

---

Doctoral Dissertations

Student Theses and Dissertations

---

1972

## Secondary techniques for increasing fault coverage of fault detection test sequences for asynchronous sequential networks

Lewis Ronald Hoover

Follow this and additional works at: [https://scholarsmine.mst.edu/doctoral\\_dissertations](https://scholarsmine.mst.edu/doctoral_dissertations)



Part of the [Electrical and Computer Engineering Commons](#)

Department: [Electrical and Computer Engineering](#)

---

### Recommended Citation

Hoover, Lewis Ronald, "Secondary techniques for increasing fault coverage of fault detection test sequences for asynchronous sequential networks" (1972). *Doctoral Dissertations*. 2078.  
[https://scholarsmine.mst.edu/doctoral\\_dissertations/2078](https://scholarsmine.mst.edu/doctoral_dissertations/2078)

This thesis is brought to you by Scholars' Mine, a service of the Missouri S&T Library and Learning Resources. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact [scholarsmine@mst.edu](mailto:scholarsmine@mst.edu).

SECONDARY TECHNIQUES FOR INCREASING FAULT COVERAGE  
OF FAULT DETECTION TEST SEQUENCES FOR  
ASYNCHRONOUS SEQUENTIAL NETWORKS

by

Lewis Ronald Hoover

A DISSERTATION

Presented to the Faculty of the Graduate School of the

UNIVERSITY OF MISSOURI - ROLLA

In Partial Fulfillment of the Requirements for the Degree

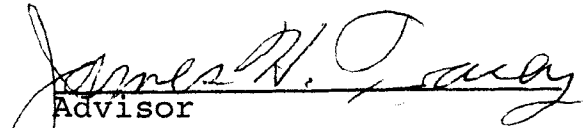
DOCTOR OF PHILOSOPHY


in

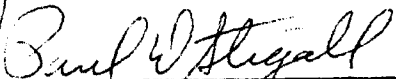
ELECTRICAL ENGINEERING

1972

T2744  
76 pages  
c. I

  
\_\_\_\_\_  
Advisor

  
\_\_\_\_\_

  
\_\_\_\_\_

  
\_\_\_\_\_

  
\_\_\_\_\_

\_\_\_\_\_

## ABSTRACT

The generation of fault detection sequences for asynchronous sequential networks is considered here. Several techniques exist for the generation of fault detection sequences on combinational and clocked sequential networks. Although these techniques provide closed solutions for combinational and clocked networks, they meet with much less success when used as strategies on asynchronous networks.

It is presently assumed that the general asynchronous problem defies closed solution. For this reason, a secondary procedure is presented here to facilitate increased fault coverage by a given fault detection test sequence. This procedure is successful on all types of logic networks but is, perhaps, most useful in the asynchronous case since this is the problem on which other techniques fail.

The secondary procedure has been designed to improve the fault coverage accomplished by any fault detection sequence regardless of the origin of the sequence. The increased coverage is accomplished by a minimum amount of additional internal hardware and/or a minimum of additional package outputs.

The procedure presented here will function as part of an overall digital fault detection system, which will be

composed of: 1) a compatible digital logic simulator, 2) a set of fault detection sequence generators, 3) secondary procedures for increasing fault coverage, 4) procedures to allow for diagnosis to a variable level.

This research is directed at presenting a complete solution to the problems involved with developing secondary procedures for increasing the fault coverage of fault detection sequences.

## ACKNOWLEDGEMENTS

I would like to express my appreciation to Dr. Tracey, not only for his helpful suggestions, constant supervision, and critical analysis of this work, but also for personal concern shown me during pursuit of my degree.

I would like to acknowledge Dr. David Rouse of Bell Telephone Laboratories, Columbus, Ohio, for supplying the TEGAS digital simulator and modifications to facilitate its use during this research.

Most of all, I wish to thank Bonnie for the understanding, encouragement and sacrifice, freely given, while obtaining my degrees.

Appreciation is also extended to the National Science Foundation for the support given my studies under a NSF Traineeship.

## TABLE OF CONTENTS

	Page
ABSTRACT.....	ii
ACKNOWLEDGEMENT.....	iv
LIST OF ILLUSTRATIONS.....	vii
LIST OF TABLES.....	viii
I. INTRODUCTION.....	1
A. Objectives of This Research.....	1
B. General Philosophies Governing Fault Detection Tests.....	3
C. Review of Existing Techniques for Fault Detection Test Generation.....	6
D. Detection Problems Unique to Asynchronous Networks.....	8
E. Design Considerations for Development of Secondary Techniques.....	12
II. SELECTIVE MONITORING OF SIGNAL LINES.....	16
A. Summary of Signal Line Monitoring Techniques.....	16
B. Solution by Cover Analysis.....	18
C. A Method Using Minimization of Additional External Contacts.....	25
D. Trade Offs Involved in Using These Two Methods.....	32
III. THE BACKWARD DRIVE METHOD FOR SETTING SIGNAL LINES.....	35
A. Summary of the Method.....	35
B. Theoretical Discussion of the Backward Drive.....	36

## TABLE OF CONTENTS (Continued)

	Page
IV. RESULTS AND CONCLUSIONS.....	45
A. Data Acquisition.....	45
B. Analysis of Data.....	48
C. Conclusion.....	54
V. APPENDIX - Sample Networks.....	57
VI. BIBLIOGRAPHY.....	65
VII. VITA.....	68

## LIST OF ILLUSTRATIONS

Figures	Page
1. Example Network.....	23
2. Networks Leading to Additional Outputs.....	31
3. General Space Domain Model.....	37
4. Singular Cover for an AND Gate.....	38
5. Space Domain Model of Figure 1.....	42

## APPENDIX

A.1. Sample Network A.....	58
A.2. Sample Network B.....	59
A.3. Sample Network C (Latched Adder).....	60
A.4. Sample Network D.....	61
A.5. Sample Network E (Master-Slave FF).....	62
A.6. Sample Network F.....	63



## LIST OF TABLES

Table		Page
1.	Faulty Machine List.....	22
2.	Simulator Output Table.....	24
3.	Cover Analysis.....	25
4.	Fault Coverage Table.....	30
5.	Singular Cover for Gate $b(k)$ .....	41
6.	Singular Covers for the Gates of Figure 5.....	43
7.	Intersection Table.....	43
8.	Results for the Sample Networks.....	52
9.	Sequence Modifications for Table 8.....	53

## APPENDIX

A.1.	Input Sequences for Sample Networks.....	64
------	--	----

## Chapter I

### Introduction

#### A. Objectives of This Research

This paper will first treat the general area of fault detection. Fault detection methods will be discussed for both combinational and sequential networks.

The following definition for a fault detection test (fdt) will be used throughout this paper:

An input sequence  $X$  (of length one or more) for a given network  $M$  is a fault detection test for fault  $f^i$  located in  $M$  if the output response to  $X$  for  $M$  with no faults, and the output response to  $X$  for  $M$  with  $f^i$  present, differ.

It can be seen that if  $M$  is a combinational network, the length of  $X$  for any  $f^i$  will be one; whereas, for an  $M$  of sequential structure, the length of the input sequence  $X$  may be greater than one.

As will be seen in section I.C., the problem of fault detection for combinational networks is solved by several methods<sup>1,2</sup>. Many of the same methods which experience great success with combinational networks are also very successful when dealing with synchronous networks<sup>3,4,5</sup>. This success can be accomplished readily when the synchronous network is considered in the space domain<sup>6</sup> (as compared to the time domain). In the space domain, the synchronous

network obeys all the restrictions placed upon a combinational network.

With the asynchronous problem, however, success is more limited. Although some methods attempt to use space domain analysis on asynchronous networks<sup>3,5</sup>, the results are less acceptable. The reason being that due to the inequality of total delays within closed paths of an asynchronous network, the space domain model fails.

It is the problem of fault detection test generation for asynchronous networks to which this research is addressed. Although most of the techniques developed herein are applicable to general networks, the major objective will be to make improvements upon the results which currently exist for the asynchronous case. When considering the sequential problem, the asynchronous case is of most interest since it is more general. Faults within a synchronous network may yield a network which does not obey the restrictions placed upon the general synchronous model.

Throughout this paper the abbreviation fdt will be used when referring to a fault detection test for a single fault; whereas, FDT will be used when referring to the collection of fdt's or sequence of fdt's which attempt to cover all faults in a network.

Further notation conventions, which will be followed where possible, will now be discussed. Lower case letters will be used when referring to elements of sets, vector

components, and signal lines. Upper case letters will be used to represent sets and vectors. Script notation, such as  $S$ ,  $M$  etc., will be used when referring to sequences of vectors, sets of sets and also when naming a general network.  $M$  will represent any arbitrary logic network in any of its faulty or fault free configurations.

Every attempt will be made to adhere to these conventions. As each new notation symbol is introduced, explanation of its function will be given.

#### B. General Philosophies Governing Fault Detection Tests

When considering the packaging techniques being used to produce MSI and LSI networks, it can be seen that the generation of an efficient FDT sequence for the network is an important production step when the reliability requirements on the packages are high. In addition to aiding in the detection of fabrication errors and burn-in faults, the FDT sequence designed for a packaged network will certainly become an important part of the much larger FDT system for the entire digital system.

It is critical that an efficient FDT sequence cover nearly all of the possible faults within a system. Another approach may be to require only that the FDT sequence cover the most probable faults. If near one hundred percent coverage is not easily attainable, then the selective coverage of the most probable faults is certainly a sound

approach. The assigning of meaningful probabilities of failure to all faults is an extremely difficult task. For this reason we will consider  $q/Q$  (where  $q$  = the number of single faults covered by an FDT sequence and  $Q$  = the total number of single faults contained in the network) as the evaluation factor for any FDT sequence.

The methods considered within this paper will be based upon the validity of the single fault assumption (sfa). This is to say that we will be applying the FDT sequence to the network with sufficient frequency so that the probability of the system containing two or more faults is very small. There is certainly good reason to believe that the single fault assumption is not valid when searching for fabrication errors in MSI and LSI packages or when detecting total systems structured from these packages. Fabrication errors caused by a bad layer within the chip or structural damage to the chip will cause the package to exhibit gross errors. It has long been accepted that FDTs designed under the sfa, which give good coverage of all single faults, will also detect the major portion of possible multiple fault patterns. A good FDT sequence would surely then detect multiple errors of the gross type discussed above.

When considering the design of FDT sequences, there are several cost factors that the engineer has available for trade off. Some of these are:

- 1) The cost of generating the FDT sequence
- 2) The cost of extra hardware within the package to facilitate easier generation and application of FDT sequence
- 3) The cost of each application of the FDT sequence
- 4) The cost of manual detection of uncovered faults

The cost incurred in design of the FDT sequence is a one-shot cost that can be quite high. The generation of the FDT is usually done by computer, and if the network has a large number of possible faults, the time required for FDT generation may be very high. Time may also be spent in minimizing the FDT sequence, and this procedure can become a significant portion of the design cost.

When the packaged elements are used in a total system design, costs which occur under 3) and 4) will require payment many times; thus, the cost of 1) may not seem so unbearable.

An alternate solution to this trade off problem is provided by 2). In present technology the cost of adding a few more elements within the package while keeping the number of external contacts (pins) nearly constant is a

small expense. With this idea in mind, it would be interesting to investigate techniques for adding internal elements which result in some of the following:

- 1) Easier generation of FDT sequences
- 2) Greater fault coverage for the FDT sequences
- 3) Shorter FDT sequences--thus less cost for each application
- 4) Small increase in the total cost of the packaged network

#### C. Review of Existing Techniques for Fault Detection Test Generation

The techniques employed for generating fdt's are greatly dependent upon the type of network under consideration. If the network is entirely combinational in structure, there exist algorithms for generating fdt's of length one for any fault within the network for which an fdt exists. Perhaps the most usable and useful algorithm for this purpose is the d-algorithm. For a particular faulty line, the d-algorithm sensitizes all possible paths between the fault and outputs so that the value on the faulty line alone will control the value of the output vector. The gate inputs which had to be fixed to provide for the path sensitization are then driven back to primary inputs to obtain the test input vector.

The d-algorithm suffers from one weakness which is characteristic of most all fdt generation methods-it cannot detect faults in redundant elements. Redundancy may exist in a network for one or more of the following reasons:

- 1) Complete functional minimization of the network not undertaken
- 2) Redundancy used as a design tool to control erratic behaviour (such as hazards in a sequential network)
- 3) To assure greater system reliability

The fact that faults cannot be detected in the redundant networks can, as Friedman<sup>7</sup> points out, be very serious. Unless the redundancy is of the Eichelberger type, a failure within the redundant portion may conceal an otherwise detectable fault in the nonredundant portion. With the exception of this one shortcoming, the d-algorithm has solved the problem of generation of fdt's for combinational networks.

The method of boolean difference<sup>2,8</sup> is also very successful when dealing with combinational networks. This method depends upon an equation solving technique. The function, realized by the fault free network, is compared with the function realized by the faulty network. This comparison is done by a pseudo-differentiation operation based upon the logical XOR operation.



There are numerous other methods for generation of fdts for combinational networks; however, the d-algorithm and boolean difference appear to be the most useful when working with networks which contain more than a few elements.

At present there is no acceptable algorithmic technique for generating fdts for all faults within general sequential networks. The methods of Kime<sup>9</sup> and Hennie<sup>10</sup> will generate fdts for sequential networks, but the length of the test sequence will in general be unacceptable. In addition to the production of unacceptably long sequences, these methods require the transition table for the network. These techniques are only applicable to machines which have distinguishing sequences. Kohavi and Lavelle<sup>11,12</sup> have demonstrated a method for imbedding a machine with no distinguishing sequence within a new machine which has a distinguishing sequence and thereby making the faults within the original machine detectable.

