INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Xerox University Microfilms

300 North Zeeb Road
Ann Arbor, Michigan 48106

76-30,264

CHUANG, Chin Sheng, 1948-
REAL TIME, ON-LINE SELF-TESTING DIGITAL SYSTEM DESIGN.

City University of New York, Ph.D., 1976
Engineering, electronics and electrical

Xerox University Microfilms, Ann Arbor, Michigan 48106

© COPYRIGHT BY

CHIN SHENG CHUANG

1976

REAL TIME, ON-LINE
SELF-TESTING DIGITAL SYSTEM DESIGN

by

CHIN SHENG CHUANG

A dissertation submitted to the Graduate Faculty in Electrical Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1976

This manuscript has been read and accepted for the Graduate Faculty in Electrical Engineering in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Aug. 23, 1976
date

Se Jeung Oh
Chairman of Examining Committee

Aug 23, 1976
date

Jacques E. Benveniste
Executive Officer

Professor Se Jeung Oh

Professor Ralph Mekel

Professor Frederick Thau
Supervisory Committee

The City University of New York

ABSTRACT

REAL TIME, ON-LINE
SELF-TESTING DIGITAL SYSTEM DESIGN

by

Chin Sheng Chuang

Adviser: Professor Se Jeung Oh

Rapid real time fault detection is essential in some digital application enviroments. Several approaches to enhance the testability of digital system by means of redundant hardware have been proposed recently.

In this dissertation, a new concept of systematic utilization of redundant hardware is introduced to reduce the number of test required for the fault detection of either a combinational or a synchronous sequential network, and to simplify the test generation procedure. Model for PLM(programmable logic module) and CMM(controllable memory module) are depicted to enhance the testability of a given network. The network topology which will enhance its testability is first explored. Systematic design algorithms to implement either a combinational or a synchronous sequential network which will satisfy the specified topology are then developed, and the corresponding fault detection procedure is inverstigated. Finally, a real time, on-line, self-testing digital system is proposed and the performance of the system is examined.

Some favorable results, obtained by paying the additional cost of the redundant hardwares required in the proposed modules in this research, can be summarized as follows:

1. A combinational network can be designed such that its stuck-at-faults can be detected by two test input patterns generated by the developed design algorithm.
2. A synchronous sequential network can be designed such that its stuck-at-faults and malfunctions can be detected by four test input patterns generated by the design algorithm.
3. A real time, on-line, self-testing digital system can be implemented by extending the developed design algorithms and with the utilization of a hard core read only memory and hard core comparator so that the stuck-at-faults and malfunctions of the system can be detected by 4M tests (M is the number of functional blocks contained in the system).

ACKNOWLEDGEMENTS

The author wishes to express his deep appreciation to his dissertation research committee chairman, Professor Se Jeung Oh, for his guidance throughout the period of research and writing of this dissertation, and to Dr. A. Friedes who carefully read this dissertation and made several useful comments, and to Professors R. Mekel and F. Thau for their helpful advice. His greatest thanks go to his wife, Win, for her encouragement, support and understanding.

This work was supported in part by NASA-Houston under the NASA grants NSG-5013, NAS9-13940 and NSG-7144, and in part by the Graduate Fellowship from the City University of New York.

TABLE OF CONTENTS

	Page
COPYRIGHT PAGE	ii
APPROVAL PAGE	iii
ABSTRACT	iv
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 TESTABILITY AND DESIGN FOR COMBINATIONAL NETWORK	12
2.1 Introduction	12
2.2 Circuit Notation and Definitions	13
2.3 Fault Analysis and Fault Classification	19
2.4 Input Sensitivity and Programmable Logic Module	24
2.5 Design with Testability Enhancement ...	30
2.6 Systematic Design Algorithm	46
CHAPTER 3 SYNCHRONOUS SEQUENTIAL NETWORK AND TESTABILITY ENHANCEMENT	59
3.1 Introduction	59
3.2 Circuit Notation and Definitions	61
3.3 Fault Detection Analysis and Fault Model	69
3.4 Network Topology and Controllable Memory Module	77
3.5 Design with Testability Enhancement ...	92
3.6 Systematic Design Algorithm for Synchronous Sequential Network	103

Page

CHAPTER 4 APPLICATION OF TESTABILITY ENHANCEMENT IN SYSTEM DESIGN 121

4.1 Introduction 121

4.2 System Structure Modeling and Analysis .. 122

4.3 Design of Real Time Fault Detection Functional Block 131

4.4 System Fault Detection 174

4.5 Summary 179

CHAPTER 5 ANALYSIS OF SYSTEM PERFORMANCE 181

CHAPTER 6 CONCLUSIONS 193

REFERENCES 196

AUTOBIOGRAPHY 201

LIST OF TABLES

Page

2.1 The Most Sensitive Input Pattern of N-Input Elementary Gates 26

2.2 Four Basic Programmable Logic Modules 31

2.3 The General Rule of Determining the Control Signal of the PLM's 41

3.1 Excitation Requirement of the Four Basic Flip-Flops 66

4.1 The Elementary Operation of Accumulator Register 136

4.2 Summary of the Test Patterns of Each Functional Block 175

5.1 Mean Time to Failure of Gate and Module in Three Models 189

LIST OF FIGURES

		Page
2.1	Logic Symbols and Definitions of Fundamental Logic Gates	14
2.2	Networks with Different Structural Characteristic ..	17
2.3	An 2-Input NAND gate	23
2.4	A Network with an Undetectable Fault	23
2.5	A Network with Special Topology	28
2.6	An Implementation of Four Basic 2-Input PLM's	32
2.7	A General Representation of Programmable Logic Module (PLM)	33
2.8	A Restricted Dual Network	36
2.9	A Network Satisfies the Topological Requirement of Theorem 2.4	39
2.10	A Typical Case-1 Network	42
2.11	A Typical Case-2 Network	44
2.12	A Network Illustrates the Definition of L(G)	47
2.13	Flow Chart of the Systematic Design Algorithm for Combinational Network	49
2.14	The Relation between the Control Signals C_3 and C_4 .	52
2.15	Network of Example 5	55
2.16	Network of Example 6	57
3.1	Canonical Model of Sequential Network	62
3.2	A General Representation of State Table and State Diagram	68
3.3	Logic Symbols of the Four Basic Master-Slave Flip-Flops	68
3.4	A Two Bit Shift Right/Shift Left Register	79

Page
3.5 The Directed Graph of the Network of Figure 3.4 ... 80

3.6 A Special Structure of Synchronous Sequential Network for Theorem 3.3 83

3.7 A Synchronous Sequential Network Illustrates Theorem 3.3 83

3.8 A Specific Configuration of Synchronous Sequential Network for Theorem 3.4 87

3.9 A Synchronous Sequential Network Illustrates Theorem 3.4 87

3.10 A General Model of Controllable Memory Module (CMM) 90

3.11 Logic Symbols of the Four Basic CMM's 90

3.12 Two Basic Structures of the Memory to Memory Signal Path (mmsp) 93

3.13 A Synchronous Sequential Network with no mmsp 96

3.14 A Synchronous Sequential Network with mmsp 99

3.15 Another Control Signal Setting of Network Shown in Figure 3.13 102

3.16 The Topology of a General Synchronous Sequential Network 104

3.17 Flow Chart of the Systematic Design Algorithm for Synchronous Sequential Network 106

3.18 Modified Two Bit Shift Right/Shift Left Register .. 114

3.19 An 00-01-10-00 Sequence Detector and Its Modified Version 117

4.1 A Simplified Stored Program Digital Processor 125

4.2 A Typical Configuration of Bus Structure and Its Timing Relationship 127

4.3 State Diagram Illustrates the Sequential Operations of the System 129

Page

4.4 A Typical Control Gating Network in Control Sequence Generator 132

4.5 Block Diagram of an Accumulator Register 134

4.6 An Implementation of a Typical Stage of Accumulator Register 135

4.7 A Modified Version of Network Shown in Figure 4.6 137

4.8 Another Description of the Block Diagram of Accumulator Register 140

4.9 A Modified Accumulator Register in the Bus Structure 143

4.10 A Typical Structure of Program Counter and Its Modified Version 145

4.11 A Network Configuration of Instruction Register and Its Modified Version 150

4.12 Block Diagram of Control Sequence Generator 155

4.13 A Five Stages Ring Counter and Its Modified Structure 156

4.14 A Structure of k words of n-Bit Memory Unit and Its Modified Version 160

4.15 A Typical Configuration of Memory Address Register and Its Modified Version 165

4.16 A Typical Stage of Memory Data Register and Its Modified Structure 169

4.17 A Combined Structure of Modified Memory Unit and Modified Memory Data Register 173

4.18 An Architectual Configuration of a Real Time Self-Testing Digital System 177

5.1 Reliability Comparison Between Gate and Module with Constant Failure Rate Model 185

5.2 Reliability Comparison between Gate and Module with Linearly Increasing Failure Rate Model 186

5.3 Reliability Comparison between Gate and Module with Non-Linearly Increasing Failure Rate Model 187

CHAPTER 1

INTRODUCTION

Many digital system applications, such as in space missions, defense systems, medical instrumentations or communication systems, demand high accuracy in system performance to insure correct operation of the system. To meet such a requirement, rapid real time fault detection will be necessary. One approach to satisfy this requirement is to include efficient fault detection feature in the system from the initial design stage.

The objective of this dissertation is to develop systematic techniques to design a digital system incorporating the capability to perform self-testing for fault free operation, on-line and in real time.

To achieve this goal, the system designed according to the specified method must be capable of self-testing its operation with least interruption to the on-going process. This requires the test operation to be simple and the time required for testing to be short. Consequently, the test generation and the testability of the system must be incorporated into the design requirements.

In this dissertation research, the problem was investigated in the following sequence:

1. Explore the network topology which will enhance

the testability of a network.

