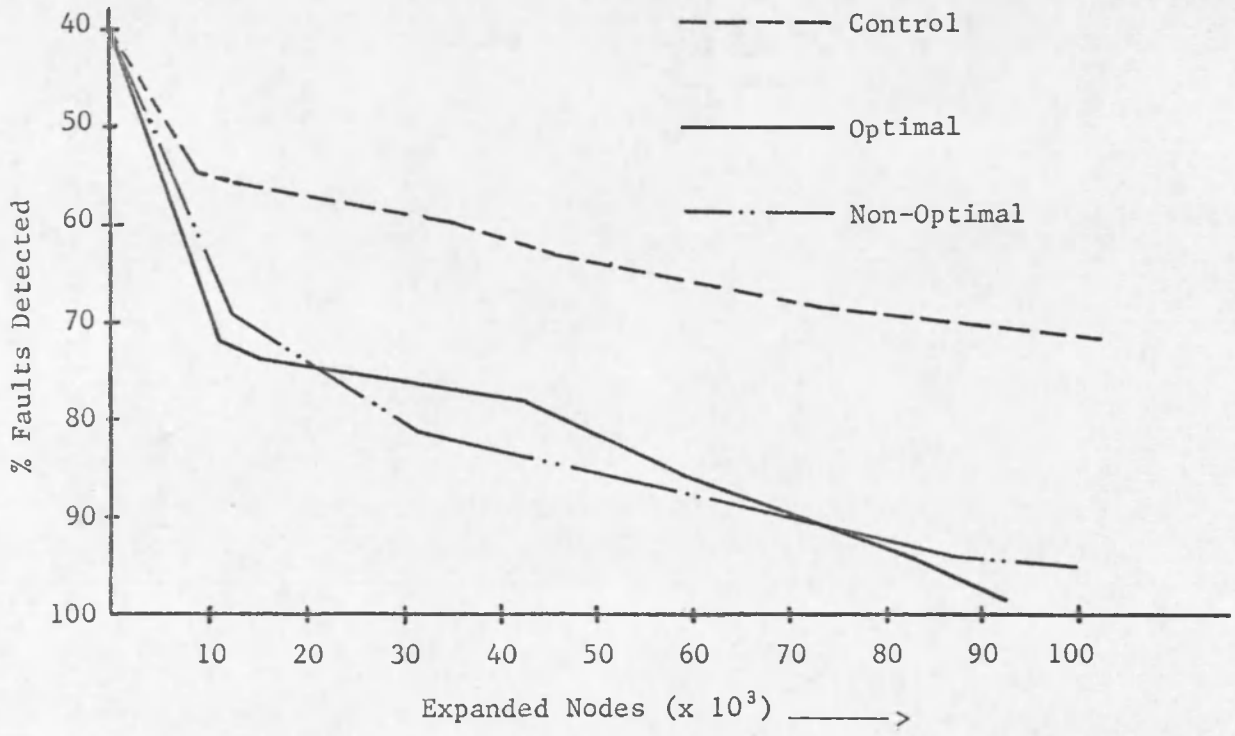


Figure 4.3 Results of Control and Weighted Runs for the Memory-Embedded Circuit

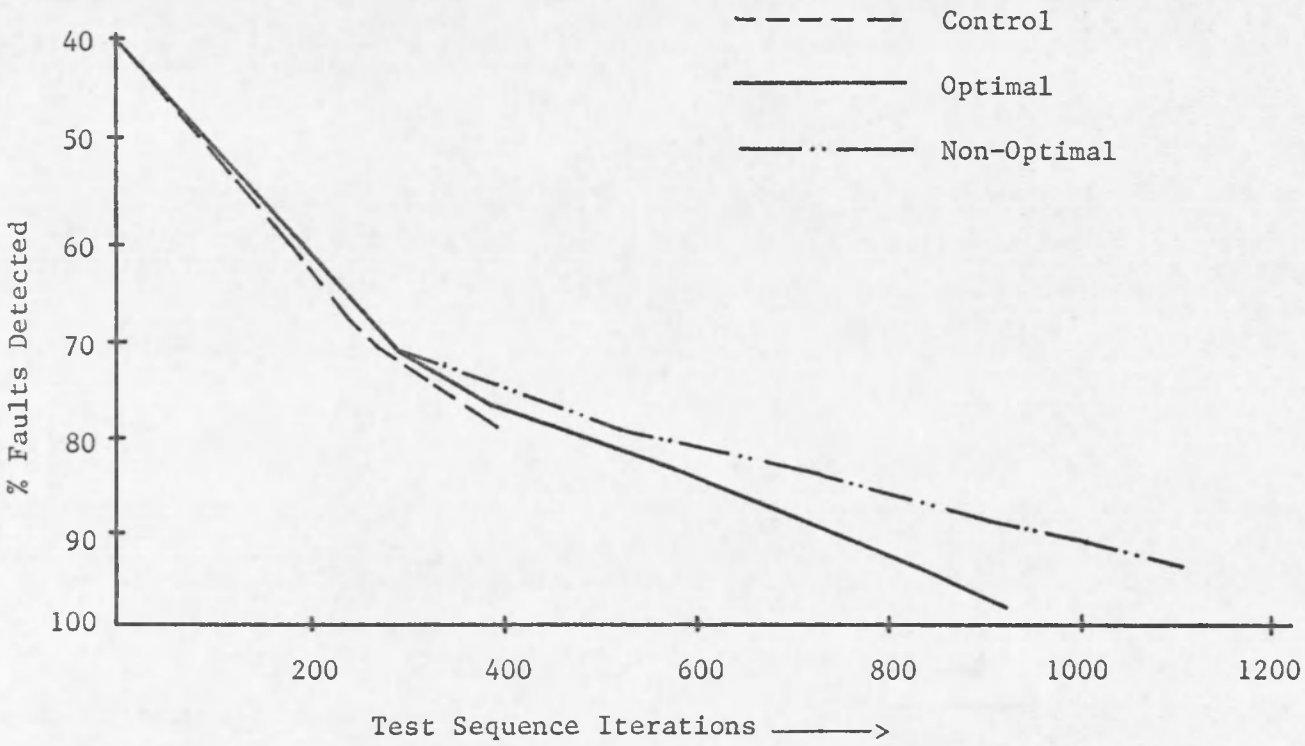


Figure 4.4 Results of Control and Weighted Runs on the Memory-Embedded Circuit

For this particular circuit shallow location of some goal nodes enabled the control run to perform efficiently in terms of test sequence length. This is only half the story, however. In twice the computer execution time and over two times the number of nodes expanded the control run was only slightly more efficient in test sequence length and still only reached a point of 80 percent faults detected.

From this point (52.2 percent of the active faults detected) the attempt was made to determine an optimal test sequence. This was accomplished in the following manner. Again, a one hundred second time limit was set and several different combinations of weights were injected into the heuristic value equations. The results of these runs are recorded in Table 4.2. Since this circuit was a formidable test for the SCIRTSS system in terms of searching to completion of one hundred percent of faults detected, it was decided to pick the two most efficient of these interim runs and carry them to completion. The results of these completed runs are plotted along with the control run in figure 4.3. While the differences between the optimal and interim runs were not particularly dramatic in figure 