If the initial constraints are met, these methods are algorithmic; however, they require the transition table description of the network and result in unacceptably long FDT sequences.

#### D. Detection Problems Unique to Asynchronous Networks

As was mentioned in I.C., the problems which arise concerning the generation of fdts for sequential networks are

much more formidable than those encountered when working with combinational networks. Furthermore, the problems encountered when considering asynchronous networks are more formidable than those encountered with synchronous networks. The methods of Kime<sup>9</sup> and Hennie<sup>10</sup> provide a solution to the problem of FDT generation for synchronous sequential networks.

Ashkinazy<sup>13</sup> has presented a method, whereby, an asynchronous network can be represented by its Augmented Differential Equivalent. This allows the methods of Kime<sup>9</sup>, Hennie<sup>10</sup>, and Kohavi and Lavelle<sup>11,12</sup> to be applied to the asynchronous problem.

Although these methods offer a solution, it is generally considered to be unacceptable for several reasons. First of all, the FDT sequences resulting from these techniques are unacceptably lengthy and thus increase the cost of each FDT application. Furthermore, these techniques develop the FDT sequence based upon investigation of the transition table for the network. In general most networks are described in some way other than transition table form, thus, the production of the transition table is an additional design step. Since the transition table is not ordinarily required in the design process, it seems that obtaining a transition table to facilitate fault detection is an unacceptable requirement. The FDT sequences obtained in these methods are unacceptably lengthy, thus, the additional design step does not yield sufficient reward.

An approach to asynchronous sequential fault detection which does not require transition tables has been described by Seshu and Freeman<sup>14</sup>. Here the good and all faulty machines are simulated in parallel. At each step of the testing sequence all possible single input changes are simulated, and the one is selected which covers the most undetected faults. This technique is based upon local optimization and therefore guarantees no global optimization. Putzola and Roth<sup>3</sup> suggest that after initially detecting many faults, this method wanders aimlessly, providing no further fault coverage.

Hsiao and Chia<sup>4</sup> have proposed a modification of the combinational boolean difference technique for use in generating fdt's for asynchronous networks-the major problem here being that the method does not guarantee maximum fault coverage. The authors suggest that for most networks tested the fault coverage was 65 to 75 percent. This percentage does not appear to be high enough to be totally acceptable. Since the number of possible paths to outputs increase with each level of feedback, it appears that this method will be most successful when dealing with a limited number of feedback lines, all of which have shallow feedback: that is, no global feedbacks.

Putzola and Roth<sup>3</sup> have recently presented a method for asynchronous fdt generation based upon a modified d-algorithm. This technique functions by first breaking

the feedback lines of the sequential network and then cascading copies of this machine in a combinational fashion. Space domain analysis is being used in place of time domain analysis. The combinational d-algorithm is then used to generate a test for a particular fault in the machine. When the test is driven back to inputs, the result is an input vector at each spacial copy of the machine. The sequence of these input vectors is the fdt sequence for the fault under consideration. Due to the unequal total delays within closed paths of an asynchronous network, the space domain model is not an accurate model for the asynchronous case. For this reason, this method must be considered heuristic and feedback loops must then be closed and the fdt sequence simulated to see if the test does indeed detect the fault under consideration. Since the path which is sensitized is not necessarily the same path through all copies of the machine, and since the space domain model is not accurate, it cannot be assumed that if a given fdt for a particular fault is successful, all faults along the sensitized path are covered by this fdt. In contrast, this assumption is guaranteed in the combinational case.

It is found that this method also results in 65 to 75 percent coverage of faults. If the sequential network being considered is at all complex, cascading copies of this network will result in a very complex combinational

network and will thus require a great amount of time for each fdt generation. An FDT sequence, generated in this fashion to cover all faults in a given network, could become very lengthy.

When considering those methods for generation of fdt sequences which do not need a transition table, it is apparent that the best that can be expected is 65 to 75 percent fault coverage for a general network. Since redundancies are often used in asynchronous networks and, as was mentioned in I.C., detection of faults within these redundancies is important, it must be considered a shortcoming that none of these techniques can handle redundancies.

It then appears critical to consider some secondary techniques which would improve the percentage of fault coverage and also facilitate fault detection within the redundant elements.

#### E. Design Considerations for Development of Secondary Techniques

If secondary techniques are to be useful in conjunction with the methods discussed in I.D., they must provide a significant increase in the total fault coverage realized by the resulting FDT sequence, without forcing a disproportionate increase in any of the cost areas associated with generation and application of the FDT sequence.

If secondary techniques can be developed, it would be essential that they work with the circuit description and information used by the primary technique and with data provided by the primary technique. All the techniques mentioned in I.D., which do not require transition tables, have as an integral part of their procedure, simulation of the generated FDT sequence on the good machine and all faulty machines. If the secondary techniques can be designed to use as input the circuit description and the output of the simulation provided by the primary technique, the additional cost caused by the secondary technique would be reduced. The output from the simulation may require modification; however, the actual simulation procedure would remain unchanged.

In light of the cost discussions presented in I.B., it may be cost effective to add some internal package elements to facilitate a more efficient and cheaper FDT sequence. If additional hardware is added, it must not significantly increase the cost of the package and it must also lend itself to fault detection. Faults within the added elements must be detectable without destroying the cost efficiency of the FDT sequence.

While the secondary techniques may be very tolerant of the addition of internal elements, the addition of external communication paths (pins) must be rigidly controlled. If the secondary technique results in a large

increase in the number of pins from the package, then the cost struggle has been lost. Therefore, if extra pins are required, these must be kept to a minimum. If the cost effectiveness of the package is determined by:

- 1) Cost to design and fabricate
- 2) Cost to utilize (wire in) the package
- 3) Cost to design the FDT sequence for the package
- 4) Cost of application of the FDT package
- 5) Cost to manually detect uncovered faults

the designer may find that a small increase in the cost of 1) and 2) may yield a greater savings in areas of 3), 4), and 5). This suggests that the addition of selected internal elements and a minimum of external pins will not necessarily cause large overall cost increase. It would be hoped that the additional elements and pins would result in an FDT sequence with much greater fault coverage.

It would also be very beneficial if the secondary techniques could be structured so as to cover faults within the redundant circuits without jeopardizing the other design objectives.

A good secondary technique would be one which greatly increases the coverage of an FDT sequence without a disproportionate increase in total package and FDT cost.

Since all existing techniques fall short of the total fault coverage goal for the asynchronous case, it will be the objective of this paper to present a secondary technique to increase the fault coverage.

Generally it is felt that the asynchronous case will not have a network independent closed solution as is provided by d-algorithm and boolean difference in the combinational case. For this reason, it seems critical to develop secondary techniques which can increase the fault coverage of FDTs for asynchronous networks.

Since the generation of FDT sequences for the asynchronous case requires a major effort, and since the resulting sequence does not usually give satisfactory fault coverage, perhaps a more acceptable solution would be to use, as an FDT for an asynchronous network, a sequence which merely exercises the machine through its stable states or along some other transition paths. This sequence could then be backed by a good secondary technique-the result being greater fault coverage, shorter FDT sequences, less FDT generation time, with a small added cost to the package being tested.



## Chapter II

### Selective Monitoring of Signal Lines

#### A. Summary of Signal Line Monitoring Techniques

In Chapter II, two techniques, which will facilitate coverage of faults that are undetectable by monitoring primary outputs under application of a given FDT sequence  $X$ , will be discussed.

Consider  $M$  to represent the set of machines which can result from a given asynchronous network  $M$  being subjected to any of its possible internal single logical faults. That is, if  $F = (f^1, f^2, \dots, f^n)$  is the set of all possible single logical faults of  $M$ , then  $M = (m^0, m^1, m^2, \dots, m^n)$  is the set which corresponds to the configurations of the network  $M$  in the presence of the elements of  $F$ . That is,  $\forall f^i \in F$  there exists a unique  $m^i \in M$ . The element  $m^0$  will be used to represent the network  $M$  in the fault free configuration.

It is assumed throughout this paper that an FDT sequence  $X$  is available for application to  $M$ . This FDT may have been generated by modified d-algorithm, boolean difference, or some other technique. However, since  $M$  may be asynchronous and observation is limited to primary outputs, in the general case  $X$  will not detect all of the single logical faults within  $M$ . Allow  $M_d$  ( $d$  for detected) to represent the set of machines such that  $\forall m^i \in M_d \quad Z^i \neq Z^0$

(where  $z^k$  represents the output sequence of  $m^k$  under the application of  $X$ ). A parallel definition exists for  $M_u$  (u for undetected). Thus, the application of  $X$  to  $M$  partitions  $M$  into two disjoint subsets,  $M_d$  and  $M_u$ . Since the mapping from the set  $(M - m^0)$  to the set  $F$  is one-to-one and onto, there exists a similar partitioning on  $F$ . That is,  $F_d$  will represent detected faults and  $F_u$  undetected faults. The sets  $M_u$  and  $F_u$  will be of concern here.

A method will first be discussed which performs a cover analysis on all lines within the network. The result of this analysis will be a set of signal lines,  $S_s$ , which can give maximum coverage to the faults of  $F_u$  under application of  $X$  to  $M$ . All signal lines are considered as possible outputs, and analysis is done to decide which set of signal lines can detect the most undetected faults under application of  $X$ . The method will lead to a set,  $S_s$ , which is minimal in number but not necessarily unique. The shortcoming of this method is that, in general,  $\forall s_k \in S_s$ , a new external contact must be added. Although a significant amount of external contact minimization can be achieved in conjunction with this method, it is found to be very network dependent, and in general, places no upper bound on the number of external contacts which must be added.

In the second method, the set of lines which must be monitored is considered to be the set of all lines which correspond to the faults of  $F_u$ . For example, if  $F_u$  has as

elements single faults  $[a(sa0), a(sal), c(sa0), e(sa0), q(sal)]$ , then the set of signal lines to be monitored will be  $S_g = (a, c, e, q)$ . Note: As will be seen later, line a must be considered in two different ways. The major advantage of this method is that the maximum number of new external contacts, which must be added to the network, is four (4). That is, a method is presented which allows all the elements of  $S_g$  to be tied to a minimum number of external pins. The disadvantage being that additional hardware is required internally to facilitate this minimization. Faults within this added hardware are also considered.

The trade offs between these two techniques are discussed in section II.D.

#### B. Solution by Cover Analysis

The cover approach to the problem of selective signal line monitoring will be considered in this section. As mentioned in section II.A., this technique places no absolute upper bound on the number of external contacts which must be added to the network.

Consider the set of all signal lines contained in  $M$  to be  $S = (s_1, s_2, s_3, \dots, s_m)$ .  $S$  contains all primary inputs, primary outputs, feedback lines, and all internal connection lines. For each  $s_i \in S$  two logical faults can be associated; that is,  $s_i(sa0)$  and  $s_i(sal)$ . The total number of faults can be collapsed across each network element,

but since this in no way influences the method of solutions, it will be ignored. For each  $s_i \in S$  there exists  $f^i \in F$  and  $f^j \in F$  and  $m^i \in M$  and  $m^j \in M$ . It has been shown that observation of the output sequence  $Z = Z_1 Z_2 Z_3 \dots Z_w$  for the application of  $X = X_1 X_2 X_3 \dots X_w$  to  $M$  performed a partitioning of  $M$  and  $F$ . This partitioning can be applied to the set  $S$ . Consider the set  $S_u$  (undetected) to represent the set of signal lines such that  $\forall s_i \in S_u$  there exists at least one  $f^j \in F_u$  corresponding to a logical fault on  $s_i$ .  $S_d$  will be the subset such that  $\forall s_j \in S_d$  there exists exactly 2 faults,  $f^k$  and  $f^l$ ,  $\in F_d$  which are associated with faults on signal line  $s_j$ . It can be seen that  $S_d$  and  $S_u$  are disjoint although  $S_u$  may contain signal lines which have only one undetected fault associated with each line.

Since it is entirely possible that by monitoring a particular line, faults on other signal lines can be detected, all elements of the set  $S$  must be considered as candidates for monitoring.

The value on signal line  $s_i$  after the application of  $X_k$ , in the  $X$  sequence, to machine  $m^j$ , will be represented by  $v(i,j,k)$ . For the application of each input vector  $X_k$ , in the  $X$  sequence, first a comparison of  $v(i,0,k)$  with  $v(i,j,k)$  is made for all  $j$  to determine which elements of  $M$  can be detected by  $s_i$  under application of  $X_k$ . This must be done  $\forall s_i \in S$ . This entire process must then be repeated for  $X_{k+1}$ . This continues until the entire sequence  $X$  has

been applied. The result from this operation will be a set of fault coverage lists of the form  $s_i, X_k, m^p, m^1, \dots, m^r$ , where this list represents the fact that by observing line  $s_i$ , while  $X_k$ , in the  $X$  sequence, is applied to  $M$ , faulty machines  $m^p, m^1, \dots, m^r$  can be detected. It is upon these fault coverage lists that the cover analysis must be performed to determine which signal lines must be monitored.