2. Develop systematic design algorithms to implement a network which will satisfy the specified topology.
3. Design a real time, on-line, self-testing digital system.
4. Examine the performance of the designed system.

The results are contained in Chapter 2 through 5, and summarized briefly at the end of this chapter.

In recent years, systematic approaches to derive test sets to detect or diagnose faults in digital systems have received a great deal of attention, and various techniques have been developed [7], [14]. The existing methods can be broadly classified into the following categories:

A. Tree-Partioning Approach

Brule and others [6] proposed a systematic method to generate sequential tree-diagram describing every possible response to certain stimulants and good functional elements (gates, flip-folps, modules) by designating a binary code for all possible cases. The number of essentially different tests for a circuit containing N elements is determined to be $(2^{N-1}-1)$, while only one faulty element is assumed. Seshu and Freeman [43], [44] apply this technique and Moore's [31] earlier work on multiple experiments to detect the faults in asynchronous sequential network and generate a

tree graph in which each node represents a set of equivalent machines (different faults will make the normal machine function differently) and each branch, a possible output response. The test patterns are chosen by an algorithm to minimize both the number of machines in the equivalent set containing the good machine, and the average number of tests required to identify each possible failure. The algorithm usually requires long computation to find tests for the last few faults. Furthermore, the algorithm requires the machines under test to be reset to some past state so that a different sequence of tests can be performed. However, in practice, it is not always possible to return the machines to a specified past state.

B. Fault Matrix

For a single output combinational network, Chang [8] introduced D-matrix in which the entry d_{ij} of the matrix is 1 if a test t_j detect a fault f_i. Otherwise $d_{ij} = 0$. Kautz [23] proposed an F-matrix in which rows represent all possible input patterns (there are 2^n possible rows for n input variables), and columns represent the fault-free output and the faulty outputs corresponding to all possible faults. A G-matrix is generated by performing exclusive-or operation between the fault-free column and each faulty column. The reduced G-matrix in which column j is deleted if column j covers column i and the row j is deleted if row

j is covered by row i will produce an optimal or a minimal set of tests. Kautz also extended this technique to build a fault-matrix for multiple output combinational networks. It is readily observed that the matrix becomes too large to handle when the number of input variables or the number of possible faults is large. Powell [33] suggested a probability weighting procedure to find a nearly best test set that diagnoses single fault to IC package level. This approach is more practical than the Kautz's because it does not examine exhaustively all possible combinations of tests. Russel and Kime [39], [40] extended the concept of fault matrix to define a diagnostic model of a system containing at most t-faulty components to determine the detectability and the diagnosability of the given system.

C. Path Sensitizing Approach

Armstrong [1] suggested reducing a combinational network to its equivalent normal form by expanding the Boolean expressions to the standard sum of products form in which each literal consists of an input variable subscripted by a sequence of gate number. A path to detect a faulty literal is identified by its subscript sequence. An algorithm for selecting near minimal tests is presented in the paper. However, this method is not guaranteed to detect all the stuck-at-failures if the network contains reconvergent fan-out. This problem was solved by the D algorithm deve-

loped by Roth [36], [37]. Roth introduced a mathematical model (D-cube) to represent the propagation of information along a set of gates in a combinational network. The D-algorithm gives us a systematic routine to compute tests to detect failure by D-intersecting the failure D-cubes and the normal D-cubes recursively until a connected D-chain to some primary observable output is formed, and utilizing the consistency (trace back) operation to determine the primary input in order to realize the sensitized paths. Similar concepts have been applied to develop a systematic method [30], [5], [19], [17], [35] for deriving test sets of a given combinational network. With increasing utilization of large scale integrated circuits (LSI), these methods become impractical since the complexity of the circuit will require extremely large number of operations. Su and Cho [47] used Roth's D-algorithm to generate a complete set of tests for diagnosing all the possible single failures in a combinational logic network up to a fatal family level, and in addition, introduced the decision making algorithm to select the essential test based on previous test results and the structure of the network. The total number of tests required to diagnose a fault is reduced by this decision making process. McCluskey and Clegg [30] introduced the fault equivalent class to simplify the test derivation process. It is extended later by To [48], Roy

[38] and Dias [13] to reduce the time required to generate a complete test set of a combinational network.

D. State Table Checking Experiment

Henni [21] proposed the transition checking method for a sequential machine. If the state table has distinguishing sequence, it serves as a locating sequence for each state. Therefore, the length of diagnosing sequence will be no more than $2n(m+1)(L+n)$. If not, the various characterizing sequence are used to serve as a locating sequence for each state, and the length of diagnosing sequence increases rapidly and is bounded by $mn^4(n+1)$, where m = the number of input variables, n = the number of states and L = the length of distinguishing sequence. Consequently, the utility of this mehhod becomes rather limited when n or m is large. Kime [24] modified Hennie's procedure to reduce the upper bounds on the length of the experiment by selecting a set of tests in the tree state diagram describing the system under test. Kohavi and Lavalle [25], [26] extended Hennie's idea to construct shorter fault detection experiments for any arbitrary sequential machine by adding additional output logic to make the machine have some special distinguishing sequences. The resulting upper bounds on the length of checking experiment are $nm + n(m-1)L + L + (m-1)(n-1)^2$ and $(m-1)n^3$ respectively for the two cases. The checking experiment approach has been further developed by several

other researchers [32], [18], [22], [15], [12]. All efforts to reduce the upper bound on the length of checking experiment by modifying a given sequential network through adding extra outputs or inputs. The resulting upper bound (W) is proportional to the number of inputs and some power of the number of states (i.e. $W \leq mn^k$ where $k > 1$) of a sequential network. For a complex system, these techniques will generate a long test sequence and require a complex algorithm to generate this sequence.

E. Mathematical Approach

Sellers et al [35] used Boolean difference to analyze the effect of errors on the output of logic circuit by exclusive-or operation between two Boolean expressions (one for normal circuit, the other for faulty circuit). The diagnostic test sequence can be properly derived by finding the Boolean difference for each fault. Several researchers [46], [2], [50] extended this idea to develop algorithms for deriving the test sets for a combinational network under a single fault assumption. Ku and Masson [27] expanded the concept of Boolean difference to derive test sets for multiple fault detection in a combinational network. The disadvantage of these techniques is that it is essential to exercise a Boolean difference for every possible combination of faults when the network has multiple faults. In addition, the derivation of each Boolean difference is a time consuming process.

Ramamoorthy [34], [29] applied the graph theoretic ideas since any discrete sequential system is isomorphic to a directed graph in which each node represents a functional element (gate, flip-flop or module) and each directed branch, a signal propagating line. He converted a given machine into a graph and partitioned it into its link subgraph and maximal strongly connected subgraph to form a reduced graph. The test points are then added to provide additional segmentation of a large subsystem and to monitor the resulting outputs.

F. Network Modification Approach

Williams and Angell [49] made use of additional logic gates in conjunction with test points to provide an easy means to set or check each state, in which one test point is used to switch the network to a test mode by reconfiguring all the flip-flops to form a shift register. Therefore the proper function of each flip-flop can be easily monitored and the testability of the network is thus enhanced. Sakauchi and Inose [41] introduced a function conversion method to modify a network structure and noted that certain fault detection testing can be achieved rapidly if the structural configuration of a network satisfies specific topological constraints. Hayes [20] inserted a redundant exclusive-or gate into each input of NAND gate as a control logic to simply the test generation and reduce the number of tests required to detect the stuck-at-fault of a com-

binational network to five tests. This technique, on the other hand, restricts the network to be implemented by NAND gates. Chuang and Oh [9], [10] proposed programmable logic modules and controllable memory modules as new building elements to design a network whose topology will satisfy certain topological constraint for enhancing the testability of the network.

From the above survey, we note that most of the existing techniques, except the network modification approaches, are devoted to the derivation of test sets for stuck-at-fault detection of a given network. For a complex network, the derivation process based on complicated algorithms which will demand a great deal of computer time and memory to generate the test sets (or sequence). Consequently, these methods are not suitable for real time application.

In this dissertation, Chapter 2 defines the concept of the most sensitive input pattern of an ordinary logic gate and the concept of a programmable logic module. These new ideas were utilized to develop an approach to enhance to testability of a network with special topology. Some theorems are derived to establish that a combinational network can be implemented so that the single stuck-at-fault in the logic module and multiple stuck-at-fault in the network can be detected by two test input patterns. Finally, a systematic design algorithm and a fault detection proce-

dure are developed to design a modified combinational network having two test input patterns which detect all the possible stuck-at-faults in the network regardless of the complexity of the network.

Fault detection in a synchronous sequential network is treated in Chapter 3. A new concept of the fault model of the four basic flip-flops is proposed in which a single stuck-at-fault is assumed. Certain structures of a synchronous sequential network are investigated which impose some topological constraints for designing a real time fault detectable synchronous sequential network through the use of the proposed modules. A systematic design algorithm and a fault detection procedure are then developed to implement a synchronous sequential network which is capable of being tested by four test input patterns for detecting the presence of stuck-at-faults and malfunctions in the network.

In Chapter 4 we extend the developed design techniques to the system level design. A simplified stored program digital processor is considered. A real time self-testing digital system is then established which will require at most 4M tests (M = the number of functional blocks contained in the system) for detecting the stuck-at-faults and the malfunctions of the entire system, and the upper bound, 4M, is independent of the complexity of the individual

functional block.

The performance of a system designed by the conventional techniques and the proposed techniques are analyzed and compared in Chapter 5. It is shown that a highly reliable digital system can be implemented by the techniques developed in this dissertation.

Finally, the concluding remarks are summarized and future topics are suggested in Chapter 6.

CHAPTER 2

TESTABILITY AND DESIGN FOR COMBINATIONAL NETWORK

2.1 Introduction

Some basic notations and definitions for combinational networks are first introduced to define and describe the network both functionally and structurally. Then the basic fault model, in a general sense, is discussed by the use of the fault monitor E_{ik}^{j}. The input sensetivity is introduced and the programmable logic module is proposed in order to enhance the testability of a network.