4.3 there can be more observed in figure 4.4. The aspects of the control run have already been examined in figure 4.4, but the differences in efficiency between the two weighted searches can be better demonstrated. The difference in test sequence length for similar percentage detection is due primarily to lowered predecessor node adjustment and fault proliferation adjustment weighting in the optimally weighted run.

TABLE 4.2

INTERIM RUNS ON THE MEMORY-EMBEDDED CIRCUIT

Run No.	For ___% Faults Detected: 53.5% iterations	53.5% nodes expanded	55% iterations	55% nodes expanded	65% iterations	65% nodes expanded	72% iterations	72% nodes expanded
1.	126		159	543	198	838	291	992
2.	126		159	543	185	921	291	1015
3.	126		159	7946	—	—	—	—
4.	126		162	14047	—	—	—	—
5.	126		—	—	—	—	—	—
6.	126		—	—	—	—	—	—

TABLE 4.2 (continued)

Run No.	W_1	W_2	W_3	W_4	W_5	W_6
1.	.0	.5	.5	.0	.5	.0
2.	.5	.5	.9	.0	.0	.5
3.	.2	.2	.5	.5	.5	.2
4.	.5	.5	.5	.5	.5	.5
5.	.2	.2	.2	.2	.2	.2
6.	.5	.5	.9	.5	.5	.5

Several observations were made starting at the point where the relative efficiencies of the control run and the weighted runs began to diverge. Based upon analysis of the circuit itself, two sets of heuristic weights were applied which enabled SCIRTSS to detect 18.3 percent of the remaining faults in the same approximate amount of time and number of expanded nodes that the control search was using to detect only 1.0 percent of the remaining faults.