The rules for performing the cover analysis will now be considered. All signal lines which are primary outputs are, by definition, going to be monitored. Consider the set of all primary output lines to be  $S_z, \forall s_i \in S_z \subset S$ ,  $s_i$  is a primary output of  $M$ . Thus, the removal of all  $s_i \in S_z$  before the analysis starts is necessary.  $\forall s_i \in S_z$ , there is associated a set of fault coverage lists of the form  $s_i, X_k, m^p, m^1, \dots, m^r$ . By combining all machines which are listed in the fault coverage lists for signal lines  $s_i$ , the set  $Mz_i$  is formed, where  $\forall m^j \in Mz_i, m^j$  can be detected by monitoring  $s_i$ . Similar sets  $Mz_k$  are formed  $\forall k$  such that  $s_k \in S_z$ . It can be seen that the set  $Md = U (Mz_i)$  for all  $i$  such that  $s_i \in S_z$  (where  $U$  is the set union operation). In a similar fashion, sets  $Ms_i$  for all  $i$ , such that,  $s_i \in (S - S_z)$  are formed. From each such set  $Ms_i$ , the elements which are common to  $Ms_i$  and  $Md$  are then removed. That is,  $Ms_i^* = Ms_i - (Ms_i \cap Md)$  is formed (where  $\cap$  is a set intersection operation). There now exists a set of the sets of form  $Ms_i^*$ , where  $\forall m^j \in Ms_i^*, m^j \in Mu$  and  $m^j$  can be detected by

monitoring  $s_i$ . To decide which signal lines of the set  $(S - S_z)$  must be monitored, first a search for critical signal lines is performed. That is,  $\forall m^i \in \text{Mu}$ , for which  $m^i$  is contained in one and only one  $\text{Ms}_j^*$ , monitoring of  $s_j$  is required. All machines which are covered by any such line  $s_j$  must now be removed from the  $\text{Ms}_k^*$  for all remaining lines in  $(S - S_z)$ . The cover analysis then proceeds using the following two rules:

- 1) The signal line with the highest value is the next line entered into the set  $S_s$ . The value for any line is equal to the number of previously undetected faults which are covered by monitoring this line.
- 2) If several lines have equal value, the choice will be arbitrary with the only priority being assigned to state variable lines.

The results of this analysis will be two sets of signal lines  $S_z$  and  $S_s$ , where  $\forall s_i \in S_z$ ,  $s_i$  is a primary output and  $\forall s_k \in S_s$ ,  $s_k$  is not a primary output. Thus,  $S_z \wedge S_s = \emptyset$  (where  $\emptyset$  represents the null set).

The members of  $S_s$  are the signal lines which will require additional primary outputs from the package to facilitate monitoring.

If  $M$  represents a general network, then  $\forall s_k \in S_s$ , it is necessary to add an additional primary output. As was mentioned earlier, some minimization can usually be accomplished; however, it is usually very network dependent. In

general, this method places no upperbound on the number of new primary outputs which must be added.

This method will be illustrated with the example shown in Figure 1.

Table 1 associates with each machine  $m^i$  of the above network a single logical fault.

$m^i$	Specific Fault
$m^1$	$x_1(sa1)$
$m^2$	$x_1(sa0)$
$m^3$	$x_2(sa1)$
$m^4$	$x_2(sa0)$
$m^5$	$x_3(sa1)$
$m^6$	$x_3(sa0)$
$m^7$	$a(sa0)$
$m^8$	$a(sa1)$
$m^9$	$b(sa0)$
$m^{10}$	$b(sa1)$
$m^{11}$	$c(sa0)$
$m^{12}$	$c(sa1)$

Table 1: Faulty Machine List

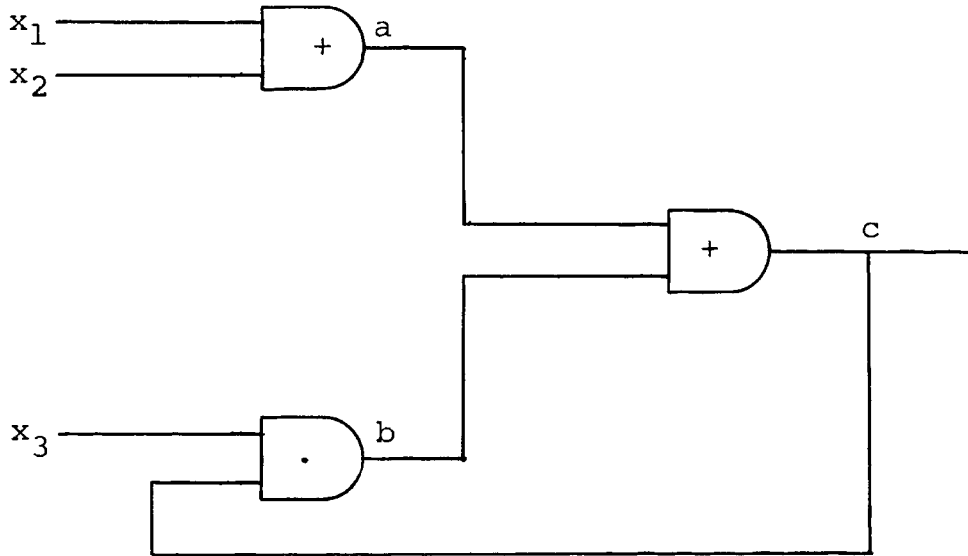


Figure 1: Example Network

For the input sequence  $X = X_1X_2X_3X_4 = (111)(101)(001)(011)$ , Table 2 shows the values of all signal lines of the network shown in Figure 1. The table includes data for the fault free and all single fault machines. Note: Line  $c = 1$  at start.

From Table 2, it can be seen that since

$$Md = [m^6, m^9, m^{11}] \text{ then}$$

$$Mx_1^* = [m^1, m^2]$$

$$Mx_2^* = [m^3, m^4]$$

$$Mx_3^* = \emptyset$$

$$Ma = [m^1, m^2, m^3, m^4, m^7, m^8]$$

$$Mb = \emptyset$$



Signal Lines	i=machine number ( $m^i$ )												Fault Coverage Lists		
	0	1	2	3	4	5	6	7	8	9	10	11		12	
$X_1$	$x_1$	1	1	0	1	1	1	1	1	1	1	1	1	1	$x_1, X_1, m^2$
	$x_2$	1	1	1	1	0	1	1	1	1	1	1	1	1	$x_2, X_1, m^4$
	$x_3$	1	1	1	1	1	1	0	1	1	1	1	1	1	$x_3, X_1, m^6$
	a	1	1	1	1	1	1	1	0	1	1	1	1	1	$a, X_1, m^7$
	b	1	1	1	1	1	1	0	1	1	0	1	0	1	$b, X_1, m^6, m^9, m^{11}$
	c	1	1	1	1	1	1	1	1	1	1	1	0	1	$c, X_1, m^{11}$
$X_2$	$x_1$	1	1	0	1	1	1	1	1	1	1	1	1	1	$x_1, X_2, m^2$
	$x_2$	0	0	0	1	0	0	0	0	0	0	0	0	0	$x_2, X_2, m^3$
	$x_3$	1	1	1	1	1	1	0	1	1	1	1	1	1	$x_3, X_2, m^6$
	a	1	1	0	1	1	1	1	0	1	1	1	1	1	$a, X_2, m^2, m^7$
	b	1	1	1	1	1	1	0	1	1	0	1	0	1	$b, X_2, m^6, m^9, m^{11}$
	c	1	1	1	1	1	1	1	1	1	1	1	0	1	$c, X_2, m^{11}$
$X_3$	$x_1$	0	1	0	0	0	0	0	0	0	0	0	0	0	$x_1, X_3, m^1$
	$x_2$	0	0	0	1	0	0	0	0	0	0	0	0	0	$x_2, X_3, m^3$
	$x_3$	1	1	1	1	1	1	0	1	1	1	1	1	1	$x_3, X_3, m^6$
	a	0	1	0	1	0	0	0	0	1	0	0	0	0	$a, X_3, m^1, m^3, m^8$
	b	1	1	1	1	1	1	0	1	1	0	1	0	1	$b, X_3, m^6, m^9, m^{11}$
	c	1	1	1	1	1	1	0	1	1	0	1	0	1	$c, X_3, m^6, m^9, m^{11}$
$X_4$	$x_1$	0	1	0	0	0	0	0	0	0	0	0	0	0	$x_1, X_4, m^1$
	$x_2$	1	1	1	1	0	1	1	1	1	1	1	1	1	$x_2, X_4, m^4$
	$x_3$	1	1	1	1	1	1	0	1	1	1	1	1	1	$x_3, X_4, m^6$
	a	1	1	1	1	0	1	1	0	1	1	1	1	1	$a, X_4, m^4, m^7$
	b	1	1	1	1	1	1	0	1	1	0	1	0	1	$b, X_4, m^6, m^9, m^{11}$
	c	1	1	1	1	1	1	1	1	1	1	1	0	1	$c, X_4, m^{11}$

Table 2: Simulator Output Table

The cover analysis is shown in Table 3.

		Elements of Mu									
		$m^1$	$m^2$	$m^3$	$m^4$	$m^5$	$m^7$	$m^8$	$m^{10}$	$m^{12}$	
Signal	$x_1$	x	x								
Lines	$x_2$			x	x						
	a	x	x	x	x			x	x		

Table 3: Cover Analysis

From Table 3, it can be seen that by monitoring signal line a, all faults coverable by this method are detected. By monitoring line a along with the primary output c, all faults except  $m^5$ ,  $m^{10}$ , and  $m^{12}$  can be detected.

### C. A Method Using Minimization of Additional External Contacts

In this section a method will be described which will allow for selective monitoring of signal lines while minimizing the number of additional external contacts required.

The set  $M$  will again be partitioned into  $M_d$  and  $M_u$  by the application of  $X$  to  $M$ . The elements of each  $M_u$  and  $F_u$  are then further partitioned into two disjoint subsets,  $F_{u_0}$ ,  $M_{u_0}$  and  $F_{u_1}$  and  $M_{u_1}$ , where  $\forall m^i \in M_{u_0}$ , the  $f^i \in F_{u_0}$  is a  $sa_0$  type logical fault, and  $\forall m^j \in M_{u_1}$  the  $f^j \in F_{u_1}$  is a  $sa_1$  type logical fault.  $\forall$  fault  $f^i \in F_{u_0}$  there is an associated signal line  $s_k$ .  $S_0$  will be the set of signal lines associated with the faults of  $F_{u_0}$  and similarly  $S_1$  and  $F_{u_1}$ .

Since, in general, we may have both logical faults  $f^i$  and  $f^j$  associated with a given line as elements of  $Fu$ , in general,  $S_1 \wedge S_0 \neq \emptyset$ . The signal lines  $s_i$ , such that,  $s_i \in (S_1 \cup S_0)$  are the lines which must be monitored. However, if under the input vector  $X_k$  from  $X$ , the signal line  $s_i$  (where  $s_i \in S_0$ ) = 1 in  $m^0$ , then  $s_i$  can be monitored to detect  $f^i$  (where  $f^i \in Fu_0$  is one of the faults associated with  $s_i$ ) during  $X_k$ . Since there may be many such  $s_i$ 's for a given  $X_k$ , there will be associated with each input vector two sets of signal lines,  $SX_k(0)$  and  $SX_k(1)$  where  $\forall s_i \in SX_k(0)$  the fault  $f^i$  (where  $f^i \in Fu_0$  is a fault associated with  $s_i$ ) can be detected by monitoring  $s_i$  during  $X_k$ . Likewise,  $\forall s_j \in SX_k(1)$ , the fault  $f^j$  (where  $f^j \in Fu_1$  is one of the faults associated with line  $s_j$ ) can be detected by monitoring line  $s_j$  during  $X_k$ . After the entire sequence has been applied to  $M$  and all of the sets of the type  $SX_k(\alpha)$  have been formed, a set  $S(0) = (SX_k(0), SX_{k+j}(0), \dots)$  is formed.  $S(0)$  is formed by including sufficient elements  $SX_k(0)$  so that  $\forall s_i \in S_0$ , for which there exists at least one  $SX_k(0)$  such that  $s_i \in SX_k(0)$ , there exists at least one  $SX_k(0) \in S(0)$ . Thus,  $f^i \in Fu_0$  can be detected by monitoring  $s_i$  during  $X_k$ . Similarly  $S(1) = (SX_1(1), SX_{1+r}(1), \dots)$ .

The following notation is now defined. If we have a set  $R = (r_1, r_2, r_3, \dots, r_k)$ , then  $\Pi(R) = \Pi(r_1, r_2, \dots, r_k) = (r_1 \cdot r_2 \cdot r_3 \cdot \dots \cdot r_k)$ , where  $(\cdot)$  represents the logical

AND operation. Similarly,  $\sum(R) = \sum(r_1, r_2, \dots, r_k) = (r_1 + r_2 + r_3 + \dots + r_k)$  where (+) is the logical OR operation.

Utilizing the above notation, the functions

$$\begin{aligned}\phi(0) &= \sum_{S(0)} [\prod(SX_i(0), I_0)] \\ \phi(1) &= \prod_{S(1)} [\sum(SX_j(1), I_1)] \quad \text{are formed.}\end{aligned}$$

The I signals are conditioning signals which will be defined later. The  $\phi$ 's express the logic function which must be realized on the additional network outputs so as to cover the faults of  $F_u$  which are detectable by this method.

In realizing  $\phi(0)$ , it can be seen that each element of  $S(0)$  will define the input list to an AND gate. That is,  $\forall SX_k(0) \in S(0)$  there will be defined an AND gate  $AX_k(0)$ . Each such  $AX_k(0)$  will have as inputs all elements of the set  $SX_k(0)$  plus an additional conditioning signal  $I_0$ . The outputs of all such  $AX_k(0)$  gates will completely define the input set for an OR gate  $\phi(0)$ . The output of  $\phi(0)$  will represent one of the additional required primary outputs.