A network with special topology is introduced and its characteristics are investigated based on their topological relationships. These properties enable us to design an easily testable network to detect any single or multiple stuck-at-faults of the network by two test input patterns. Undetectable or masked faults present in some redundant networks containing fan-out nodes and/or reconvergent fan-out nodes will invalidate the test set generated for certain fault models. This problem can be solved by the design technique developed later in this chapter (section 2.5).

Finally, a systematic design algorithm and a fault detection procedure are presented which enables us to design any combinational network such that two test input patterns suffice to detect all the possible stuck-at-faults in the network regardless of the size of the network.

2.2 Circuit Notation and Definitions

Let B=(0,1) be a binary set. A general Boolean switching function F of n-inputs and m-outputs can be defined by the mapping

$$F : B^n \rightarrow B^m$$

F is called a multi-output switching function whose domain is the binary valued n-tuple vectors and codomain is the binary valued m-tuple vectors.

If we let $x_1, x_2, \ldots, x_n$ be the n input variables where $x_i \in B$ for all $1 \le i \le n$, then the single-output Boolean switching function f ($f : B^n \rightarrow B$) can be defined as $f = f(x_1, x_2, \ldots, x_n)$. Consequently F can be considered as an ordered m-tuple vector of single-output Boolean functions, i.e., $F = (f_1, f_2, \ldots, f_m)$, where $f_i = f_i(x_1, x_2, \ldots, x_n)$ for all $1 \le i \le m$.

For n=2, m=1, the elementary 2-input, single-output logic gate can be defined and classified into 16 (i.e., 2^{2^n}) different switching functions based on the binary operator AND(.), OR(+), and complement('). Figure 2.1 shows the logic symbol and its corresponding function of the most commonly used logic gates, and the symbols will be used throughout the thesis.

A <u>combinational network</u> is a physical realization of a Boolean logic gate without feedback between them.

Let C be an n-input and m-output combinational network, and $x = (x_1, x_2, \ldots, x_n)$ be the input vector and $F = (f_1, f_2, \ldots, f_m)$ be the output vector of C. We call the input variables

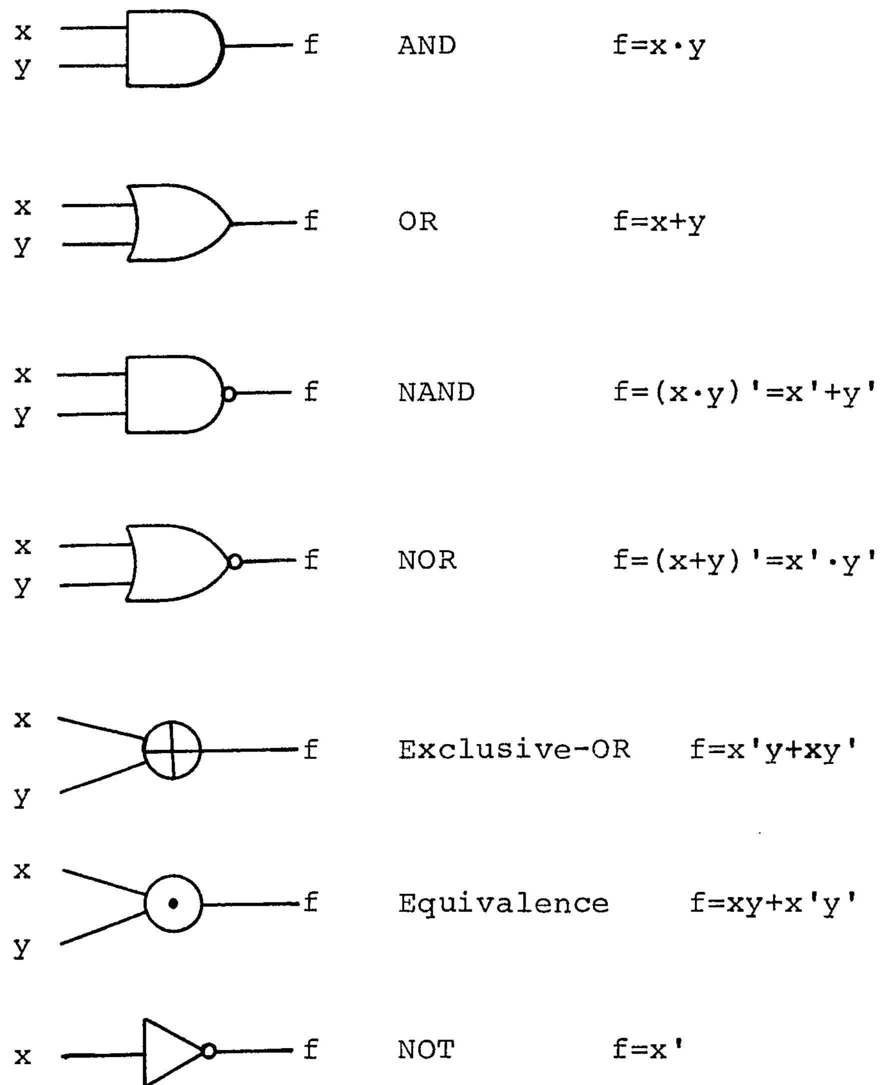


Figure 2.1
Logic Symbols and Definitions of Fundamental Logic Gate

x_1, $x_2, \ldots,$ x_n the primary inputs (PI) which are always observable. Therefore the network C has 2^n possible primary input vectors and 2^m possible primary output vectors.

Let $X_k = (x_1, x_2, \ldots, x_n)_k$ where k is the equivalent binary integer defined by the input vector X_k. Let $\mathcal{X} = (X_0, X_1, \ldots, X_{2^n-1})$ be all the 2^n possible PI vectors. If we substitute $X=X_k$ in $f(X)$, then $f(X_k) \in B$ will be the value of f defined by PI vector X_k. Similarly, the output vector F can be defined as

$$F(X_k) = [f_1(X_k), f_2(X_k), \ldots, f_m(X_k)]$$

From the graphic point of view, every combinational network has its equivalent directed graph. The direction of the edge indicates the signal flow between the nodes. Some definitions which relate the graphic property of a combinational network are listed as follows:

1. Each gate in the network defines a signal node and set of all signal nodes is denoted as SN.
2. Each primary input variable in the network defines a primary input node and the set of primary input nodes is denoted as PIN.
3. Each primary output variable in the network defines a primary output nodes and the set of all primary output nodes is denoted as PON.
4. Any interconnection between two nodes (either signal node, primary output node or primary output node) is

called a signal line and the set of all signal lines is denoted as SL.

5. The number of nodes to which a node g fans out is called as the fanout of node g and is denoted as λ_g.

6. A node g is called a fanout node if $\lambda_g \geq 2$, and the set of all fanout nodes is denoted as FN.

7. A node having some pair of inputs which have at least one common preceeding fanout node is said to be a reconvergent node and the set of all reconvergent node is denoted as RN.

8. A network which has no RN in it is called non-reconvergent fanout (NRF) network.

9. A network whose FN is a empty set is said to be a fanout free (FF) network.

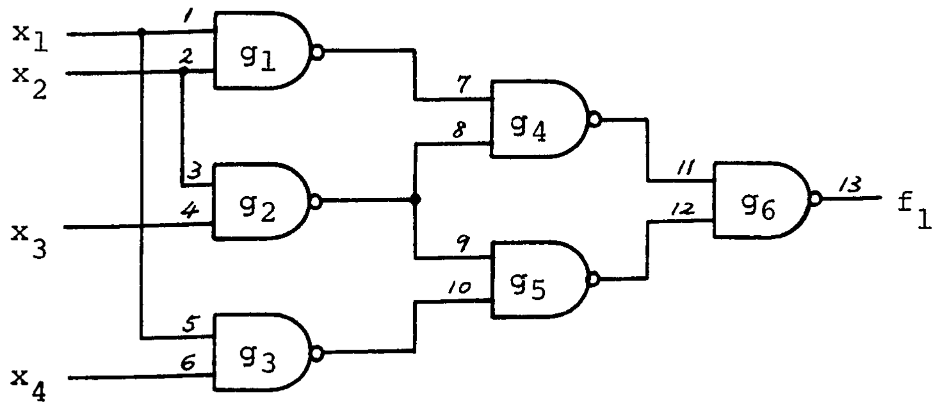
10. A network in which the only fanout nodes are PIN's is said to be a limited fanout free (LFF) network.

11. The signal line of a sequence of nodes from PIN to PON forms a signal path and the set of all signal paths is denoted as SP.

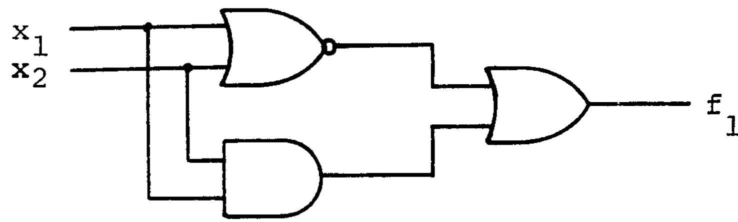
12. Let $|A|$ denote the number of elements in the set A.

Consider the networks shown in Figure 2.2. Network C_1 is properly labeled in order to show the properties of C_1 based on the above definitions:

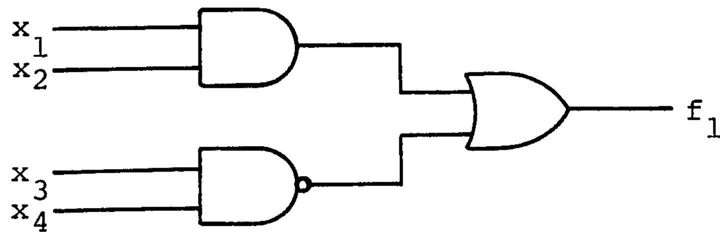
$$PIN = [x_1, x_2, x_3, x_4]$$



(a) Network C_1



(b) Network C_2



(c) Network C_3

Figure 2.2
Network with Different Structural Characteristic

$PON = [f_1]$

$SN = [g_1, g_2, g_3, g_4, g_5, g_6]$

$SL = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]$

$FN = [x_1, x_2, g_2]$

$RN = [g_6]$

$SP = [x_1g_1g_4g_6f_1, x_1g_3g_5g_6f_1, \ldots, \text{etc.}]$

It is obvious that C_1 is neither a FF, LFF nor a NRF network. Network C_2 illustrates an example of LFF network and network C_3, an example of FF network. Notice that if a network has RN, then its FN contains at least one element.