It was noted here that the success of the search was significantly improved by increasing the value of the Boolean distance adjustment relative to that of the predecessor node adjustment. This was done by setting the Boolean distance adjustment to 0.5 while reducing the predecessor node adjustment to zero. This particular combination of adjustments allowed the search to attain considerable depth into the graph tree and find goal nodes with a minimum expansion of nodes. The higher weighting of the Boolean distance adjustment was quite effective since the nature of the memory circuit allowed small portions of the circuit to be changed without effecting the remainder. This allowed for gradual but successful approach of a goal node. This was particularly true in the sensitization mode searches. This pattern was definitely present from the point of 52.2 percent fault detection to completion of the runs. It was in fact this combination which optimized the better run. This resulted in a decreased average number of iterations in each sensitization search within optimal run.

A third factor also became important in the calculation of heuristics. This was the fault proliferation weighting. After inspection

of the circuit, it was immediately apparent that faults would be fairly easy to embed in flip-flops but considerably more difficult to propagate to outputs. In all of the circuits tested a relatively high setting for the fault proliferation weighting had been beneficial. This was initially attempted in the case of the Memory-Embedded Circuit. But it was discovered that for higher settings of this value the length in iterations of the sensitization searches increased. After several of the interim runs it was determined that a median weight of 0.5 worked the best. This was significant in that it was the lowest beneficial setting of this adjustment for any circuit tested. This is also significant in that the fault proliferation weight is not present in the computation of the heuristic values used by the sensitization searches.

Although, as stated, the fault proliferation weighting does not interact with the sensitization mode searches, it did seem to have some indirect effect in the case of this particular circuit. Consider the magnitude of the difference in performance between the optimal and non-optimal completed runs. The lesser of the two completed runs expanded an average (arithmetic mean) of 2863.7 nodes in each sensitization mode search. The optimal run expanded an average of only 1198.6 nodes in each of its sensitization mode searches. This meant that in terms of reaching a goal node the optimal search was over twice as efficient in driving faults into registers or flip-flops.

A cursory inspection of figure 4.3 shows the great advantage enjoyed by the heuristically weighted search when compared in terms of

percentage of faults detected versus expanded nodes. This seems to be the true test of SCIRTSS efficiency. Since iterations relate directly to Boolean distance between start node and goal node, it was not surprising to discover that in terms of iteration efficiency, the control search performed about as well as the optimal search and better than the lesser completed search. The optimal search had detected 77.8 percent of the faults in 386 iterations and the control search had detected 80.1 percent in 390 iterations.

Search Comparison

It is for this circuit that an example of the difference between the approach of the breadth-first search and the heuristically weighted search is presented. This example should also point out the reasons for the previously stated observations. For instance, take a comparison between two fault propagation mode searches, both with identical starting points. Both searches occurred as the runs had reached a point at which 53.5 percent of the faults had been detected. Both were in control state one, with one fault stored and available for propagation. Propagation of this fault to output was the goal of the search. The control search, expanding along every possible path in a breadth-first search, found a goal node six iterations from the starting node after expanding 799 nodes. The optimal search found a goal node nine iterations from the starting node after expanding only 142 nodes. So, for this particular propagation search a breadth-first approach required 478.2 percent more expanded nodes than the optimal