Note: This discussion has been based, for simplicity, upon two level AND-OR logic. Certainly, the type logic elements actually utilized and the method of interconnection is unrestricted so long as the function realized is unaltered.

A similar two level OR-AND structure can be described for the  $\phi(1)$  function. Due to the parallelism between these

two functions, the verbal description of  $\phi(1)$  is omitted.

The  $I_0$  and  $I_1$  signal lines are used to facilitate fault detection of the added hardware.  $I_0 = 1$  during the application of every  $X_k$  to  $M$ , for which  $SX_k(0) \in S(0)$ .  $I_1 = 0$  during the application of every  $X_k$  to  $M$ , for which  $SX_k(1) \in S(1)$ . It must be mentioned that if the network is such that every  $X_i$  of  $X$  has associated with it an  $SX_i(\alpha) \in S(\alpha)$  (for  $\alpha = 0$  or  $\alpha = 1$ ), then an additional input vector must be added to  $X$  to facilitate the detection of the gates in the  $\phi(\alpha)$  network. That is, if line  $I_\alpha$  must be used to condition the gates of network  $\phi(\alpha)$  during the entire  $X$  sequence, then an additional input vector must be added to  $X$  so that  $I_\alpha$  can be used to detect faults in the  $\phi(\alpha)$  network.

From the above discussion it can be seen that if the machine is fault free, then  $\phi(0) = 1 \forall X_i$  for which there exists an  $SX_i(0) \in S(0)$ . Likewise, if we have the fault  $f^j \in Fu_0$  on the signal line  $s_i \in S_0$ , then  $\phi(0) = 0$  for all  $X_k$ , such that  $s_i \in SX_k(0)$ .

A sa0 fault on the output of gate  $AX_k(0)$  of the  $\phi(0)$  network will result in  $\phi(0) = 0$  during  $X_k$ . Also,  $\phi(0)$  sa0 will be detected by  $\phi(0) = 0$  during an  $X_k$  for which  $SX_k(0) \in S(0)$ . If there exists an  $X_r$  such that  $SX_r(0) \notin S(0)$ , then setting  $I_0 = 0$  during  $X_r$  yields  $\phi(0) = 0$  for  $m^0$ ; but  $\phi(0)$  will equal 1 if any gate in the  $\phi(0)$  network is sal.

A similar argument can be given for the output values and the faults within the  $\phi(1)$  network.

The procedure for realizing the  $\Phi(\alpha)$  function will be demonstrated by Figure 2.

Referring to the network of Figure 1, the following sets are enumerated to further clarify the theoretical discussion.

$$M_d = [m^6, m^9, m^{11}]$$

$$M_u = [m^1, m^2, m^3, m^4, m^5, m^7, m^8, m^{10}, m^{12}]$$

$$M_{u_0} = [m^2, m^4, m^7]$$

$$M_{u_1} = [m^1, m^3, m^5, m^8, m^{10}, m^{12}]$$

$$F_{u_0} = [x_1(sa_0), x_2(sa_0), a(sa_0)]$$

$$F_{u_1} = [x_1(sal), x_2(sal), x_3(sal), a(sal), b(sal), c(sal)]$$

$$S_0 = [x_1, x_2, a]$$

$$S_1 = [x_1, x_2, x_3, a, b, c]$$

From Table 4, it can be seen that:

$$S(0) = (SX_1(0)) \text{ or}$$

$$S(0) = (SX_2(0), SX_4(0))$$

$$S(1) = (SX_3(1))$$

The networks which lead to outputs  $\emptyset(0)$  and  $\emptyset(1)$  are shown in Figure 2.  $S(0) = [SX_2(0), SX_4(0)]$  is used to give an example of a two level result.

Signal Lines	$S_0$				$S_1$					
	$m^0$	$x_1$	$x_2$	$a$	$x_1$	$x_2$	$x_3$	$a$	$b$	$c$
$X_1 =$ (111)	$x_1$	1	x							
	$x_2$	1		x						
	$x_3$	1								
	$a$	1						x		
	$b$	1								
	$c$	1								
$X_2 =$ (101)	$x_1$	1	x							
	$x_2$	0				x				
	$x_3$	1								
	$a$	1						x		
	$b$	1								
	$c$	1								
$X_3 =$ (001)	$x_1$	0				x				
	$x_2$	0					x			
	$x_3$	1								
	$a$	0								x
	$b$	1								
	$c$	1								
$X_4 =$ (011)	$x_1$	0				x				
	$x_2$	1								
	$x_3$	1								
	$a$	1								
	$b$	1								
	$c$	1								

$SX_1(0) = [x_1, x_2, a]$   
 $SX_1(1) = \emptyset$

$SX_2(0) = [x_1, a]$   
 $SX_2(1) = [x_2]$

$SX_3(0) = \emptyset$   
 $SX_3(1) = (x_1, x_2, a)$

$SX_4(0) = (x_2)$   
 $SX_4(1) = (x_1)$

Table 4: Fault Coverage Table

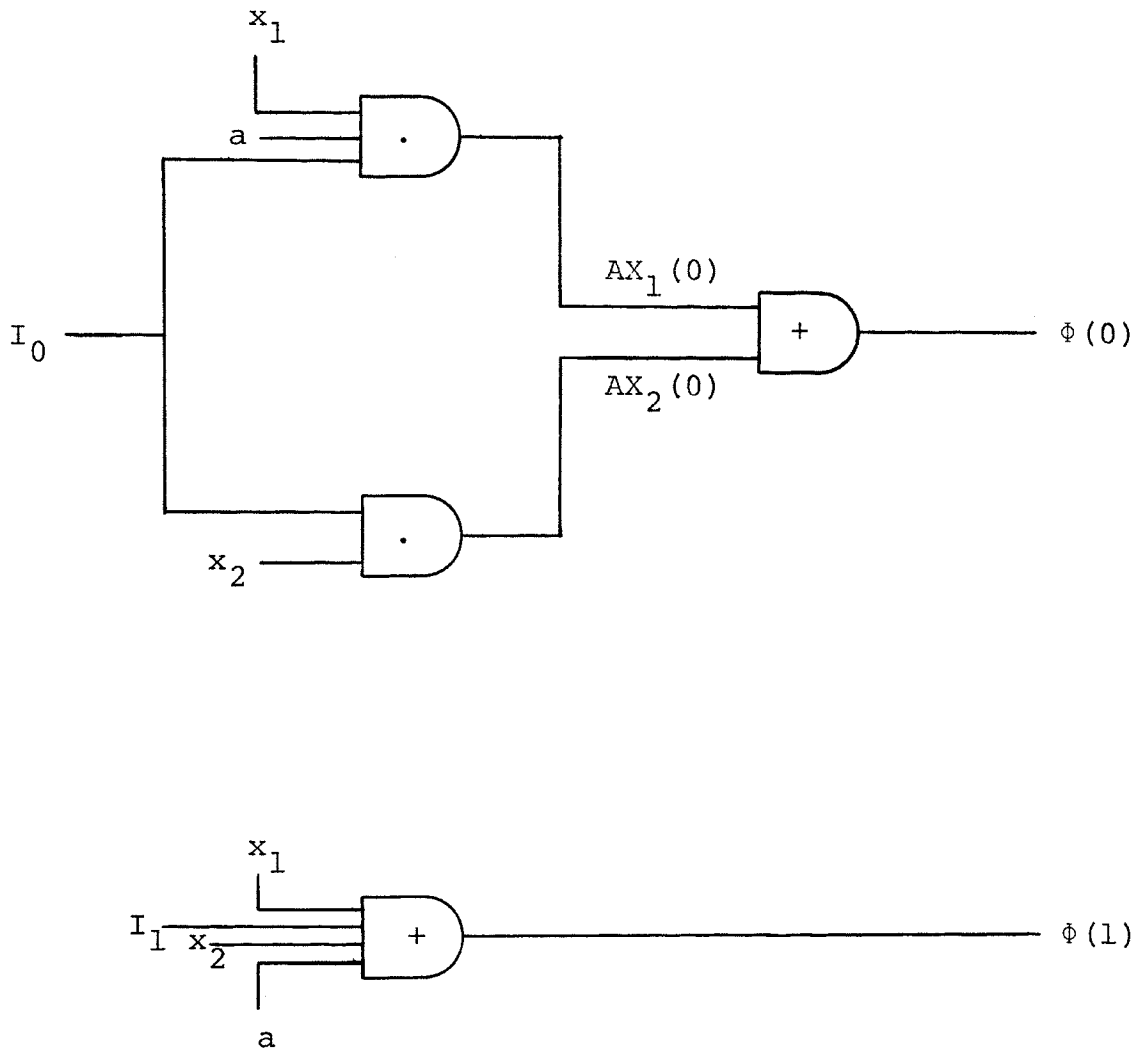


Figure 2: Networks Leading to Additional Outputs

Considering the example above, assume that while  $I_0 = 1$ ,  $X$  is applied to  $m^0$ . With  $X = X_1$  or  $X_2$ , then  $\phi(0) = 1$ . However, if any of the signal lines which constitute the sets contained in  $S(0)$  are  $sa0$ , then during either  $X_1$  or  $X_2$   $\phi(0)$  will equal 0. To check for any gate within the  $\phi(0)$  network  $sa1$ ,  $I_0$  is set to 0 and this should yield  $\phi(0) = 0$ . Any gate contained in the  $\phi(0)$  network which is  $sa0$  will cause  $\phi(0) = 0$  during  $X_1$  and/or  $X_2$ .



Assume that while  $I_1 = 0$ ,  $X$  is applied to  $m^0$ . With  $X = X_3$ ,  $\phi(1)$  should equal 0. If any of the signal lines which constitute the sets contained in  $S(1)$  are sal,  $\phi(1)$  will equal 1. With  $I_1 = 1$ ,  $\phi(1)$  should equal 1 for all  $X$ . However, if any gate within the  $\phi(1)$  network is sa0,  $\phi(1)$  will equal 0 for  $I_1 = 1$ . With  $I_1 = 0$ , any gate within the  $\phi(1)$  network which is sal will yield  $\phi(1) = 1$  during some  $X_k$  for which  $\phi(1)$  should = 0 under fault free conditions.

#### D. Trade Offs Involved in Using These Two Methods

Both methods presented above are attempts to yield increased fault coverage for any general asynchronous network  $M$ . These methods will also be effective on combinational or synchronous sequential networks for which the associated  $X$  does not give total coverage. It will be the purpose of this section to discuss the relative value of these methods.

The obvious trade offs are very evident. The method of section II. B., to be referred to as method 1, required no additional hardware elements within the package. There are networks where the maximum number of new outputs, which must be added for method 1, will be less than the four contacts which method 2 (the method of section II.C.) yields as an upper bound. Certainly, on networks of this type, method 1 should be used. Since no upper bound can be placed on the number of new contacts required by method 1, it is felt that a technique similar to method 2 is critical. It

seems necessary to establish an upper bound on new contacts required from any package since the actual interconnection of external contacts is a major portion of the cost. In light of present fabrication techniques, the costs associated with addition of internal package elements is very minor. Most networks, which have been analyzed by method 2, have required a relatively small percentage of additional internal elements. For these reasons, method 2 appears to present the most satisfactory solution to the general problem.

Since method 1 and method 2 require the same type of data for analysis, it seems likely that an attempted solution by method 1, which does not yield success, could be followed by method 2 without an additional major analysis cost.

It would certainly be hard to argue that a method 2 solution with four new network contacts and additional internal hardware is better than a method 1 solution which requires five new network contacts. The decision mechanisms would have to consider such variables as: cost of additional internal elements, cost to "wire in" each new network contact, cost of analysis of test sequence outputs. On some problems, it has proven beneficial to monitor several lines by method 1 and then switch to method 2 to guarantee a realistic upper bound on the number of new contacts required.

It should be mentioned that either method allows faults within redundant logic sections to be detected. Inability

to detect faults in the redundancies is one of the major shortcomings of the most popular FDT sequence generation schemes for all networks - combinational and sequential.

One trade off for method 2 can yield an upper bound on the number of new network contacts of three. This can be done by eliminating one of the I lines. If the sequence  $X$  is applied twice, a single I line can serve as  $I_0$  during the first application of  $X$  and serve as  $I_1$  during the second application. During the first application, output  $\phi(0)$  would be observed; while during the second application, attention would be on  $\phi(1)$ . The only restriction, which must be met, is that all state requirements, which must be fulfilled by  $M$  before  $X$  is applied, must also be satisfied before the second application. The implications of this restriction are outside the scope of this paper.

Since it has been assumed that the generation of the  $X$  sequence was accomplished by one of the many existing methods, it can be seen that given an FDT sequence  $X$  for  $M$  the percentage of faults covered by  $X$  can be increased by selective signal line monitoring. Since for the asynchronous case, percentage of fault coverage has been generally much less than one hundred, it appears that the additional costs involved in adding a minimum of new package contacts is a cost which might be willingly paid.

## Chapter III

## The Backward Drive Method for Setting Signal Lines

## A. Summary of the Method

It was shown in section II.A., that when an FDT sequence  $X$  was applied to  $M$ , observation of the output sequence performed a partitioning of  $M$  into two disjoint subsets,  $M_d$  and  $M_u$ , where  $\forall m^i \in M_d, z^i \neq z^0$  (where  $z^k$  is the output sequence for  $m^k$  under the application of  $X$ ) and  $\forall m^j \in M_u, z^j = z^0$ .