A <u>signal line</u> in a network is said to be <u>redundant</u> if and only if by removing the line and connecting it to a fixed binary values, the output function remains normal under all the possible input patterns. Otherwise, the signal line is said to be irredundant. The set of all the irredundant signal lines is defined as an irredundant set.

A <u>network</u> is said to be <u>irredundant</u> if and only if no set of SL can be removed and connected to a fixed binary values (0 or 1) without changing the output function of the network under all the possible input patterns.

<u>Theorem 2.1</u>: The SL, set of signal lines of a network is an irredundant set, if and only if the network is irredundant.

<u>Proof</u>: $\Leftarrow$ If a network is irredundant, then the removing of any line of SL will cause the output to change, so no line of SL can be removed. This indicates that the set SL is

irredundant.

⇒ If the set SL is irredundant, it means that any disconnection of signal line will cause the output to change, so that the network is irredundant.

2.3 Fault Analysis and Fault Model

The faults of a combinational network can be classified into two classes:

1. A permanent fault is a physical defect in the device which will cause the device to function incorrectly. These faults are usually associated with diode, transistor, connectors, etc. It generally causes a signal line to assume the constant logic value of 0 or 1 and is refered to as a stuck-at-0 or stuck-at-1 fault respectively.

2. An intermittent fault will also cause a device to function improperly, but it can not always be duplicated under the similar conditions which caused the previous intermittent fault. Intermittent faults may be caused by the use of device's marginal conditions and some enviromental failures. Since it is almost impossible to duplicate the intermittent fault, therefore, very difficult to detect them.

The most important fault class of combinational network in the currently used technologies is the permanent fault. The fault model of a network is restricted to be the single stuck-at-fault in the following discussions.

Consider an arbitrary network C, Let $H = [h^1, h^2, \ldots, h^q]$ be the set of all possible single stuck faults in C. Each fault h^j defines a mapping

$$h^j : C \rightarrow C^j$$

Let C^0 denote the fault free network of C, and C^j be the faulty network of C due to the existence of fault h^j. Let the output vector of C^0 and C be $F^0(X) = [f_1^0(X), f_2^0(X), \ldots, f_m^0(X)]$ and $F^j(X) = [f_1^j(X), f_2^j(X), \ldots, f_m^j(X)]$, for all $1 \leq j \leq q$, respectively. Then for any arbitrary input pattern X_k, there exists two possible conditions:

$$1.\ F^j(X_k) = F^0(X_k)$$

$$2.\ F^j(X_k) \neq F^0(X_k)$$

Condition 1 implies that the fault h^j is being masked under the input pattern X_k. In another word, input vector X_k can not detect the fault h^j. On the other hand, Condition 2 shows that the input pattern X_k will detect the fault h^j in C.

Hence, we define a fault monitor E_{ik}^j as follows:

$$E_{ik}^j = f_i^j(X_k) \oplus f_i^0(X_k) \in B$$

where $1 \leq i \leq m$, $1 \leq j \leq q$, and $0 \leq k \leq 2^n-1$.

If $E_{ik}^j = 1$, then we say that a fault occurs at the output line i due to a fault h^j under the input pattern X_k. Otherwise, there is no error on the output line i and the fault h^j in C is masked under X_k.

From the network functional point of view, the above

discussions show that all faults in C can be partitioned into the following two categories:

1. A fault h^j is said to be a <u>detectable fault</u> if there exists an input pattern set $\mathcal{X}^1$, where $\mathcal{X}^1$ is a nonempty subset of $\mathcal{X}$ such that

$$E^j_{ik} = 1 \text{ for some } X_k \in \mathcal{X}^1$$

2. A fault h^j is said to be an <u>undetectable fault</u> for all the input pattern X_k in $\mathcal{X}$, none of the output line will be faulty due to h^j, i.e.,

$$E^j_{ik} = 0 \text{ for all } 0 \leq k \leq 2^n-1$$

Let H denotes the set of all possible single stuck-at-faults of a combinational network C, then $|H| = 2 \cdot (|SN| + |PIN| + \sum_i \lambda_{gi}$ for all $g_i \in FN)$.
The proof is obvious from the fact that each primary input, signal node output and signal node input has two possible stuck-at-faults.

<u>Theorem 2.2</u>: A combinational network C is irredundant if and only if each fault in H, the set of all possible stuck-at-faults in C, is a detectable fault.

<u>Proof</u>:$\Leftarrow$ If each fault in H of C is a detectable fault, there always exists at least one input pattern X_k such that $E^j_{ik} = 1$ for some $1 \leq j \leq q$, then it implies that every fault in H will be detected. By definition, C is an irredundant network.

$\Rightarrow$ If C is an irredundant network, then any removal of signal line will change the output of the network. Therefore

every fault in C must be detectable, i.e., there exists at least one input pattern X_k such that $E^j_{ik} = 1$ for all j. This in turn implies that every fault in C is a detectable fault.

On the contrary, a network is redundant if and only if there exist at least one undetectable fault in the network.

In a single output network, the fault h^i and h^j are said to be <u>equivalent</u> under the input pattern X_k, if $E^i_{ik} = E^j_{ik} = 1$.

Let a signal line q stuck-at-1 be denoted as q/1, and q stuck-at-0 be denoted as q/0.

Consider a simple 2-input gate shown in Figure 2.3, the total set of single stuck-at-fault $H = [x_1/1,\ x_1/0,\ x_2/1,\ x_2/0,\ f/1,\ f/0] = [h^1,\ h^2,\ h^3,\ h^4,\ h^5,\ h^6]$ respectively, and $\mathcal{X} = [00,\ 01,\ 10,\ 11] = [X_0,\ X_1,\ X_2,\ X_3]$. We find that

$$E^2_{13} = E^4_{13} = E^5_{13} = 1$$
$$E^3_{12} = E^6_{12} = 1$$
$$E^1_{11} = E^6_{11} = 1$$

So faults h^3, h^4, h^5 are equivalent, so are the faults h^3, h^6 and the faults h^1, h^6. Notice that the faults h^1, h^3 are not equivalent since there exists no input pattern which will detect them simultaneously.

The network in Figure 2.4 shows an example of the existence of undetectable fault $x_3/1$ in the redundant network, since

$$E^j_{2k} = 0 \text{ for all } 0 \leq k \leq 7,\ j = x_3/1$$

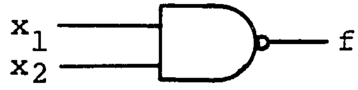


Figure 2.3
An 2-Input NAND Gate

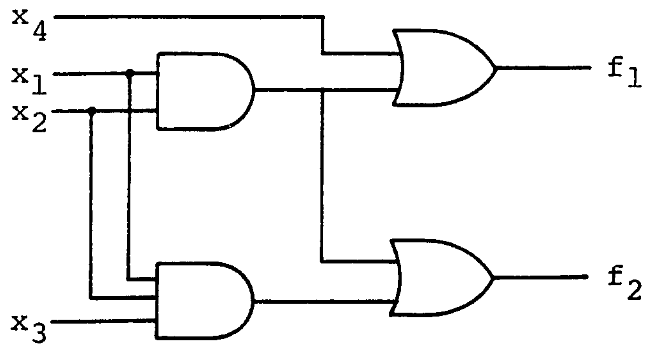


Figure 2.4
A Network with an Undetectable Fault

2.4 Input Sensitivity and Programmable Logic Module

The conventional techniques for detecting the stuck-at-fault in a combinational network C require the generation of the complete test set for detecting the fault set H in C. As the complexity of C increases, the complete test set increases. Thanks to the recent advances in semiconductor technology, the complexity of circuits fabricated on a single LSI chips tends to increase rapidly. The fault detection of a network utilizing MSI or LSI components poses an extremely difficult problem because of the structural complexity of components and their limited accessibility. Therefore, the conventional test generation scheme becomes impractical.

The question of how to enhance the testability of a network during the initial phase of the network design becomes an important problem. Several approaches to enhance the testability of a network by means of redundant hardware have been proposed recently [49], [41], [20].

This section introduces a new method of employing hardware redundancy to reduce the number of tests for fault detection of a combinational network.

Let $A \subset H$ where H is the set of all possible single stuck-at-fault in a network. Let $|A|$ be the number of element in set A. If for each fault $a \in A$, there exist an input pattern X_k such that $E^a_{ik} = 1$ for all a in A, then all the faults in A are equivalent under X_k, and $E_{ik} = |A|$ is

defined as the sensitivity of the input pattern X_k.

From the above example, $A = [x_1/0, x_2/0, f/1]$, $B = [x_1/1, f/0]$, $C = [x_2/1, f/0]$ where $A \cup B \cup C = H$, and $E_{13} = 3$, $E_{11} = E_{12} = 2$, $E_{10} = 1$

The input pattern x_k is said to be the <u>most sensitive input pattern</u> (MSIP) of a network if

$$E_{ik} \geq E_{pq} \text{ where } i \neq p,\ k \neq q \text{ and}$$
$$1 \leq i,\ p \leq m, \text{ and } 0 \leq k,\ q \leq 2^n-1$$

By this definition, we see that the MSIP for a two input NAND gate is $X_3 = 11$. It is also true for two input AND gate.