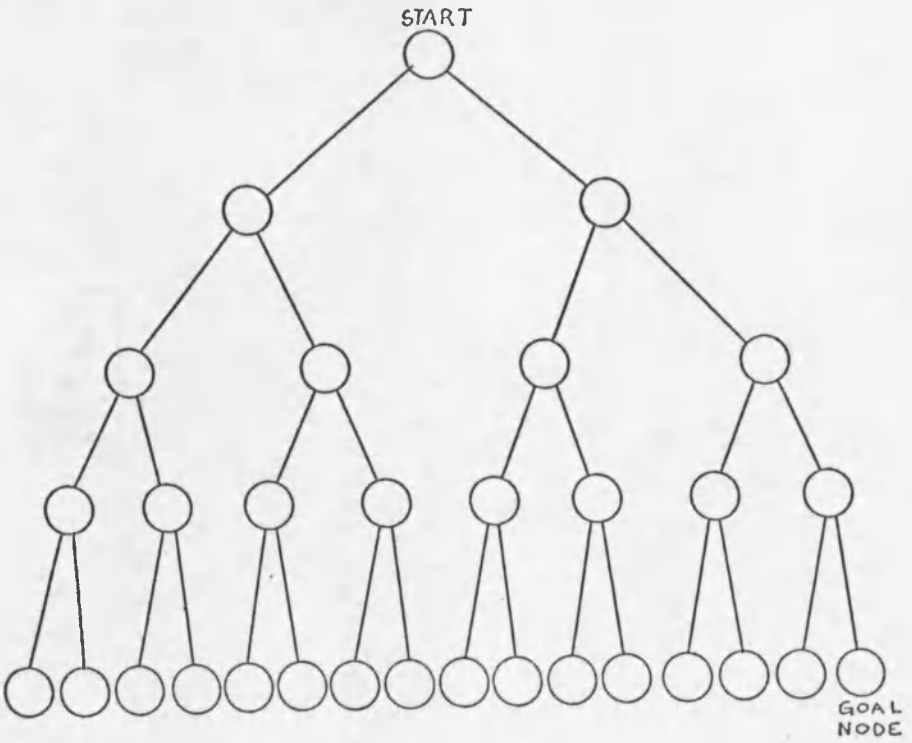
search. The SCIRTSS computer print-outs for the breadth-first search and the optimal search are shown in figures 4.5 and 4.6 respectively. As an example of what is occurring in this instance consider the following simulated graphical example. In figure 4.7 two graph tree examples are shown. In graph (A) a breadth-first expansion of the tree has been performed and in order to find the goal node 31 nodes have been expanded. Obviously, if the search is allowed to continue expanding indefinitely in this manner it will eventually find the goal node if it is attainable. The goal node is found a distance of 4 iterations from the starting node. In graph (B) the graph is expanded based upon the heuristic values shown on each leg. Only the nodes shown crossed have actually been expanded. In this way the same goal node, four iterations from the starting node, was reached by expanding only 10 nodes. This then seems to be the real benefit of the heuristic graph expansions. SCIRTSS heuristically weighted searches can compare only equally with searches based upon completely brute force application as in the case of the breadth-first searches. It is in the area of time and computational efficiency, reflected in the number of nodes expanded, that provides a true means of comparison.

```
A TOTAL OF 140 FAULTS OF ALL CLASSES ARE UNDETECTED:  46.5 PERCENT.
          1 FAULTS ARE STORED IN FLIP-FLOPS.

SELECT  1 OF   1 ELIGIBLE CLASS 3 FAULTS STORED IN CLASS 1
     FLIP-FLOPS FOR FAULT PROPAGATION:          JKFF  19  OUTPUT  S-A-0.


SEARCH (FPS/1) CALL NUMBER   2 RESULTS:
---------------------------------------

          START NODE: 1000000000111111110DU111111

SEARCH SUCCESSFUL AFTER  799 NODE SIMULATIONS.

   STEP     C.S.     EXTERNAL INPUTS/FLIP-FLOP STATES
   ----     ----     --------------------------------
     0.       1      000000000/100000000011111110DU111111
     1.       2      000001111/010000000011111110DU111111
     2.       4      010111100/001000000011111110DU111111
     3.       1      000010000/100000000001DDD110DU111111
     4.       2      101110000/010000101110DDD110DU111111
     5.       3      000001110/001000101110DDD110DU111111
     6.       1      111001000/100000101110DDDDDC0U111111


FAULT SIMULATOR CALL NUMBER   3 RESULTS:
---------------------------------------

130  00000111
     1        1000000000111111101U111111

131  01011110
     0        0100000000011111101U111111

132  00001000
     0        0001000000011111101U111111

133  10111000
     0        1000000000011111101U111111

134  00000111
     0        0100001011101111101U111111

135  11100100
     0        0010001011101111101U111111

135+          1000001011101111101U111111
                                  162. [ 137]  JKFF   19  OUTPUT  S-A-0.
                                  163. [ 2 1]  OR     27  INPUT 1 S-A-0.
                                  164. [ 655]  AND    85  OUTPUT  S-A-0.
```