By selective monitoring of various state variable and internal signal lines, further partitioning of  $M_u$  into two disjoint subsets,  $M_{u_d}$  and  $M_{u_u}$  was accomplished. A similar partition exists on  $F_u$ ; that is,  $\forall m^i \in M_{u_d}$ , then  $f^i \in F_{u_d}$  and  $\forall m^j \in M_{u_u}$ , then  $f^j \in F_{u_u}$ . If the external contacts, which have been added to facilitate this partition are considered to be the  $r$  components of an output vector  $P$ , then for the application of  $X$  on  $M$  the results are:

- 1)  $\forall m^i \in M_{u_d}, \rho^i \neq \rho^0$  (where  $\rho^k$  is the output sequence of  $P$  vectors from  $m^k$  under application of  $X$ ).
- 2)  $\forall m^j \in M_{u_u}, \rho^j = \rho^0$ .

Application of the FDT sequence  $X$  to  $M$  has been successful in detecting all single faults except those which result in the set  $M_{u_u}$ . Since these faults could not be detected by direct monitoring of the signal line,

it is apparent that under the application of  $X$  to  $M$ , the signal line associated with fault  $f^i$ ,  $\forall f^i \in Fu_u$ , did not assume the proper value to allow for detection of  $f^i$ . As an example, to facilitate detection of the fault, line a (sal), the FDT sequence must force line a in  $m^0$  to assume the value 0 at least once. The problem is to develop a heuristic which will allow modification of  $X$  so as to enable detection of the faults  $f^i \in Fu_u$ . The heuristic technique presented here borrows on the theory which has developed around the use of Roth's d-algorithm<sup>1</sup>. A similarity will be seen between this method and the consistency test or backward drive segment of the d-algorithm.

#### B. Theoretical Discussion of the Backward Drive

Following Breuer<sup>6</sup> it is suggested that the time domain analysis of the system  $M$  be mapped into its corresponding spacial equivalent. This mapping can be accomplished if, for each new input vector, a new copy of  $M$  is allowed. Since it is the goal to force a given value on a particular line in  $m^0$ , the multiple copies of  $m^0$  will be labeled  $C(k)$ ,  $C^0(k-1)$ , ---  $C^0(k-L+1)$ . The length  $L$  of the new sequence  $X m^i$  is generated in this manner can be dynamically determined within reasonable restraints. The space domain analysis can be understood by observing Figure 3.

The copies of the machine are interconnected in such a way that in addition to the original input vector,  $C^0(k-d)$  has as inputs on its  $Y_{(k-d)}$  lines the state variable vector

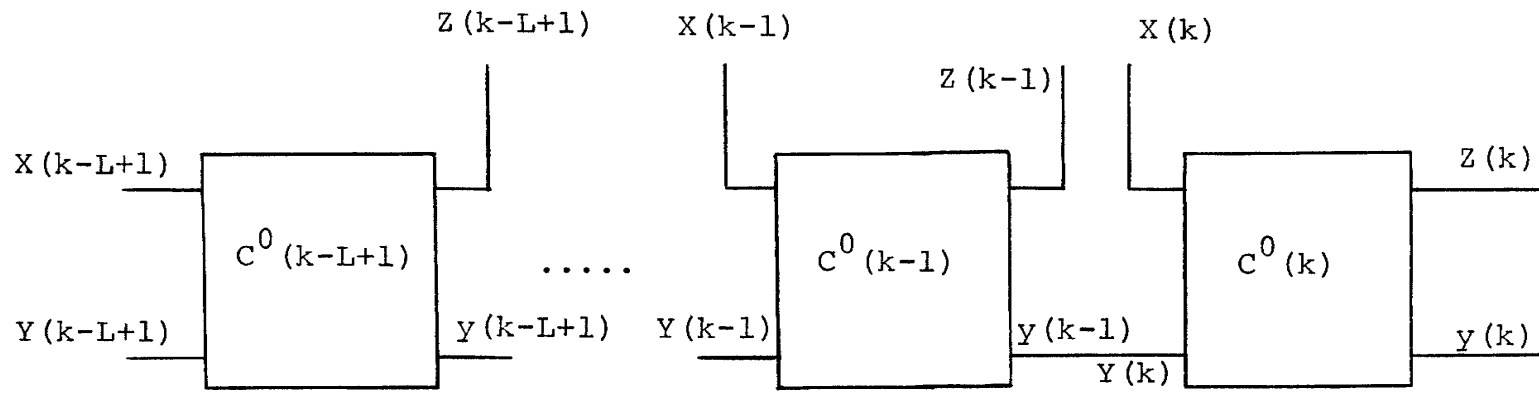


Figure 3: General Space Domain Model

$Y_{(k-d-1)}$  from copy  $C^0(k-d-1)$ .

Assume that it is necessary to generate an input sequence  $X m^i$  of length  $L$  to aid in detecting  $f^i \in Fu_u$ , a sal fault on line  $\underline{a}$ . First, assign line  $\underline{a}$  in  $C^0(k)$  the value 0 and attempt to drive this signal from  $C^0(k)$  back through all copies to  $C^0(k-L+1)$ .

The method for accomplishing the backward drive will now be discussed. For all gates along the signal paths which control line  $\underline{a}$  of  $C^0(k)$ , the singular covers<sup>15</sup> must be formed. An example of the singular cover for a 3 input AND gate is given in Figure 4.

The singular cover for  $C^0(k)$  is formed between inputs and signal line  $\underline{a}$ . The required value on line  $\underline{a}$  is then driven backward to the inputs of  $C^0(k)$  by performing intersections on the singular covers of the gates along the path. All parallel paths must be intersected simultaneously. However, intersections need not be made with singular cover vectors for gates whose outputs are unrestricted. The rules for intersection are:

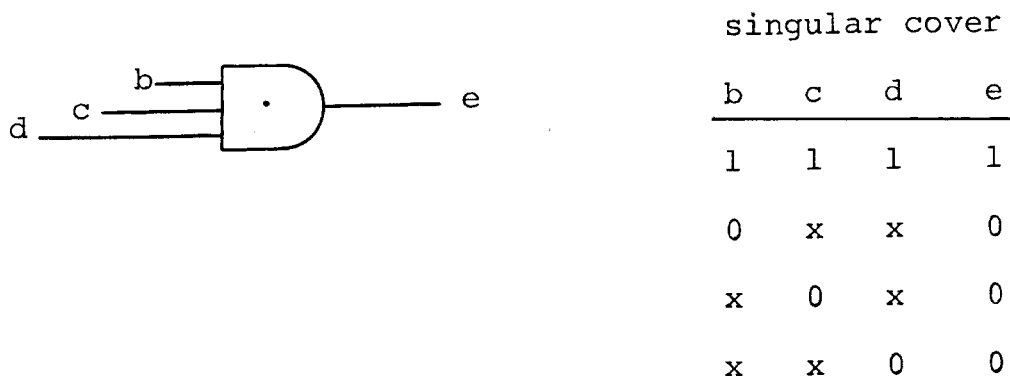


Figure 4: Singular Cover for an AND Gate

$$\begin{aligned}
 1 \wedge 0 &= \emptyset & x \wedge 0 &= 0 = 0 \wedge x \\
 0 \wedge 1 &= \emptyset & x \wedge 1 &= 1 = 1 \wedge x
 \end{aligned}$$

If at any time during the backward drive a  $\emptyset$  results, then an inconsistency exists and a retrace is required beginning with a new vector from the appropriate singular cover.

If  $M$  is asynchronous, care must be taken when picking vectors from the singular cover for intersection. It must be assured that  $D[X(k-r) - X(k-r+1)] \leq 1$  (Where  $D$  is the Hamming inter-vector distance). As an example, if  $X(k-2) = [0xx1]$  and  $X(k-1) = [01x1]$ ,  $D = 1$ . This, however, may force the reevaluation of  $D[X(k-1) - X(k)]$ .

When the backward drive to the inputs of  $C_0(k)$  is completed, the values required on the input vectors  $X(k)$ , and  $Y(k)$  must be investigated. If the state variable vector  $Y(k)$  which is being input from the  $C^0(k-1)$  copy is  $Y(k) = [xxx\dots x]$  (unrestricted), then the result is a sequence  $X m^i$  of length  $L = 1$ . However, if  $Y(k) \neq [xxxx\dots x]$ , the backdrive must continue through  $C^0(k-1)$ . This procedure continues until at some level  $(k-L+1)$ ,  $Y(k-L+1) = [xxx\dots x]$ . This strategy is required so that the sequence which is generated is not state dependent. Therefore, the sequence  $X m^i$  is forced to produce the desired result on line a regardless of the state of  $M$  when  $X m^i$  is applied. If, due to network configuration, information concerning machine state is known, this requirement can be appropriately relaxed. If at the  $(k-r)$  level the condition  $Y(k-r) = [xx\dots x]$  is not satisfied, the procedure must continue



to the  $(k-r-1)$  level. However, this process must not be allowed to continue indefinitely. One criteria for stopping the process short of success would be to determine some cost effective constant  $R$  and require that  $L \leq R+1$ .

If this technique yields a sequence  $X m^i$  and if  $M$  is synchronous,  $X m^i$  is certain to assign the proper value to line a; that is, if  $X m^i = X(k-L+1), X(k-L+2), \dots, X(k-1), X(k)$  is applied to  $m^0$  beginning at time  $t = t_0$ , line a will assume the desired value at  $t = t_0 + L$  (with  $L$  assigned time units). If  $M$  is asynchronous, the space domain model fails; thus, the technique is heuristic, and  $X m^i$  must be simulated to check on its validity. In either case, if  $X m^i$  is valid, the new FDT which covers the set of faults,  $(f^i + F - Fu_u)$  is  $XX m^i$ . If there are other faults,  $f^j \in Fu_u$ , which are not covered by  $XX m^i$  then this procedure would be repeated for  $f^j$ . There is no guarantee that the  $X m^i$  found in this manner is optimal. The length of  $X m^i$  is dependent upon the choice of vectors from the singular covers.

After all sequence modifications of the form  $X m^j$  have been produced, the total modifications are then simulated with  $X$ , to determine their success. If the  $X m^j$ 's are successful these results must be combined with either method 1 or method 2.

To illustrate this method, an example follows based upon the network of Figure 1.

Assume that signal line  $\underline{b}$  (sa0) is a fault which has not been detected. It is necessary to force a logical 1 on  $\underline{b}$ . This procedure begins by turning to the space domain analysis and forming the singular cover of the network from line  $\underline{b}$  to the inputs of copy  $C^0(k)$ . The space domain model is shown in Figure 5.

$x_3(k)$	$c(k-1)$	$b(k)$	label
1	1	1	A
1	x	0	B
x	1	0	C

Table 5: Singular Cover for Gate  $b(k)$

Table 5 shows the singular cover vectors for  $b(k)$  in  $C^0(k)$ .

Since the feedback line  $c(k-1) \neq x$  when  $b(k) = 1$ , the process must proceed to the  $(k-1)$  level. Therefore,  $C^0(k-1)$  is added to Figure 5 and the singular covers listed in Table 6 are formed.

The singular cover vector A from  $b(k)$ , labeled  $A_{b(k)}$ , can be intersected with either A or B of the singular cover of  $c(k-1)$ . Since  $\underline{b}$  is the gate which is influenced directly by the feedback line, the intersection between  $A_{b(k)}$  and  $B_{c(k-1)}$  is performed. This intersection will place less restrictions on the feedback line which is input to gate  $b(k)$ . The results of the intersections are shown in Table 7.  $A^*$  need not be intersected with any of the

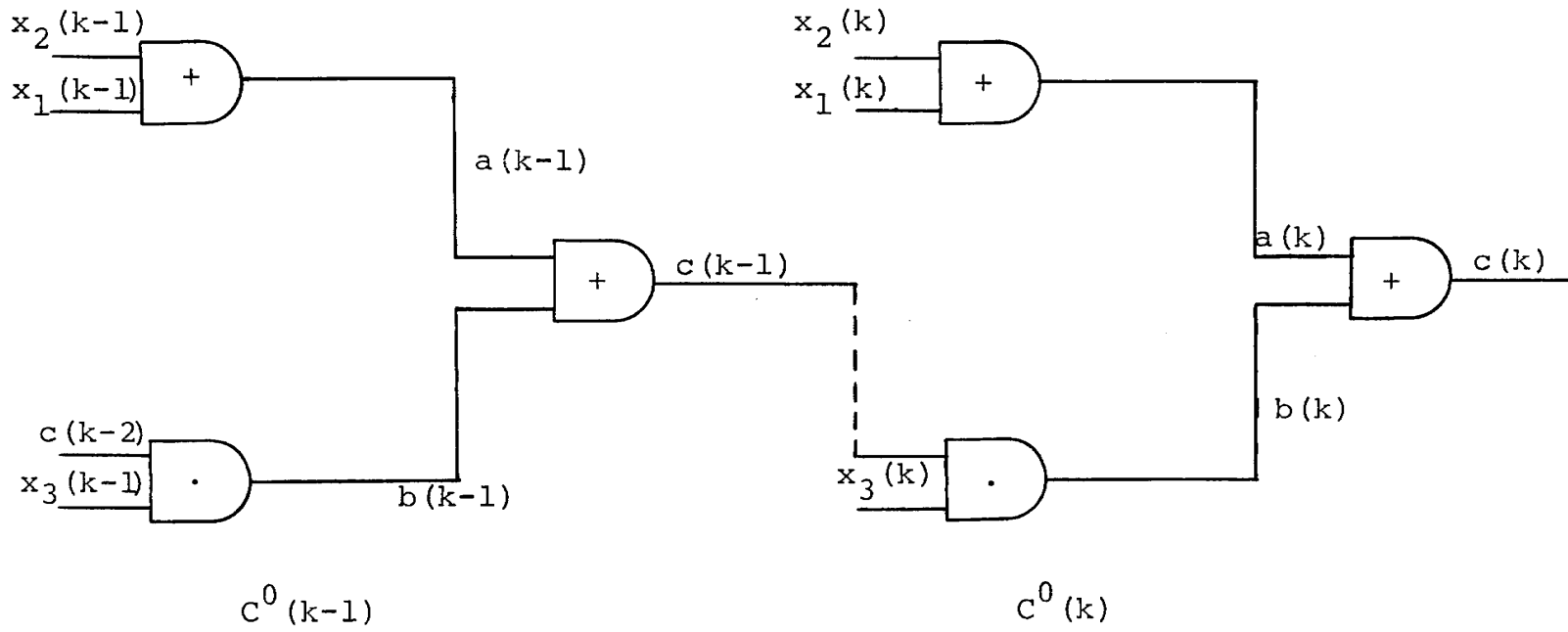


Figure 5: Space Domain Model of Figure 1

$c^0(k-1)$			$c^0(k)$				label	gate name
$x_1(k-1)$	$x_2(k-1)$	$c(k-2)$	$x_3(k-1)$	$a(k-1)$	$b(k-1)$	$c(k-1)$		
				x	1	1	A	c(k-1)
				1	x	1	B	
				0	0	0	C	
		1	1			1	A	b(k-1)
		0	x			0	B	
		x	0			0	C	
x	1			1			A	a(k-1)
1	x			1			B	
0	0			0			C	