Consider a two-input OR gate, we find that

$$E_{10} = 3,\ E_{11} = E_{12} = 2,\ E_{13} = 1$$

by the similar analysis. Therefore $X_0 = 00$ is the MSIP for a two-input OR gate. It is also true for a two input NOR gate. The MSIP for a N-input ordinary gate is found by extending the same argument and is listed in Table 2.1.

Table 2.1 shows that the MSIP is unique for OR and NOR gates and for AND and NAND gates. The unique input pattern is independent of the number of inputs to the gate, and will detect half of the total $2(N + 1)$ faults. This characteristic provides us some means of optimum fault detection.

Notice that the equivalence ($\odot$) and exclusive-or ($\oplus$) gates do not have this property in which each input pattern in $\mathcal{X} = (00, 01, 10, 11)$ has equal sensitivity (i.e. $E_{1k} = 3$ for all $0 \leq k \leq 3$). Each input pattern in $\mathcal{X}$ is a MSIP

N-Input Gate	The Most Sensitive Input Pattern
OR	00...0 (N 0's)
AND	11...1 (N 1's)
NOR	00...0 (N 0's)
NAND	11...1 (N 1's)

Table 2.1 The most sensitive input pattern of N-input elementary gates.

and will detect half of the total possible faults.

A signal path is defined as a sensitizing path if the fault information can propagate along the path to the observable primary output under the test input pattern X_k.

Network A and network B are said to be topologically equivalent if their sets of signal nodes, fan-out nodes, reconvergent nodes, primary input nodes, primary output nodes and signal paths are the same.

Consider networks A and B shown in Figure 2.5. A is topologically equivalent to B by definition. Both networks have the following property

$$PIN = [x_1, x_2, \ldots, x_7]$$

$$PON = [\ f_1]$$

$$SN = [g_1, g_2, \ldots, g_6]$$

$$FN = \emptyset$$

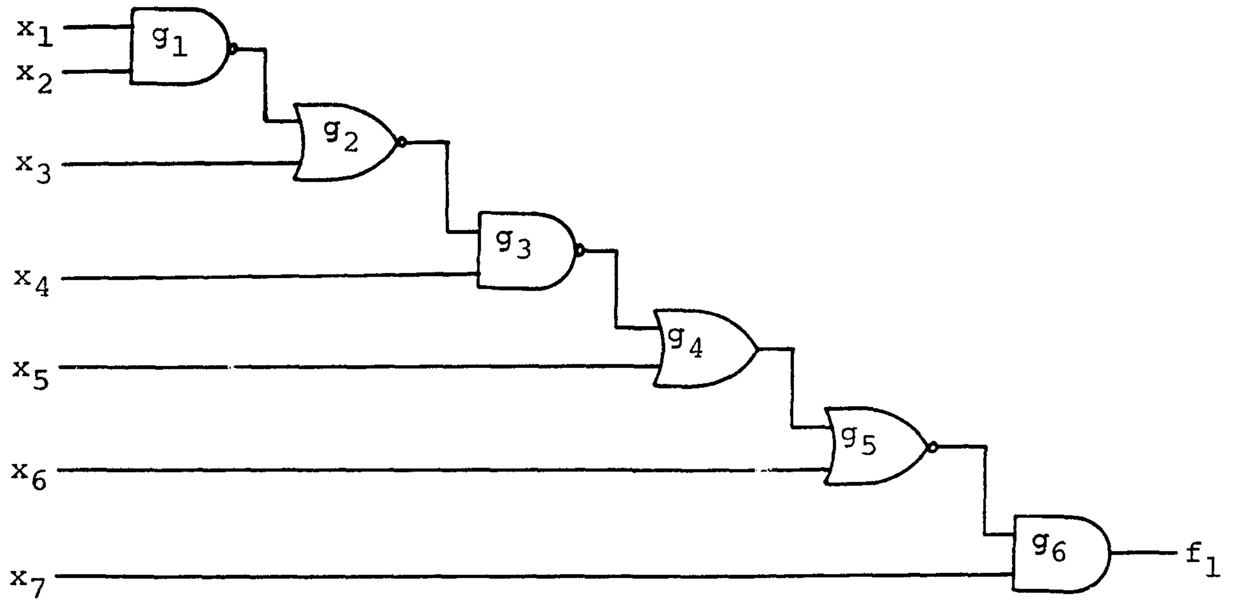
$$RN = \emptyset$$

$$SP = [x_1g_1g_2g_3g_4g_5g_6,\ x_2g_1g_2g_3g_4g_5g_6,\ x_3g_2g_3g_4g_5g_6,\ x_4g_3g_4g_5g_6,\ x_5g_4g_5g_6,\ x_6g_5g_6,\ x_7g_6]$$

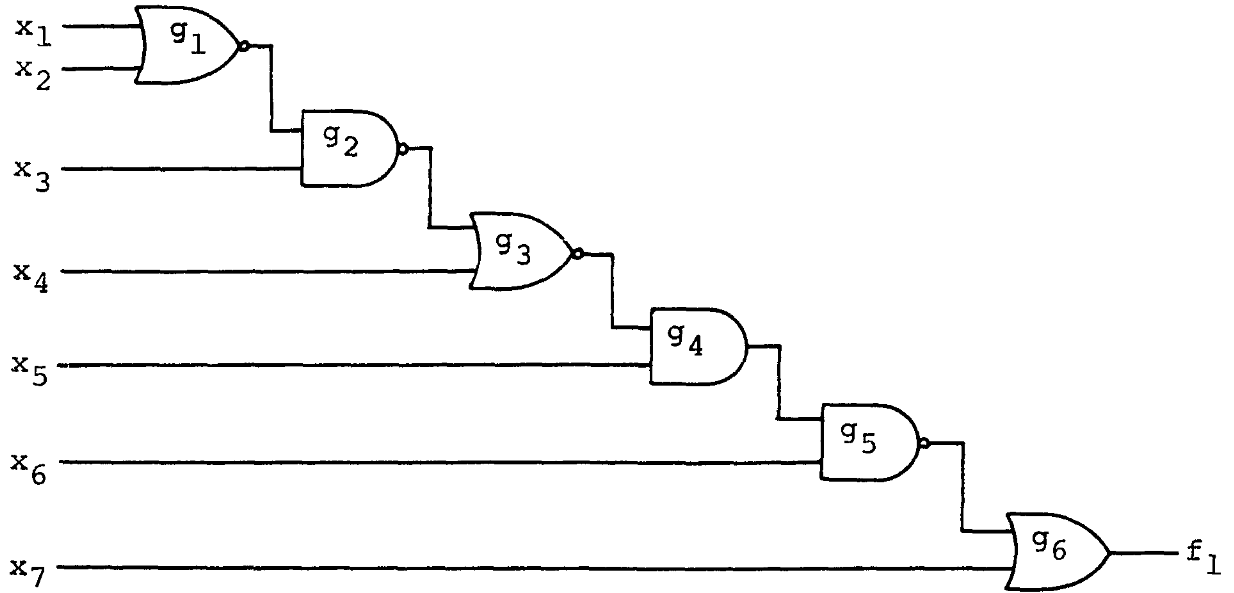
$$|H| = 2 \times (\ |SN| + |PIN|\ ) = 26$$

There exists an input pattern X_a = 1101001 in network A will detect 13 out of 26 faults, i.e., half of the total possible stuck-at-faults can be detected. All the signal paths in SP are sensitizing paths under X_a.

If we convert each gate in the network A to its logical dual gate (i.e., OR $\leftrightarrow$ AND, NOR $\leftrightarrow$ NAND), it is transformed



(a) Network A



(b) Network B

Figure 2.5
A Network with Special Topology

into a topologically equivalent network B. Network B possesses the interesting property that the complement of input pattern X_a($\overline{X}_a$ = 0010110) will detect the other half of the faults in H, and all the signal path in SP are sensitizing paths under $\overline{X}_a$.

The set of signal paths SP is said to be a <u>sensitizing set</u> under input pattern X_k if each element in SP is a <u>sensitizing path</u> under X_k.

<u>Theorem 2.3</u>: The set of signal paths SP of a network C is a sensitizing set under X_k if and only if the logic value at all the signal lines to each gate in C are the MSIP to each gate under X_k.

Proof: $\Leftarrow$ If the inputs to each gate are MSIP under X_k, then at least half of the faults in each gate will be detected and error information is propagated into the PON. So each signal path of the network C is a sensitizing path. Hence SP is a sensitizing set under X_k.

$\Rightarrow$ If SP is a sensitizing set of C under X_k, then each signal path is a sensitizing path under X_k. Therefore each input of a gate is sensitive to a fault under X_k. We conclude that the input to each gate in C is the MSIP under X_k.

In other word, Theorem 2.3 tells us that a network whose sensitizing path set SP is a sensitizing set under an input pattern X_k, if and only if half of the H in the network will be detected under X_k.

An <u>elementary network</u> EN is a network which is implemented with the ordinary logic NOT, OR, NOR, AND and NAND gates.

A <u>dual network</u> DN is defined as an EN which has two topologically equivalent modes A and B which are controlled by an external command and each node of SN in mode A is logical dual to corresponding node in mode B.

A new building block is proposed in order to implement a dual network.

A <u>programmable logic module</u> PLM is defined as a multiple input-single output combinational network in which each logic function performed by the module can be uniquely specified by a set of control signals.

The specific logic function of PLM for AND, OR, NAND and NOR gates are defined in Table 2.2 where C is the external command to control the mode of PLM. Each PLM has two modes of logic functions. Notice that the control signal, C, is assumed to be fault free in the subsequent discussion.

There are numerous ways to implement the desired PLMs. Figure 2.6 shows one of the implementation of PLM's for two input-one output gates. The general representations of PLM's are shown in Figure 2.7.