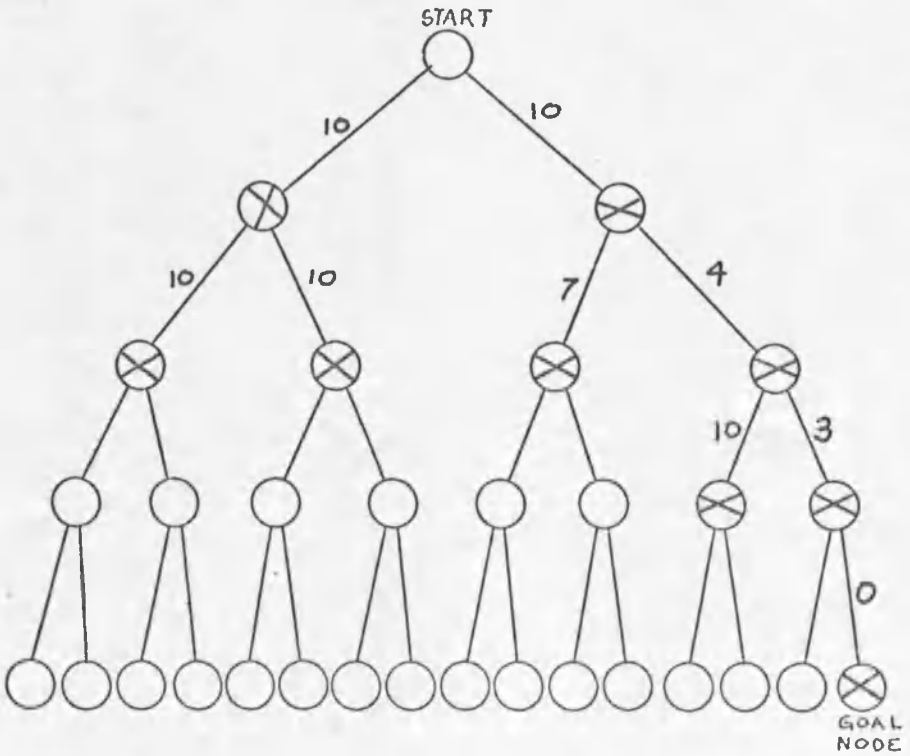
Figure 4.5 Example Printout For Breadth-first Search

```
A TOTAL OF 140 FAULTS OF ALL CLASSES ARE UNDETECTED:  46.5 PERCENT.
          1 FAULTS ARE STORED IN FLIP-FLOPS.

SELECT  1 OF    1 ELIGIBLE CLASS 3 FAULTS STORED IN CLASS 1
     FLIP-FLOPS FOR FAULT PROPAGATION:          JKFF  19  OUTPUT  S-A-0.


SEARCH (FPS/1) CALL NUMBER   2 RESULTS:
---------------------------------------

          START NODE: 1000000000111111110DU111111

SEARCH SUCCESSFUL AFTER  142 NODE SIMULATIONS.

   STEP     C.S.     EXTERNAL INPUTS/FLIP-FLOP STATES
   ----     ----     --------------------------------
     0.       1      000000000/1000000000011111110DU111111
     1.       2      000001111/0100000000001111110DU111111
     2.       4      010111100/0001000000001111110DU111111
     3.       1      000010000/100000000000DDD110DU111111
     4.       2      110111011/0100001101111DDD110DU111111
     5.       3      110111101/0010001101111DDD110DU111111
     6.       1      101111000/1000001101111DDDU01DU111111
     7.       2      101110001/0100001011110DDDD01DU111111
     8.       3      110011001/0010001011110DDDD01DU111111
     9.       1      111110000/1000001011110DDDD0DDU111111


FAULT SIMULATOR CALL NUMBER   3 RESULTS:
-----------------------------------------

130  00000111
     1        1000000000011111110lU111111

131  01011110
     0        0100000000001111110lU111111

132  00001000
     0        0001000000001111110lU111111

133  11011101
     1        1000000000001111110lU111111

134  11011110
     1        0100001101111111110lU111111

135  10111100
     0        0010001101111111110lU111111

136  10111000
     1        1000001101111111001lU111111

137  11001100
     1        0100001011101110011U111111

138  11111000
     0        0010001011101110011U111111

138*          1000001011101110011U111111
                                 162. [ 137]  JKFF   19  OUTPUT  S-A-0.
                                 163. [ 2 1]  OR     27  INPUT 1 S-A-0.
                                 164. [ 655]  AND    85  OUTPUT  S-A-0.
```