Table 6: Singular Covers for the Gates of Figure 5

$x_1(k-1)$	$x_2(k-1)$	$c(k-1)$	$x_3(k-1)$	$a(k-1)$	$b(k-1)$	$c(k-1)$	$x_3(k)$	$b(k)$	label	description
				1	x	1	1	1	A*	$A_b(k) \wedge B_c(k-1)$
x	1			1	x	1	1	1	B*	$A^* \wedge A_a(k-1)$

Table 7: Intersection Table

singular covers of  $b(k-1)$  since  $b(k-1) = [x]$ .  $A^*$  is now intersected with either  $A_{a(k-1)}$  or  $B_{a(k-1)}$ . The result is shown for  $A_{a(k-1)}$ . This final vector has  $Y(k-1) = c(k-2) = [x]$ . Therefore, the procedure stops with  $L = 2$ . The  $X m^i$  sequence is  $X_1 X_2 = (x|x)(x|x)$ . It can be verified by hand simulation that this sequence does indeed force line b to have a value 1.

## Chapter IV

### Results and Conclusions

#### A. Data Acquisition

In order to do fault detection analysis on any network, it is necessary to simulate the behavior of the network in all of its faulty configurations under the application of  $X$ . This can be done most efficiently by utilizing a digital logic simulator with a parallel simulation feature. The TEGAS<sup>16</sup> simulator is such a system. This system simulates 32 different network configurations with each pass through the network.

The sample networks, which were used in collecting data, are shown in the appendix. Networks were chosen which exhibit features that generally complicate the problem of realizing total fault coverage.

Network A is an asynchronous sequential network which is highly redundant. By writing the output function for this network, it can be seen that the  $x_1$  input is unnecessary.

Network D is a well known<sup>17</sup> combinational network, which contains reconvergent fan-out. This network, with a large section of added redundancies, appears in Table 8 as network D'. D', in conjunction with network A, provides a good test for the ability of the secondary techniques to cover faults within redundancies.

Networks B, E, and F are asynchronous sequential networks. These networks all contain several feedback lines, and the feedbacks are to several levels within the networks. This type problem is the most difficult type asynchronous network to handle. The inequality of the total delays within closed feedback paths causes the analogy between time domain analysis and space domain analysis to break down. It was felt that these networks would provide the most serious challenge for the secondary techniques.

Network C is a synchronous sequential network. This network, along with network D, was included to demonstrate that the secondary techniques presented herein are applicable to all types of networks.

The input sequences which were applied to the sample networks are listed in Table A.1 of the appendix. Although several algorithms for generating FDT sequences were discussed in Chapter II, these methods were not applied here. To generate sequences by any of these methods would require a computerized implementation of the algorithm. Since this was not readily available, the input sequences were generated in other ways. If a state table for the network was available, one of the sequences was chosen to exercise the network through its stable states. Otherwise, the sequences are random sequences. In the asynchronous networks the sequences were designed so that only one input variable was changing at a time. The input vector was

applied as a constant input to the asynchronous networks until the network stabilized (fundamental mode). It would be interesting to observe the performance of the secondary techniques when working in conjunction with an FDT sequence of algorithmic origin. However, since no algorithmic technique for FDT generation can assure total fault coverage, the structure of the FDT merely governs the degree of dependence upon the secondary techniques.

The TEGAS simulator is now implemented on an IBM 360/50 system. The actual time required to do the simulation for the examples was very short. There was no test run which required more than 1 minute and 40 seconds of computer time. On most of the test runs,  $2/3$  of the actual computer time was spent preprocessing the data, while  $1/3$  was spent doing the actual simulation. This fraction is dependent upon the network and the length of the sequence being simulated. Assuming network structure independence, the actual time for simulation increases linearly with the number of network elements.

The simulator presents the network data in a form which is readily usable by the secondary techniques. The signal line values can be readily interrogated at any time to detect fault coverage. Although the actual data analysis for the secondary techniques was done manually, this process will be program implemented and interfaced with the simulator.



Based upon an analysis of the operations actually performed by the simulator and the operations required by the secondary techniques, it appears that an increase in simulation time of less than 30% would be required by the secondary techniques. This increase would represent the total cost associated with the secondary techniques since the preprocessing step would remain unchanged. It is believed that after the secondary techniques have been program implemented and interfaced with TEGAS, the total run times on networks similar to the ones tested will be in the neighborhood of 2 minutes. These techniques are not intended for application to logical networks of entire systems. Based upon the above run times, it can be seen that the computer cost associated with doing fault coverage analysis, including application of secondary techniques, would be very acceptable on modular networks.

#### B. Analysis of Data

The results presented in this section do not totally exhaust the data collected; however, they are considered to be a representative sample.

Table 8 is a compilation of the results obtained from analysis of the test run data. The Network-Sequence row label of Table 8 acts as a joint pointer to the network and the corresponding input sequence which gave rise to the data in the associated row of the table. This pointer can be followed to the figures and tables of the appendix to find

the circuit diagram and the corresponding input sequence.

The data columns within Table 8 are as follows:

- I) The total number of faults considered
- II) The number of faults which were detectable by monitoring primary outputs only
- III) The number of additional signal lines which require direct monitoring as suggested by method 1 (Section II.B)
- IV) Additional fault coverage yield by method 1
- V) Additional hardware required by method 2 (Section II.C)
- VI) Additional fault coverage yield by method 2
- VII) A pointer to Table 9 where the input sequences generated by the backward drive are listed
- VIII) Additional fault coverage yield by the backward drive technique
- IX) Total final fault coverage as a percentage of column I

The total number of faults considered for each network was the total of all possible single faults within the network after fault collapsing was performed across each gate. That is, for an  $n$  input AND gate,  $n+2$  single faults are considered (as opposed to  $2n+2$ ): each input ( $sa1$ ) and the output ( $sa1$ ) and ( $sa0$ ). A ( $sa0$ ) fault on an input of an AND gate is equivalent to the output ( $sa0$ ).

The data of columns III and IV is associated with the method of selective signal line monitoring presented in Section II.B. This method places no upper bound on the number of additional signal lines which must be directly monitored; however, if it results in 4 or fewer signal lines, it is to be preferred over the method of Section II.C, which places an upper bound of 4 on the number of additional network contacts which must be added, but requires additional internal hardware to assure this maximum.

Columns V and VI contain data associated with the application of method 2.

Although it is entirely possible that a combination of methods 1 and 2 could yield a joint solution on a particular network which would be more acceptable than the solution presented by either method independently, no example of this type was encountered while running the tests shown in Table 8.

Columns VII and VIII contain information associated with the backward drive technique presented in Chapter III. Column VII contains a pointer into Table 9. By following this pointer, the input sequence modifications, which were generated by the backward drive technique, can be found in Table 9. This sequence was concatenated with the X sequence from Table A.1 and resimulated to ascertain if it is successful in increasing fault coverage. The success or failure of this technique is reflected by value in column VIII.

Column IX lists the final fault coverage percentage. This is calculated by finding the total of either columns II, IV, and VIII or columns II, VI, and VIII and then comparing this with column I.

The (--) symbols within Table 8 indicate that for the test run under consideration, the method indicated by the (--) was not needed.

The backward drive technique was completely successful in all cases except example E-2. After the backward drive was applied, the resulting sequence modifications were simulated to determine the success of the modified sequence. In all cases, except E-2, the success was total. Network E has multiple feedback lines to varying levels. On a network of this structure, the space domain model for the time domain system is a poor model. In this situation, the model failed, and the modified sequence was unable to detect the remaining 3 faults.

On networks A and D', which are highly redundant networks, the secondary techniques handled the faults within redundancies with no problem and resulted in total fault coverage for all input sequences.

None of the example networks were exceedingly large. However, as will be mentioned in Section IV.C, it appears that the results on larger networks will be equally successful. In fact, it is expected that there will be improvement in the area of percentage of increased cost associated with

Network-Sequence	I	II	III	IV	V	VI	VII	VIII	IX
A-1	11	4	2	3	---	--	a	4	100%
A-2	11	10	1	1	---	--	---	--	100%
A-3	11	8	2	3	---	--	---	--	100%
B-1	24	7	4	13	---	--	b	4	100%
B-2	24	4	4	12	---	--	c	8	100%
B-3	24	9	4	13	---	--	d	2	100%
C-1	31	19	4	12	---	--	---	--	100%
C-2	31	16	4	12	---	--	e	3	100%
D-1	25	7	7	13	3 gates	13	f	5	100%
D-2	25	11	7	12	3 gates	12	g	2	100%
D-3	25	13	5	9	4 gates	9	h	3	100%
D'-1	29	2	8	25	6 gates	25	i	2	100%
D'-2	29	6	8	23	6 gates	23	---	--	100%
E-1	20	19	1	1	---	--	---	--	100%
E-2	20	6	4	11	---	--	j	0	85%
F-1	28	18	6	10	4 gates	10	---	--	100%
F-2	28	21	3	7	---	--	---	--	100%

Table 8: Results for the Sample Networks

a	b	c	d	e	f	g	h	i	j
x10	xxx0	1111	xxx0	0000	1111	110x	1x0x	x11x	0
1xx	xx10	1110	xx10		110x		1x10		1
	011x	1010	011x						
		1110							
		0110							

Table 9: Sequence Modifications for Table 8

the extra contacts and hardware required by methods 1 and 2. It is assumed that these secondary techniques would be applied at the packaged component level rather than at the total system level. For this reason, the size of the networks would be limited.

By observing Table 8, in conjunction with Table A.1, it can be seen that the success of fault detection is very sequence dependent. The dependency is upon both length of the sequence and the order of the input changes. The secondary techniques presented herein had a high degree of success, regardless of the input sequence. In most cases, when using these secondary techniques, the changing of an input sequence affects the cost of realizing total single fault coverage. In contrast, existing methods have the input sequence as the only variable which can be exercised to yield increased fault coverage.

### C. Conclusion

Using the results presented in Section IV.B as a basis upon which to draw conclusions, the secondary techniques presented herein are extremely successful. The goal of total single fault coverage was realized on every example except one. Several of the example networks had multiple feedbacks to various levels, and several contained redundant sections of logic which would ordinarily introduce many undetectable faults. On these examples, the secondary techniques were very successful in obtaining total single fault coverage.

The amount of extra hardware required by method 2 was relatively high. Method 2 never results in any fewer than 3 or 4 additional gates; studies indicate it also seldom requires more than 6 or 7 additional gates. With this in mind, it seems probable that on networks with large number of gates, the percentage of required additional gates will decrease. The same argument is offered with respect to the additional contacts required for methods 1 and 2. The maximum of 4 will be more acceptable when this number represents a smaller percentage increase. This will certainly be the case when the network has more elements and more external contacts.

Two design criteria can be suggested which would yield easily detectable networks: 1) limiting the number of feedback lines and the levels of logic over which the feedback is passed and 2) designing the network so as to keep the delay in all feedback loops nearly equal. These design criteria are attempts to strengthen the analogy between time domain and space domain analysis for the asynchronous case. The goodness of this analogy is the basis for the success of some of the FDT generation algorithms and for the backward drive secondary technique. However, since these design rules are not and can not always be followed, the need for a reliable set of secondary techniques is critical.

The success of existing methods for doing fault detection on general networks is a direct function of the network



being considered. Regardless of the effort spent in refining the FDT sequence, the level of fault coverage will be limited by the structure of the network. Although the additional cost required by the secondary techniques presented herein is dependent upon the input sequence, the level of fault coverage is much less dependent upon network structure.

This study indicates that if these secondary techniques are utilized in conjunction with a reasonable FDT sequence, total coverage of all single faults within a network is generally realizable.