2.5 Design with Testability Enhancement

The use of PLMs as a building block provides us with the ability to implement a dual network having the following properties:

Module	Desired Logic Function When : C=0	C=1
PLM1	AND	OR
PLM2	OR	AND
PLM3	NAND	NOR
PLM4	NOR	NAND

Table 2.2 Four basic programmable logic modules.

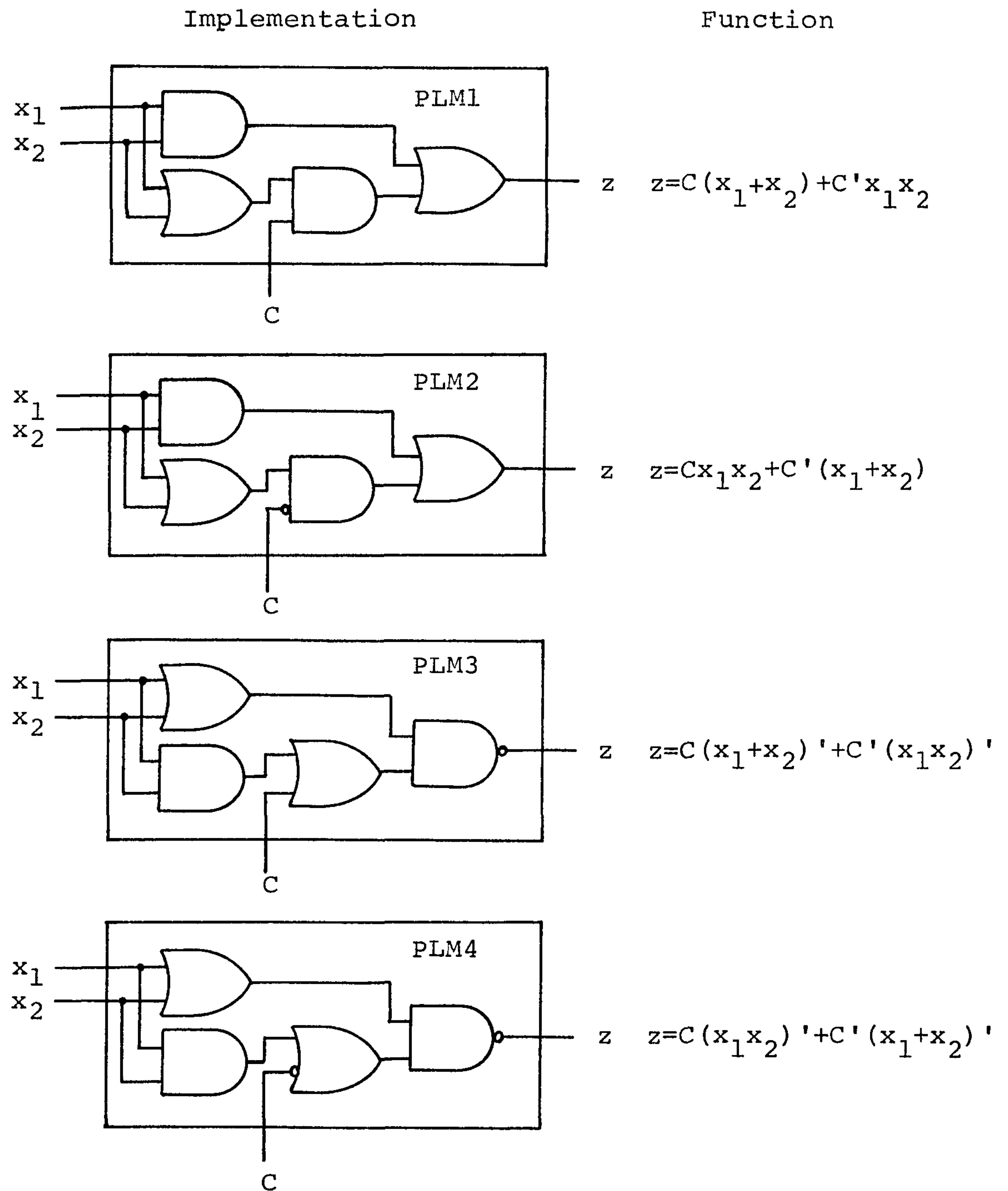
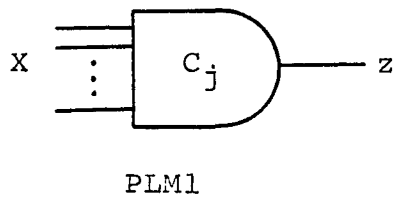
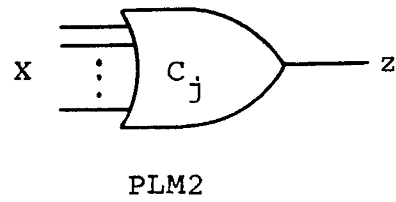
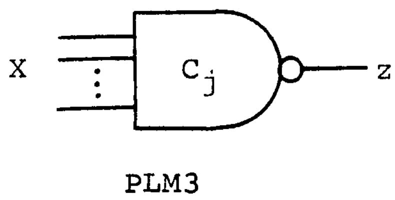
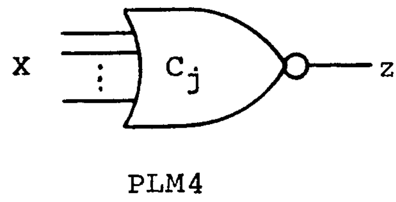


Figure 2.6
An Implementation of Four Basic 2-Input PLM's









C_j is the control signal of PLM's

Figure 2.7
A General Representation of Programmable Logic Modules (PLM)

Theorem 2.4: SP, the set of signal paths in a dual network DN is a sensitizing set under X_k in mode A if and only if the SP is also a sensitizing set under $\bar{X}_k$ in mode B.

Proof: ⇒ If the SP is a sensitizing set under X_k in mode A, then, by Theorem 2.3, all the signal lines in SL of DN can provide the MSIP to each signal node in mode A. By the dual nature of DN, the input pattern $\bar{X}_k$ will also provide the MSIP to each signal node in mode B. Again, by theorem 2.3, we conclude that SP is a sensitizing set under $\bar{X}_k$ in mode B.

⇐ It is obvious by the similar argument.

A pair of input pattern (X_a, X_b) is called a complement pair input pattern CPIP if $X_a = \bar{X}_b$.

Theorem 2.5: All the faults in H of a dual network DN can be detected by a CPIP(X_a, X_b) of DN if and only if the set of signal path SP in DN is a sensitizing set under the CPIP.

Proof: ⇐ If SP of DN is a sensitizing set under the CPIP (X_a, X_b), then X_a will provide the MSIP to each signal node in mode A which will detect half of the faults in H by Theorem 2.3. The other half of the faults in H will be detected by the input pattern X_b in mode B similarly. Hence the CPIP can detect all the faults in H of DN.

⇒ If all the faults in H of DN can be detected by the (X_a, X_b), then, from the dual nature of DN and the complementary relationship of X_a and X_b, each input pattern in CPIP

will detect half of the faults in H. So each input pattern in CPIP will provide the MSIP to each signal node in the corresponding mode of DN. Therefore, the SP of DN must be a sensitizing set under CPIP.

A dual network which satisfies the conditions of Theorem 2.5 is said to be restricted dual network (RDN)

From Theorem 2.5, we know that the set of faults H of an RDN can be partitioned into two distinct subsets H_a and H_b ($H_a \cap H_b = \emptyset$), such that H_a can be detected by X_a in mode A of RDN and H_b can be detected by X_b in mode B of RDN and $H_a \cup H_b = H$.

Example 1

Consider the RDN shown in Figure 2.8 which is the same circuit used in the previous example, except that we use PLM's for each signal node. H_a = [a/0, b/0, c/1, d/0, e/1, f/1, g/0, h/1, i/0, j/1, k/1, l/0, z/0] and H_b = [a/1, b/1, c/0, d/1, e/0, f/0, g/1, h/0, i/1, j/0, k/0, l/1, z/1] and $H = H_a \cup H_b$. The RDN is in its mode A when C =0, and RDN is in its mode B when C=1. Then X_a = 1101001 and X_b = 0010110 form the CPIP of RDN such that X_a detects H_a in mode A and X_b detects H_b in mode B. So that (X_a, X_b) forms the complete test pattern for the RDN.

Theorem 2.6: In a restricted dual network RDN, all the multiple faults in the signal lines of RDN will be detected by the CPIP(X_a, X_b).

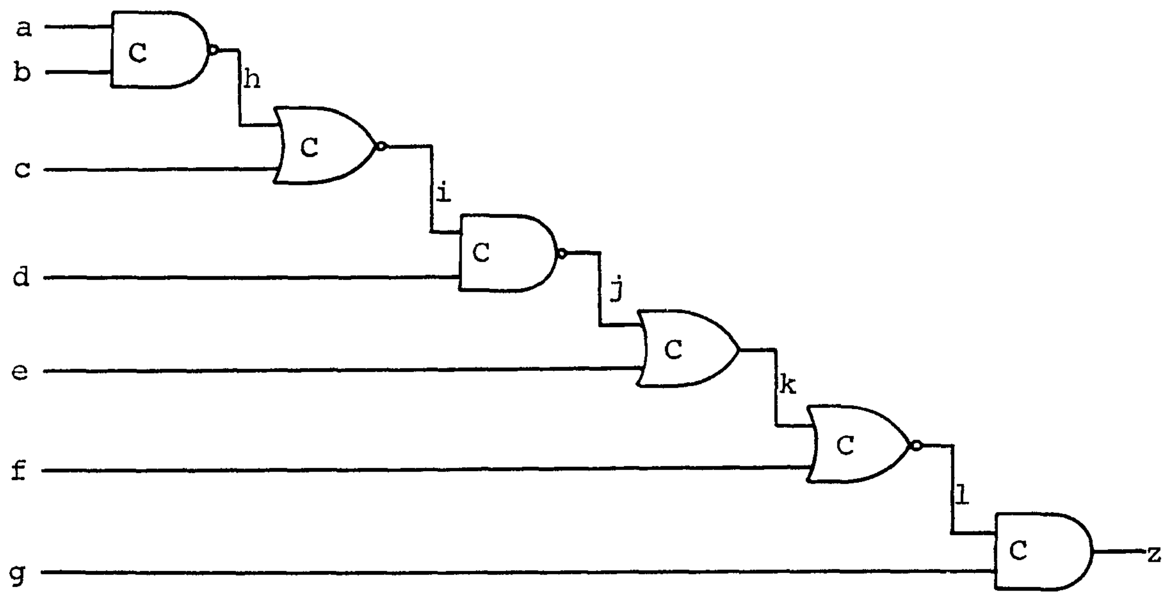


Figure 2.8
A Restricted Dual Network

Proof: Each signal path in SP of an RDN is a sensitizing path under the test input pattern X_a in mode A and X_b in mode B. In either case, any combination of faults in H will not make the path insensitive. Consequently, their existence will not mask any other fault and will not invalidate the test set(X_a, X_b). Therefore, they will be detected by either X_a or X_b.