Figure 4.6 Example Printout For Weighted Search



Graph (A)



Graph (B)

Figure 4.7 Example Graph Trees

CHAPTER 5

CONCLUSIONS

Before any additional conclusions can be drawn regarding the specifics and minor observations of the analysis undertaken, the first and most major question presented must be answered. That question was simply whether or not the SCIRTSS system was more efficient in determining a successful test sequence than a breadth-first search performed on the same circuit. In answer to that question let us turn to the results obtained in the previous chapters concerning the example circuits.

In past works on this subject the most common method of comparison for SCIRTSS results against other methods has been in terms of test sequence length for a given number of faults detected or the percentage of faults detected. For the example circuits consider how well SCIRTSS performed in terms of faults detected per test sequence iteration. First, in the Data Shuffle Circuit, starting from the test origination point the control run never equaled in efficiency the best heuristically weighted search. The answer for the Data Shuffle Circuit must be that the heuristic search was, in fact, more efficient in terms of test sequence length per fault detected.

In the case of the Narrow Window Circuit the results are even more dramatic. As was pointed out in Chapter 3, not only did the

heuristic search perform more efficiently, but as both the optimal and control searches neared completion the difference in efficiency between the two actually widened. The optimal search reached its final point 134 iterations before the control search. This amounted to a difference of 25.5 percent in search efficiency. Here again, it is clear that the weighted search performed more efficiently in terms of percentage of faults detected per test sequence iteration.

For the Memory-Embedded circuit the results appear initially to be somewhat different. First, as was stated in Chapter 4, the control run, had done slightly better than the optimal search up to the point where 72.5 percent of the faults had been detected. It detected 72.5 percent of the active faults after 276 iterations while the optimal search required 294 iterations to reach the same 72.5 percent level. There is, however, an explanation for why this occurred. An analysis of the optimal and control runs for the Memory-Embedded circuit revealed that because of an NSIM call limit of 2000 nodes for both runs the control run was failing in many of its sub-searches (propagation and sensitization) due to shallow but broad penetration of the graph tree. When it did reach a goal node it was relatively close to the start node in terms of test sequence iterations. In the case of the heuristically weighted search there were very few sub-search failures, but penetrations of the graph tree were extremely narrow and deep. This manifested itself in the above stated results. This situation appeared from all available evidence to be

circuit related in that circuits with either a narrower graph tree or deeply embedded goal nodes were not similarly affected.

Turning now to comparisons based upon percentage of faults detected for a given number of expanded nodes, the example circuits can be examined. Again the question is asked, do the heuristically weighted searches perform more efficiently than the control searches? Consider again the Data Shuffle Circuit. It can be seen by re-examining figure 2.3 that although the control search was actually ahead of the optimal search until over 97 percent of the faults in the circuit had been detected the optimal had a higher slope from the 93 percent point to completion. This allowed the optimal run to overtake the control run and complete its fault detections 373 node expansions in advance of the control run. So, for the Data Shuffle Circuit it can be seen that while a lapse did occur there was definitely an improvement in efficiency in the optimal run in terms of number of faults detected per expanded node.

In the Narrow Window Circuit some additional observations can be made. In Chapter 3 (figures 3.3 and 3.4) it was shown that in terms of faults detected per test sequence iteration and per expanded node the optimal run diverged from the control run at the same rate in both. In this circuit the control run never did as well in any portion of the search. In fact, in terms of number of nodes expanded, SCIRTSS, with the benefit of heuristically computed values, out-performed the breadth-first method of searching.

For the Memory-Embedded circuit this particular means of comparison between the optimal and control runs was the most dramatic. After an equal number of node expansions (92,000) the optimal run had succeeded in detecting over 98 percent of the active circuit faults while the control run had detected only 69.5 percent. After expanding 99,438 more nodes the control run had only detected ten percent more of the active faults or a total of 80.1 percent. Clearly the weighted search could be judged more efficient than the breadth-first search.