## V

## APPENDIX

## Sample Networks

This section contains the circuit diagrams for the networks which were used in collecting the data that is presented in Section IV.B. Table A.1 lists the input sequences which were applied to the networks of Figures A.1-A.6 to obtain the data shown in Table 8.

The input sequences for the asynchronous networks were structured within the constraints of the single input change restriction.

Table A.1 does not display the timing diagram for the application of input sequences. In all sample runs, except E-2, the input variable vector has held level until the network stabilized.

It was found that changing the input vector, before the network stabilized, gave better fault coverage for sequence 2 on network E. This procedure was suggested by the observation of an apparent cyclic condition within network E under the application of sequence 1.

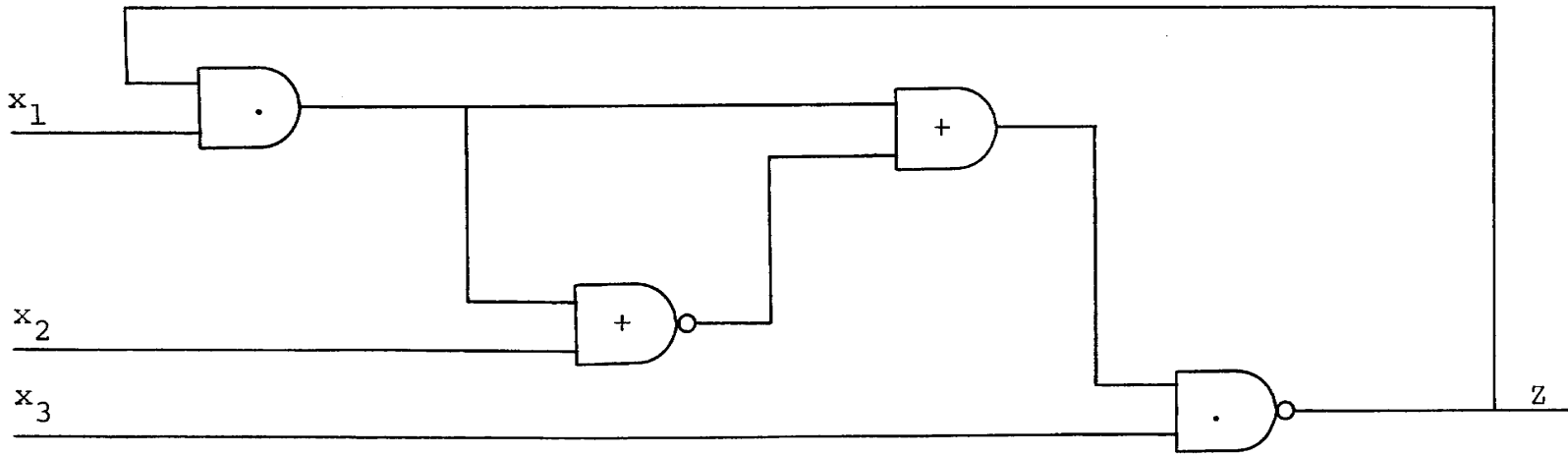


Figure A.1: Sample Network A

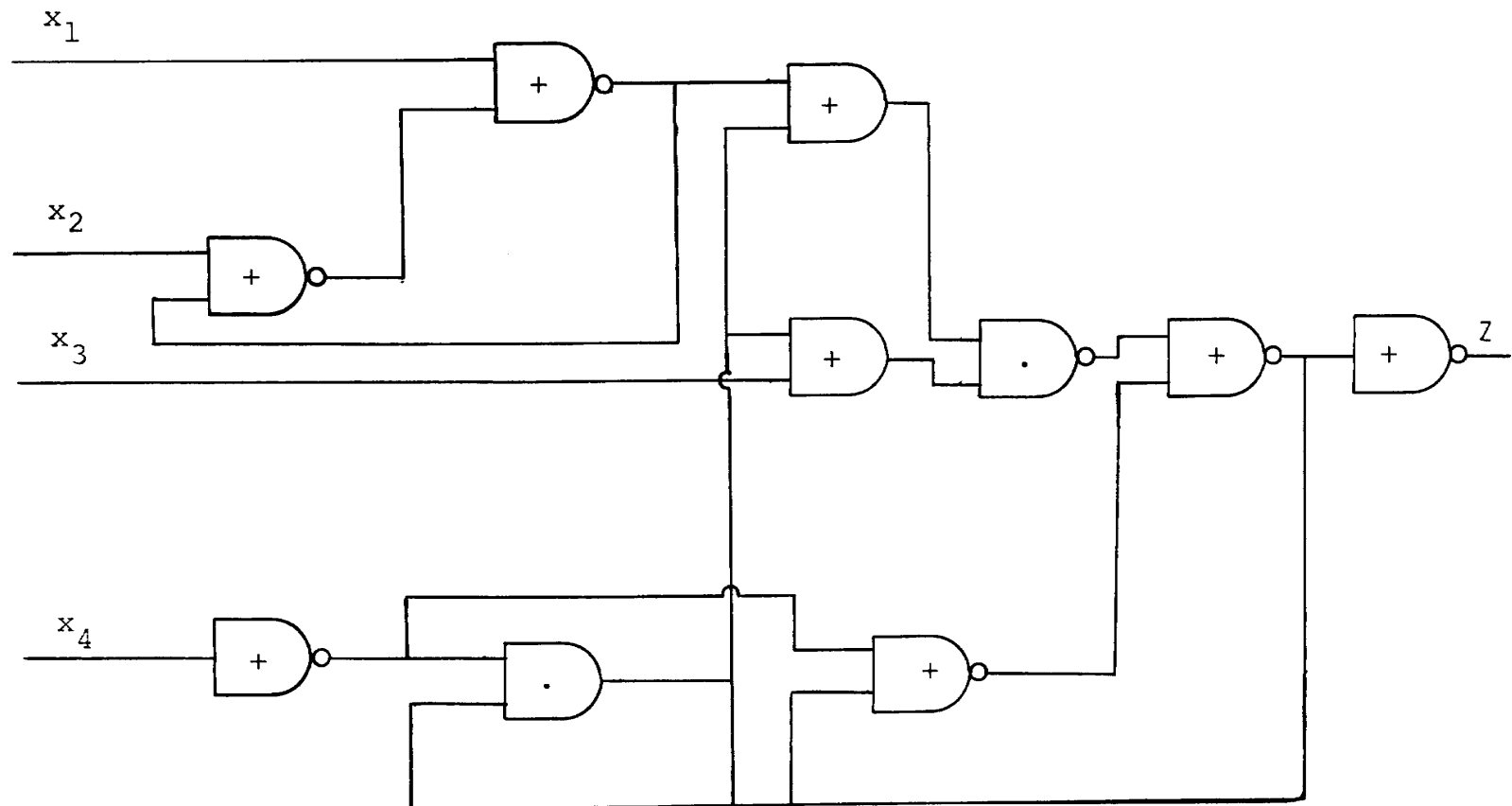


Figure A.2: Sample Network B

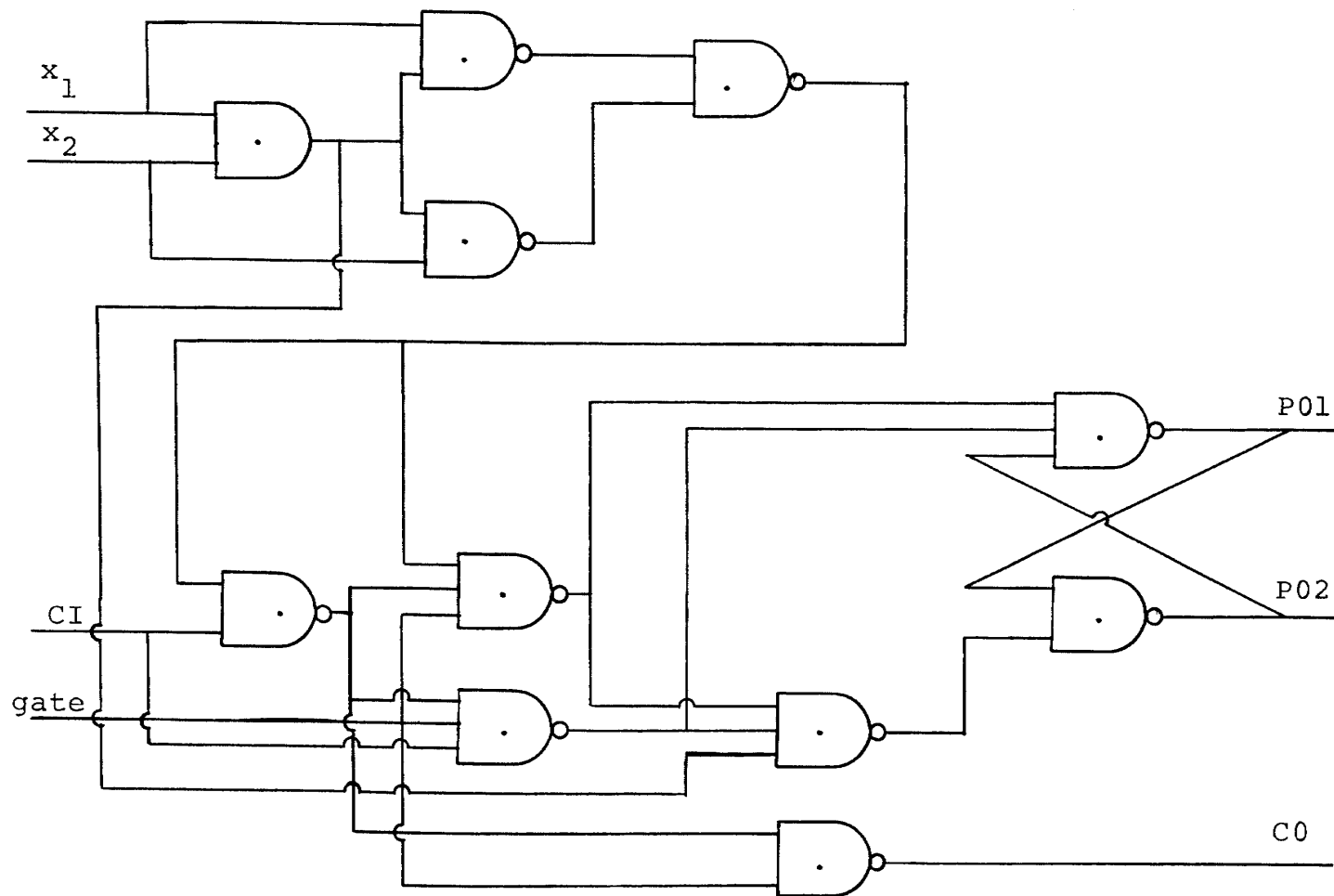


Figure A.3: Sample Network C (Latched Adder)