It is well known that the existence of multiple faults will, in some cases, invalidate the complete test set of a given network generated by the conventional technique[14] under the single fault assumption. But in an RDN, this problem will not exist by its special property described in Theorem 2.6.

If we allow for each control signal of the PLM's in a dual network DN acting independently, then the number of modes of DN will be proportional to the distinct combinations of the control signal settings for the PLM's. In this case, the design will become excessively complex. To alleviate this difficulty, we assume that the DN has only two control commands C_1 and C_2, i.e., each control signal of PLM's has one of the two choices. Therefore, the DN has four modes of operations defined as follows:

1. The DN is in its normal mode if $C_1C_2 = 00$
2. The DN is in its Test A mode if $C_1C_2 = 01$
3. The DN is in its Test B mode if $C_1C_2 = 10$

4. The fourth mode of DN is never being used since we don't use the control command $C_1C_2 = 11$.

Let H_a and H_b be the subset of the fault set H in DN such that H_a can be detected by the test A mode and H_b can be detected by the test B mode. Notice that $H_a \cup H_b$ may not equal to H for an arbitrary DN.

In general, any combinational network can be converted to its DN by substituting the corresponding PLM's for each signal node. But not every dual network is RDN. This means that Theorem 2.5 will not be satisfied and the improtant characteristic of CPIP will not exist.

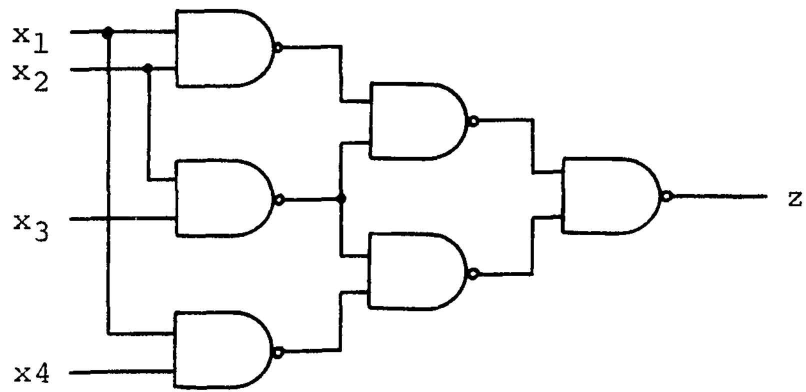
For some class of DN, the test A mode and B mode of DN will convert DN to a RDN. The following example illustrates the existance of this class of network.

Example 2 Consider the combinational network for $Z = (x_1x_2)'(x_2x_3)' + (x_1x_4)'(x_2x_3)'$ as shown in Figure 2.9(a). We convert it to a DN by using the PLM3 to replace each NAND gate and connect the control signals as shown in Figure 2.9 (b), in which

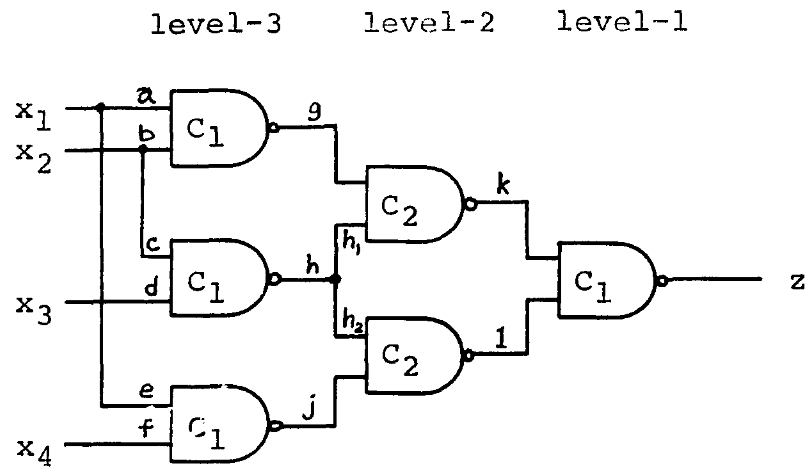
$$H_a = [a/0,\ b/0,\ c/0,\ d/0,\ e/0,\ f/0,\ g/1,\ h/1,\ h_1/1,\ h_2/1,\ j/1,\ k/0,\ l/0,\ Z/1]$$

$$H_b = [a/1,\ b/1,\ c/1,\ d/1,\ e/1,\ f/1,\ g/0,\ h/0,\ h_1/0\ h_2/0,\ j/0,\ k/1,\ l/1,\ Z/0]$$

For normal operation, we set $C_1C_2 = 00$, all the PLM3 function as NAND gates, and the network performs the normal operation. For testing, we set $C_1C_2 = 01$(test A mode), consequently, all



(a)



(b)

Figure 2.9
A Network Satisfies the Topological Requirement of Theorem 2.4

the gates in odd level become NAND and all the gates in even level becomes NOR and X_a = 1111 will detect H_a. Then, we set C_1C_2 = 10(test B mode), now, all the gates in odd levels become NOR and all the gate in even level become NAND, and X_b = 0000 will detect H_b. By observation we find $H_a \cup H_b$ = H and $H_a \cap H_b = \emptyset$. So the test A and B modes convert the DN to an RDN, and the CPIP is easy to produce.

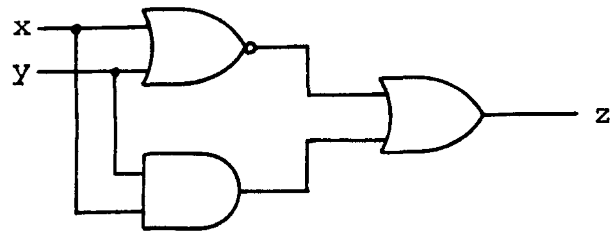
For this class of networks, we can convert them to a specified structure which will satisfy the requirement of Theorem 2.5 during the testing modes by properly setting the control signals of PLM's. The choice of control signal settings for this task is therefore important. The relation of control signals between the consecutive signal nodes are established in Table 2.3.

For a restricted class of networks which contain fanout nodes and/or reconvergent nodes, it will, in some cases, be impossible to convert them to satisfy the topological constraint posed by Theorem 2.5 just by adjusting the settings of control signals for PLM's. These difficulties can be classified into two cases as illustrated by the following examples:

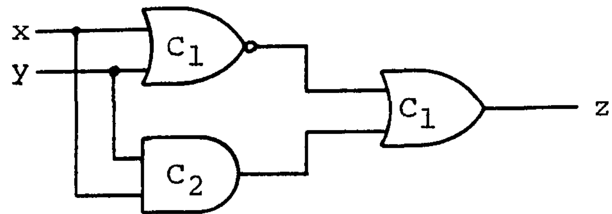
Example 3 Consider the Boolean expression Z = x'y' + xy as shown in Figure 2.10(a). It is converted to Figure 2.10 (b) by replacing each gate with the proper PLM's. It is impossible to construct the circuit to satisfy the connection

Module	Its Successor	The Relation of Their Control signal
PLM1	PLM2/PLM4	Different
	PLM1/PLM3	Same
PLM2	PLM2/PLM4	Same
	PLM1/PLM3	Different
PLM3	PLM2/PLM4	Same
	PLM1/PLM3	Different
PLM4	PLM2/PLM4	Different
	PLM1/PLM3	Same

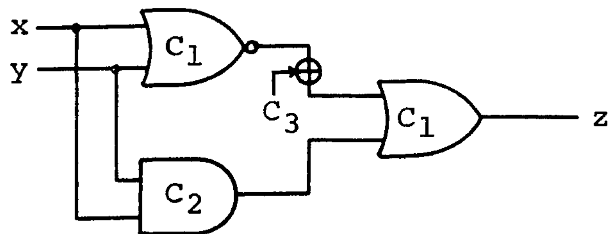
Table 2.3 The general rule of determining the control signal of the PLM's.