Let us turn now to conclusions regarding the assignment and/or significance of the various weights used in the heuristic value calculations. The first characteristic which all three example circuits had in common was the fact that up to a level of 40 to 50 percent of the active faults detected, breadth-first searches appeared to be more efficient than weighted searches. The exact reason for this observation was not determined, but one possibility for further study might be whether or not the breadth-first searches initally detected more easily found faults. This type of fault may be linked to large groups of faults which could then be driven out in the fault simulation portion of the program.

Another interesting conclusion was drawn from the common experience on all three example circuits. This was the detrimental effect on searches caused by non-optimal weight assignments. It was observed in all circuits tested and was determined to have been caused by high control state disparity index adjustment settings in circuits that had to stay in a small number of control states in order to

exercise a certain portion of the circuit. Another factor causing this was the Boolean distance adjustments being improperly set too high which caused searches to penetrate deeply into the graph tree in search of a goal node while another goal node not so far into the graph tree could have been found more quickly by a breadth-first search. A zero setting of the fault proliferation adjustment while other weights were set above zero caused detrimental effects due to inefficient fault propagation mode searches.

Search Sensitivity to Weight Changes

One facet of the run results which must be analyzed is that of weight sensitivity. To which weights is the search most sensitive? How sensitive is the search to those particular weights? First, it was seen that in the Narrow Window Circuit small changes (ΔW_2 and ΔW_5) in the Boolean distance weighting in both the fault propagation and sensitization mode searches caused noticeable changes in the search efficiency. This was not as noticeable in other circuits. The same type of result was noted for the fault proliferation weighting in the Memory-Embedded circuit. The conclusion drawn from these observations can be only that Boolean distance weighting and fault proliferation weighting are candidates for weights to which the search mechanism seems particularly sensitive. However, it must also be concluded that these sensitivities are at best extremely circuit dependent and general statements regarding their performance on all circuits simply cannot at this point be made.

Circuit Dependent Tendencies

Another area of interest which must be discussed is that of optimal weight settings which are circuit dependent. From previous discussion in this chapter and from material presented in other chapters it is apparent that optimal weight assignments for certain weights differ between circuits. It can therefore be definitely concluded that optimal settings are to some degree circuit dependent. Specifically, the optimal control state disparity index weighting is extremely variant from circuit to circuit, but a brief circuit analysis can enable that its proper setting be made. Optimal Boolean distance weighting in both propagation and sensitization modes varied, but in none of the circuits tested was it optimal at a setting lower than 0.5. The optimal fault proliferation weighting corresponded very closely to that of the Boolean distance weighting. It was lower where the Boolean distance weighting was lower and higher where the Boolean distance weighting was higher. Although analysis did not reveal at this point the cause for this action it can be stated that there is evidence that the same circuit characteristics which cause one setting probably cause another and should be considered in any further study on the subject.

The optimal predecessor node adjustment varied considerably on each circuit. Its optimal value is therefore extremely circuit dependent. However, after analysis of the test circuit results specific trends in the predecessor node adjustment did not lend themselves to categorization.

Overall it must be concluded that all the optimal weight settings are circuit dependent to some degree. Only the control state disparity index weighting is highly predictable and can be set fairly accurately from circuit analysis.

Circuit Classifications

Earlier in this paper it was stated that based upon results obtained in the fault detection experiments an attempt would be made to classify circuits such that specific settings for certain weights would apply to certain classes of circuits. After all circuit run results had been reviewed several conclusions were reached. The first of these was that the most easily identifiable tool for circuit classification is the control state diagram. Characteristics such as the Narrow Window in that circuit and the expanding and contracting tree in the Data Shuffle related directly to the values picked for optimal weights. This, combined with a cursory inspection of the AHPL description can offer a relatively large aid in properly setting up the heuristic value equations for specific circuits. Specifically, the results offered the following conclusions.

Based upon the material studied in this and previous works on the subject it is evident that circuits with internal counting registers fall into two catagories; those where the counting process affects transitions between isolated portions of the control state diagram and those where it does not. Where the counting process does affect transitions to and from isolated portions the most important weight is the

control state disparity index weight. A high setting of this weight causes detrimental effects in the later stages of the search process. The opposite is true for circuits with counting registers which do not affect such transitions. Other than the above stated observations, no concrete evidence was available for the formation of circuit classes. Unless future research presents a better means of classification, it is the conclusion of this paper that each circuit must be taken as an individual case. Its characteristics must be observed and weighed in conjunction with search performance in each case in order to pick an optimal set of heuristic weights.