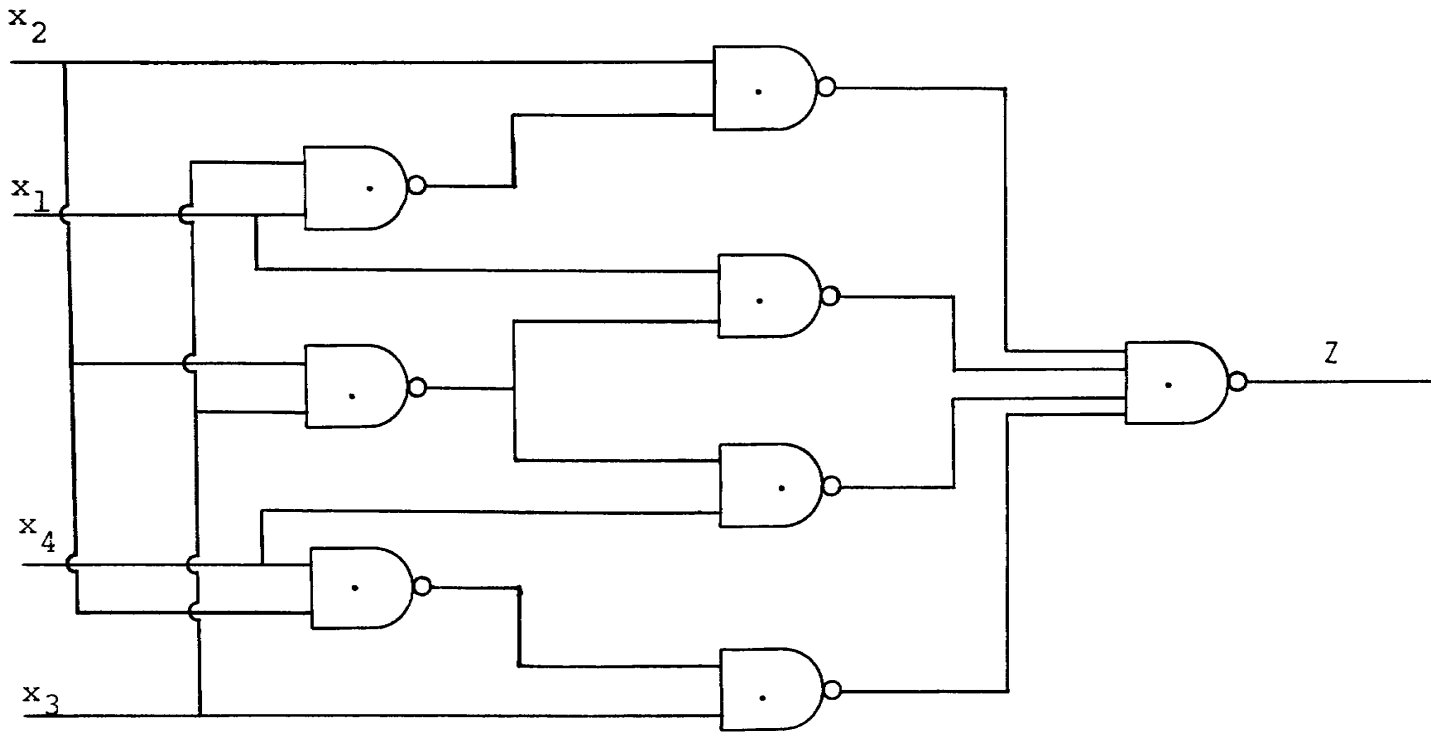


Figure A.4: Sample Network D

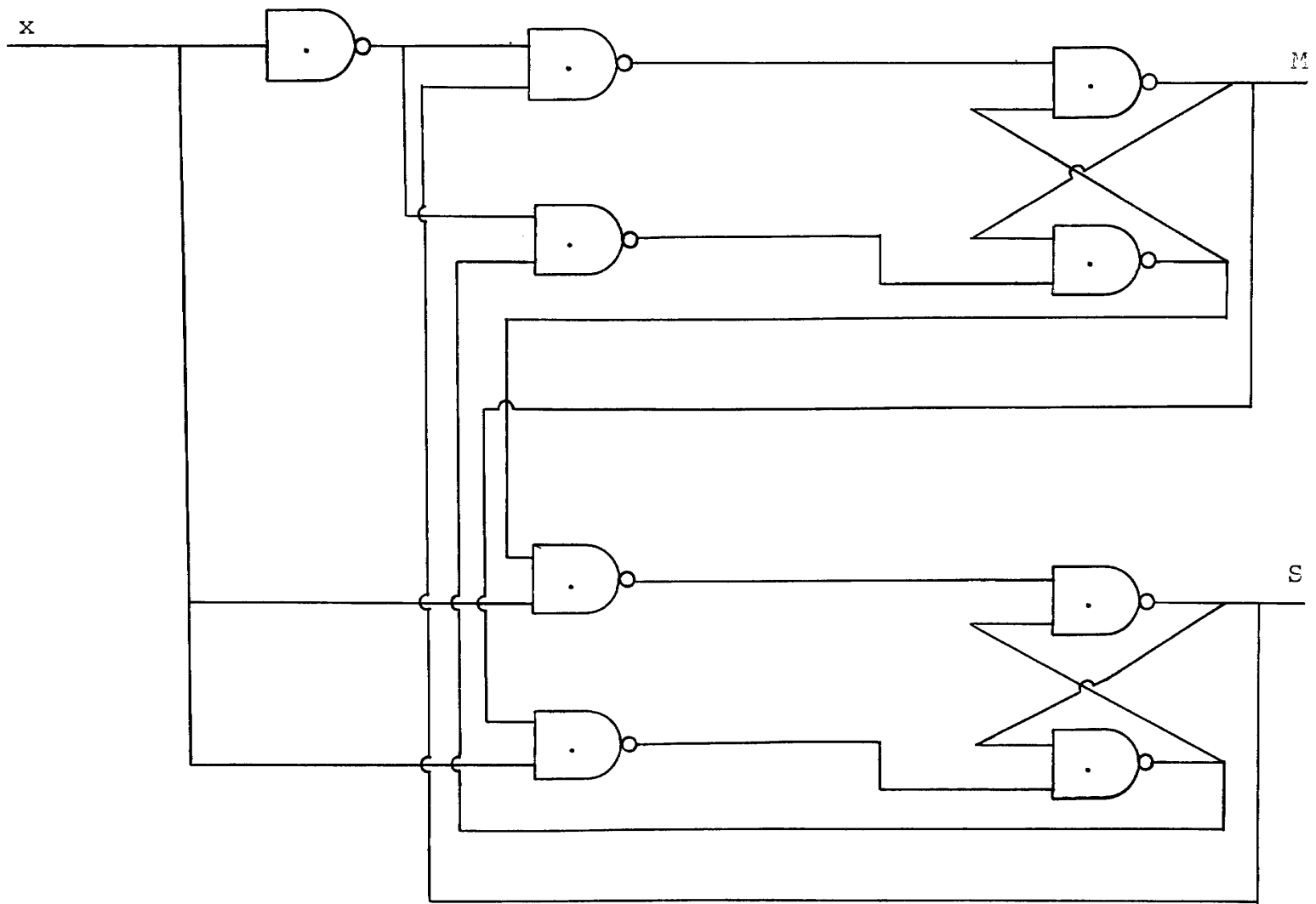


Figure A.5: Sample Network E (Master-Slave FF)

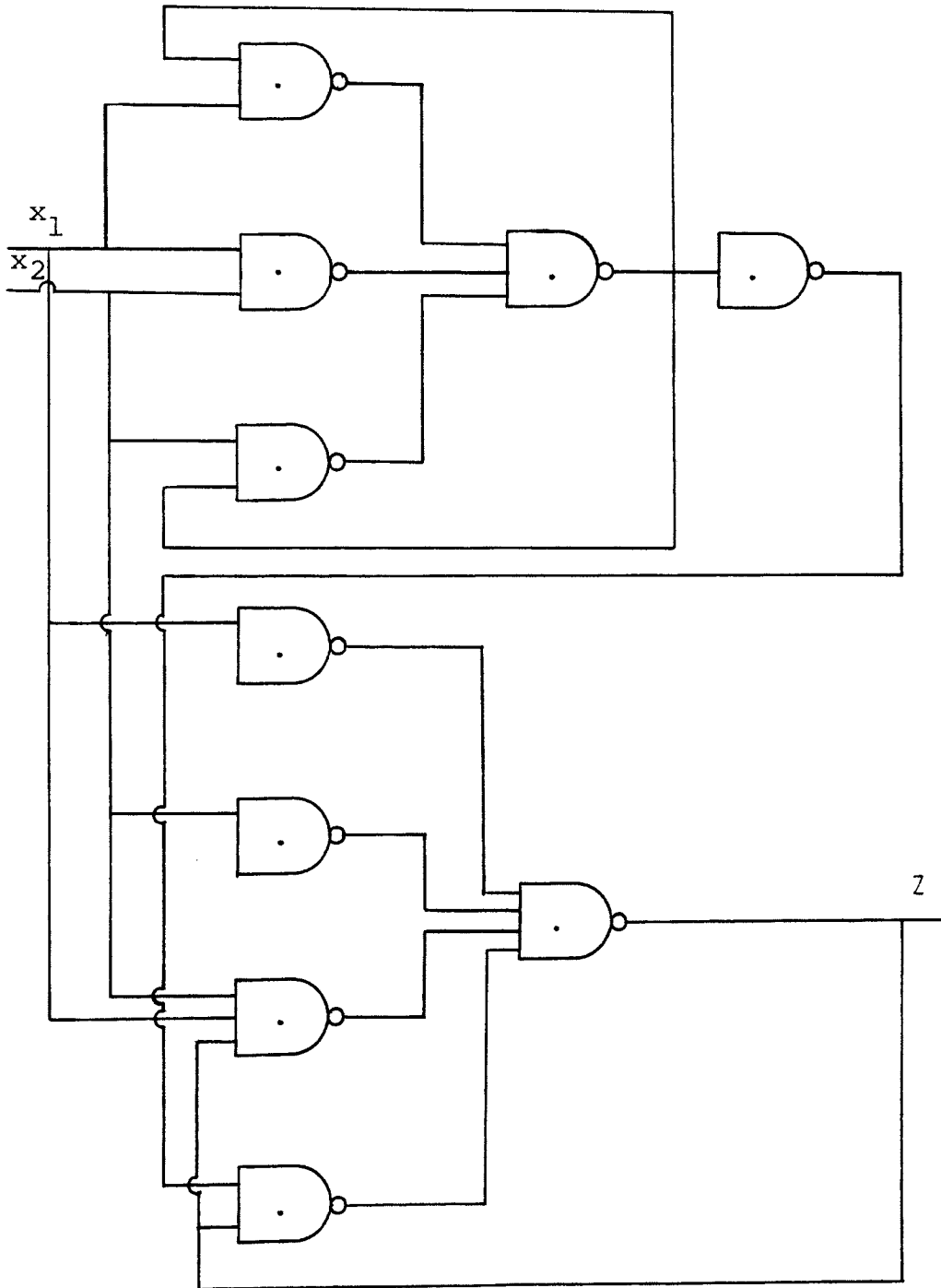


Figure A.6: Sample Network F



		Network					
		A	B	C	D	E	F
Sequence							
	1	000	0001	1100	0001	1	00
001		0011	1101	0000	0	01	
101		0111	1110	0001	1	11	
		1111	1111	0011		10	
		1110	0000				
			0001				
			0000				
2	000	0001	1100	0001	1	00	
	001	0011	1101	0000	0	10	
	101	0111	1110	0001	0	11	
	111		1111	0011	1	01	
	110		1110	1111	0	11	
					1	10	
						00	
3	000	0001		0001			
	100	0011		0100			
	101	0111		0101			
	111	1111		0111			
	110	1110		0000			
	010	1010					
	011	0010					

Table A.1: Input Sequences for Sample Networks

## BIBLIOGRAPHY

1. J.P. Roth, "Diagnosis of Automata Failures: A Calculus and A Method", IBM J. Res. Develop., Vol. 10, pp. 278-291, July, 1966.
2. M.Y. Hsiao and D.K. Chia, "Fundamentals of Boolean Difference for Test Pattern Generation", Proc. 4th Annual Princeton Conf. Inform. Sci., March, 1970.
3. G.R. Putzola and J.P. Roth, "A Heuristic Algorithm For the Testing of Asynchronous Circuits", IEEE Trans. on Elec. Comp., Vol. C-20, pp. 639-647, June, 1971.
4. M.Y. Hsiao and D.K. Chia, "Boolean Difference for Fault Detection in Asynchronous Sequential Machines", IEEE Trans. on Elec. Comp., Vol. C-20, pp. 1356-1361, Nov., 1971.
5. W.G. Bouricius et al., "Algorithm for Detection of Faults in Logic Circuits", IEEE Trans. on Elec. Comp., Vol. C-20, pp. 1258-1264, Nov., 1971.
6. M.A. Breuer, "A Random and an Algorithmic Technique for Fault Detection Test Generation for Sequential Circuits", IEEE Trans. on Elec. Comp., Vol. C-20, pp. 1364-1370, Nov., 1971.

7. A.D. Friedman, "Fault Detection in Redundant Circuits", IEEE Trans. on Elec. Comp., Vol. EC-16, pp. 99-100, 1967.
8. F.F. Sellers, M.Y. Hsiao and L.W. Bearnson, "Analyzing Errors With The Boolean Difference", IEEE Trans. on Elec. Comp., Vol. C-17, pp. 676-683, 1968.
9. C.R. Kime, "An Organization for Checking Experiments on Sequential Circuits", IEEE Trans. on Elec. Comp., Vol. EC-15, pp. 113-115, 1966.
10. F.C. Hennie, "Fault Detecting Experiments for Sequential Circuits", Proceedings of the 5th Annual Switching Theory and Logical Design Symposium, S-164, pp. 95-110, 1964.
11. Z. Kohavi and P. Lavallee, "Design of Sequential Machines With Fault Detection Capability", IEEE Trans. on Elec. Comp., Vol. EC-16, pp. 473-484, 1967.
12. Z. Kohavi and I. Kohavi, "Variable Length Distinguishing Sequences and Their Application to the Design of Fault Detection Experiments", IEEE Trans. on Elec. Comp., Vol. C-17, pp. 792-795, 1968.

13. A. Ashkinazy, "Fault Detection Experiments for Asynchronous Sequential Machines", Conference Record of the Eleventh Annual Symposium on Switching and Automata Theory, pp. 88-93, October, 1970.
14. S. Seshu and D.N. Freeman, "The Diagnosis of Asynchronous Sequential Switching Systems", IRE Trans. on Elec. Comp., Vol. EC-11, No. 4, pp. 459-465, August, 1962.
15. H.V. Chang, E. Manning, G. Metze, Fault Diagnosis of Digital Systems, New York: John Wiley and Sons, 1970, pp. 29-47.
16. D.M. Rouse, "A Simulation and Diagnosis System Incorporating Various Time Delay Models and Functional Elements", Ph.D. Dissertation, University of Missouri - Rolla, Rolla, Missouri, 1970.
17. P.R. Schneider, "On the Necessity to Examine D-Chains in Diagnostic Test Generation - An Example," IBM Journal of Research and Development, Vol. 11, p. 14, 1967.

## VITA

Lewis Ronald Hoover was born on July 23, 1940, in Martinsburg, Pennsylvania. He received the Bachelor of Science degree in 1962 from Shippensburg State College. Following graduation, he married Bonnie Lou Spealman, a college classmate.

In August, 1963, he was awarded the M.A. degree from Washington University, St. Louis, Missouri.

He was on the Physics Department faculty of Shippensburg State College from 1964 to 1969. During the summers of 1967, 1968, and 1969, he was a graduate student in the Computer Science Department of the University of Missouri - Rolla. In 1970 he was on the staff of the same department.

Along with his duties as the father of three young daughters, he is currently studying under a National Science Foundation Fellowship in the Electrical Engineering Department of the University of Missouri - Rolla.

He is a member of Phi Sigma Pi and the Institute of Electrical and Electronic Engineers.

Upon completion of his degree, he will begin duties as a Professor of Mathematics and Computer Science at West Chester State College, West Chester, Pennsylvania.