Optimization Process

It is at this point in the compilation of conclusions that all the empirical, analytical and intuitive results should be drawn together and presented in the form of assistance to any user of the SCIRTSS program to help in determining the best test sequence in terms of economy and efficiency. The analysis of the circuit and optimization of the weights should proceed in the following manner.

First, the operator should determine whether SCIRTSS will require a significant amount of running time (over 100 seconds) in order to detect 100 percent of the circuits active faults. Circuits with an excess of 150 elements will generally fall into this category. In this situation at least two breadth-first searches should be made on the circuit with about 50 seconds running time each. In this way most of

the fairly accessible faults will be detected in probably the shortest possible test sequence.

From these runs the best one should be picked and the point at which approximately 50 percent of the faults have been detected should be recreated by an octal input sequence as a basis for future runs. From this point heuristically weighted searches have an extremely high probability of improving the efficiency of the total run. The best method for commencing this procedure is to initially use a high (0.9) setting for the fault proliferation adjustment. In addition the Boolean distance adjustment in both fault propagation mode and sensitization mode should be started at 0.5. The best initial setting for the predecessor node adjustment is zero. The circuit should then be analyzed to determine the proper setting for the control state disparity index weighting. The information presented in the circuit classification section of this chapter should be heeded. If the circuit relies on either anti-random or counting processes for certain isolated transitions in its control state diagram then this weight should be set to zero. If not, an initial value of 0.5 provides a good starting point.

After some (two or three) runs the operator can then apply an optimization process to the weight settings. If propagation mode searches are longer in terms of nodes expanded in the weighted runs than in the breadth-first run, the Boolean distance adjustment setting for this mode should be increased slightly. Also, the predecessor node adjustment should be raised to around 0.5. If the propagation mode

searches become even longer, the fault proliferation adjustment setting should be lowered for better results. Also, check the sensitization mode searches. If there are any search failures the Boolean distance adjustment setting should be increased to 0.8 or 0.9 in order to achieve greater depth into the graph tree. Iteration efficiency could possibly suffer, but run economy will probably benefit.

It must be remembered that these are only guidelines in an optimization process which has been concluded to be essentially intuitive and highly circuit dependent. It has also been concluded that from past results and current data that sufficient material is not provided to form the basis for any general classifications of circuits such that a specific set of weights always optimizes searches on a certain class of circuits. However, the results obtained do indicate that there is basis for at least partial classifications. With more extensive work on both digital circuits in general and more specifically AHPL designed circuits, more rigorous means of classifications may become available.

LIST OF REFERENCES

Belt, J. E., "A Heuristic Search Approach to Test Sequence Generation for AHPL Described Synchronous Sequential Circuits," Ph.D. dissertation, Department of Electrical Engineering, University of Arizona, 1973.

Carter, E. A., "Fault Test Generation for Sequential Circuits Described in AHPL," Ph.D. dissertation, Department of Electrical Engineering, University of Arizona, 1973.

Hill, F. J., and Huey, B. M., (unpublished) "SCIRTSS: A Search System for Sequential Circuit Test Sequences," Department of Electrical Engineering, University of Arizona, 1975.

Hill, F. J., and Peterson, G. R., Digital Systems: Hardware Organization and Design, Wiley, New York, 1973.

Huey, B. M., (unpublished) "Memorandum on PROBE Subroutine," Department of Electrical Engineering, University of Arizona, 1973.

Ng, W. W., "Evaluation of a LSI Fault Detection Program Using A Four Bit Microprocessor Circuit," M.S. thesis, Department of Electrical Engineering, University of Arizona, 1974.

Nilsson, N. J., Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, New York, 1971.

Roth, J. P., "Diagnosis of Automata Failures; a Calculus and Method," IBM Journal of Research and Development, vol. 10, pp. 278-291, 1966.

Van Helsland, M. C., "Evaluation of SCIRTSS Performance on Sequential Circuits Biased Against Random Sequences," M.S. thesis, Department of Electrical Engineering, University of Arizona, 1974.