(a) NN



(b) DN



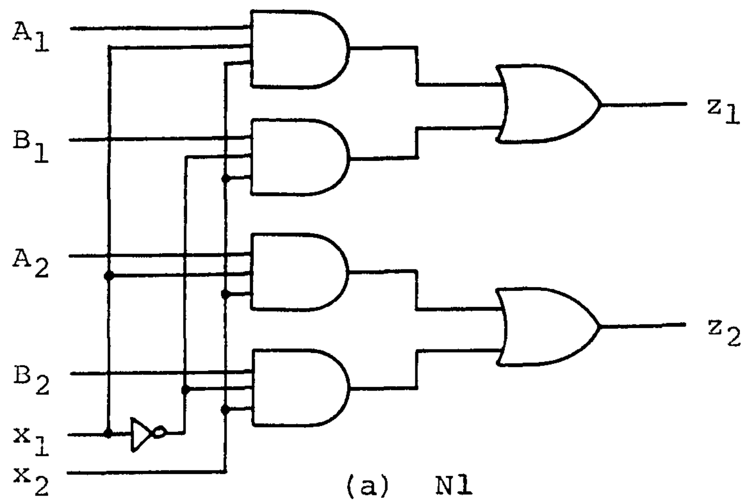
(c) RDN

Figure 2.10
A Typical Case-1 Network

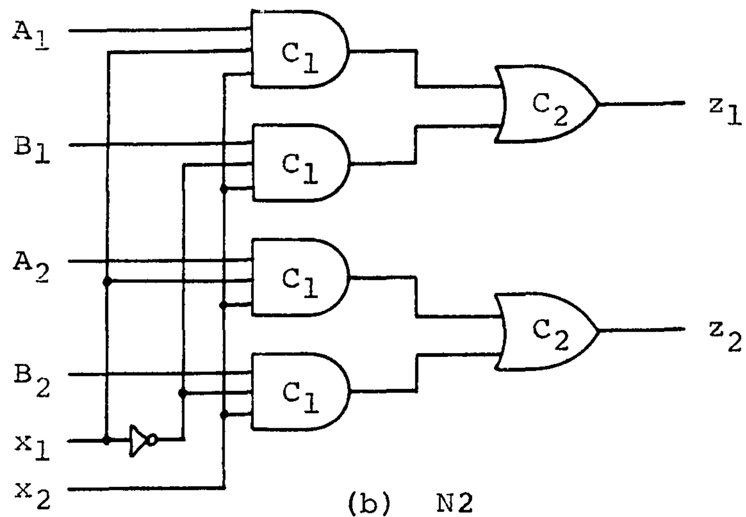
requirement of Theorem 2.5 by the technique of setting control signals. The only solution to this problem is to insert a controlled signal inverter(CSI) at the troublesome signal lines as shown in Figure 2.10(c) in order to make the network satisfy the conditions of Theorem 2.5. The CSI is actually an exclusive-or gate in which one of the input is used as level control signal C_3. Then the control commands $C_1C_2C_3$ = 011 and 101 will convert the DN to RDN such that X_a = 00 will detect H_a in H and X_b = 11 will detect H_b in H and $H_a \cup H_b = H$. The control command $C_1C_2C_3$ = 000 will bring the network back to its normal function.

<u>Example 4</u> Consider the networks shown in Figure 2.11, N_1 is a dual 2-line to 1-line data selector and multiplexer. N_2 is the dual network by substituting the gates in N_1 with the proper PLM's. The control signal setting is established based on Table 2.3 as shown. It is clear that the NOT gate disables the DN to satisfy the requirement of Theorem 2.5. If we replace this NOT gate with CSI in which the additional input is used as level control signal C_4, then the control setting $C_1C_2C_4$ = 010 and 100 will make N_1 to have the property of RDN such that X_a = 111111 and X_b = 000000 with detect the set of faults H in N_3. For normal network operation, we set $C_1C_2C_4$ = 001.

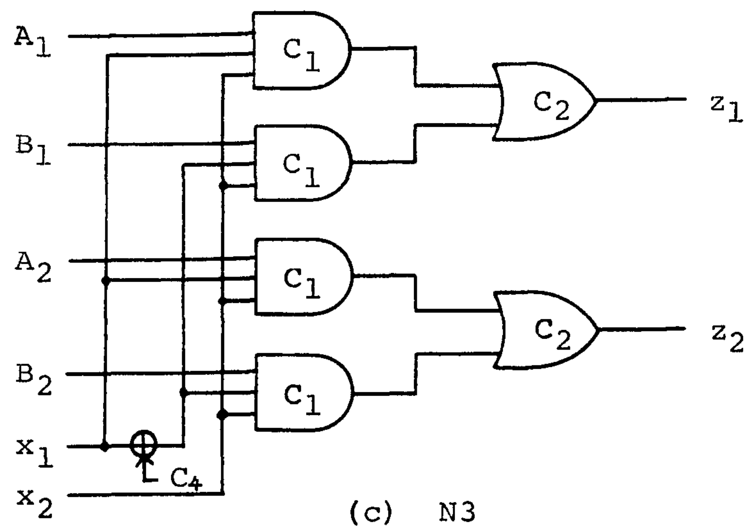
These two examples show that the problem in <u>case 1</u> can be



(a) N1



(b) N2



(c) N3

Figure 2.11
A Typical Case-2 Network

solved by the insertion of CSI in the signal line and the use of control signal C_3. And the problem in case 2 can be solved by the use of CSI instead of NOT gate and control signal C_4. We then conclude the following theorem.

Theorem 2.7 The fault set H of any dual network DN can be detected with two test input patterns if the CSI and two additional control signals C_3 and C_4 are used.

Proof: Although the DN now has four control signals $C_1C_2C_3C_4$, it still operate in three operational modes. i.e.,

(1) $C_1C_2C_3C_4$ = 0001 for normal mode

(2) $C_1C_2C_3C_4$ = 0110 for test A mode

(3) $C_1C_2C_3C_4$ = 1010 for test B mode

Using the CSI's, two causes of difficulties can be removed as illustrated in the above example. It means that the control commands $C_1C_2C_3C_4$ = 0110 and 1010 will make the DN to preserve the property of RDN. Then there exists a CPIP which will detect the fault set H of DN.

A dual network implemented with PLM's and CSI which satisfying the conditions of Theorem 2.5 is called a modified restricted dual network MRDN. Notice that a MRDN has a maximum of four control signals C_1, C_2, C_3 and/or C_4. It follows then the fault set H in any elementary network can be detected by two test input patterns since we can always convert it to its MRDN.

A systematic design algorithm is presented in the

following section based on the application of the theorems derived in this section. It allows us to design any EN to have its own CPIP in its MRDN.

2.6 Systematic Design Algorithm for Combinational Networks

We treat a combinational network as a directed graph in which each node is belong to the set [OR, NOR, AND, NAND, NOT, PI, PO] and each directed edge is the signal line between the node. Some definitions are introduced to facilitate the discussion of the systematic design algorithm.

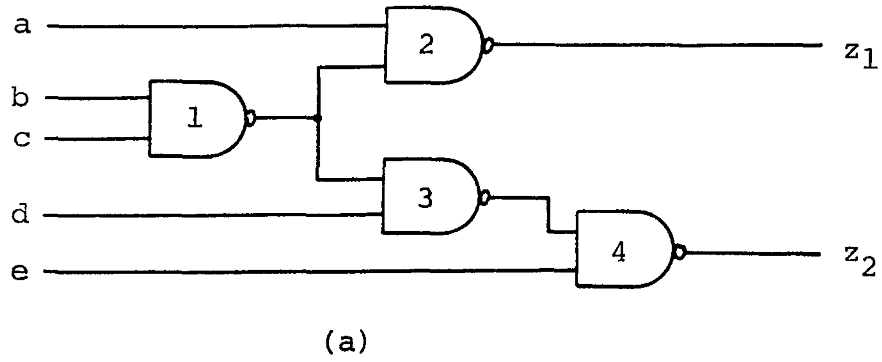
A node G_i is said to be a predecessor node of a node G_j if there exists a signal line from node G_i to node G_j. Conversely, G_j is the successor node of G_i.

A fan out node(FN) has more than two successor nodes, all signal node other than FN have only one successor node. The PI node has no predecessor and PO node has no successor.

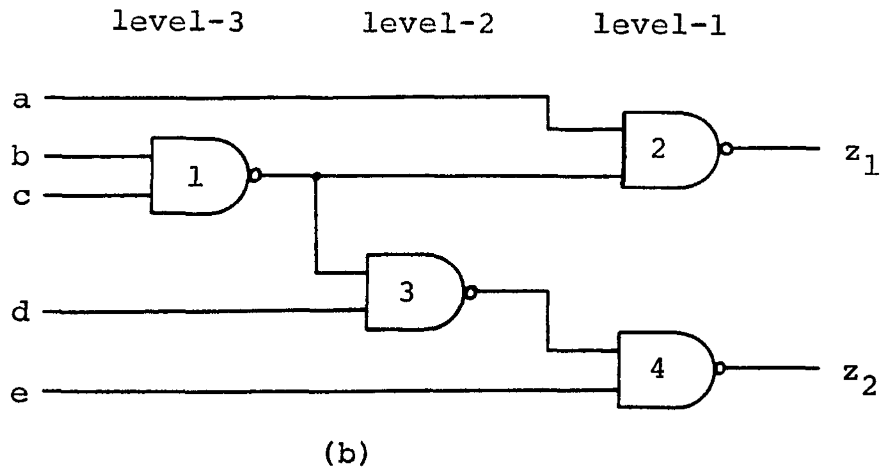
Let the level of a node G be denoted as L(G). Then L(G) = 1 if PO is its only successor node. Otherwise, L(G) = 1 + the maximum level of the successors of node G.

Consider the circuit shown in Figure 2.12, where $Z_1 Z_2$ are PO. Then $L(G_2) = L(G_4) = 1$ because their node outputs are POs. $L(G_3) = L(G_4) + 1 = 2$ and $L(G_1) = 1 + \max [L(G_2), L(G_3)] = 3$ according to the definition.

Let the set of all OR, NOR, AND, NAND gates in the level i be denoted as A_i and all the signals lines incident into A_i is denoted as the set SA_i. Let the set of all NOT gates in the level i be denoted as B_i and all the signal



(a)



(b)

Figure 2.12
A Network Illustrates the Definition of L(G)

lines incident into B_i as the set SB_i.

Let each node in SN U PIN of a network have a control signal(CS). The CS of NOT gate serves the dummy reference and the CS of PI will set the logic value of PI and define the CPIP of the modified dual network.

Then, an algorithm for the systematic design procedure of the combinational network is generated from the property of the proposed PLM's and the introduced definitions. The flow chart is shown in Figure 2.13 and the details are described below.

Systematic Design Algorithm

Step 1. Determine the level of each signal node of a given combinational network and reconstruct the network to a controllable dual network by using PLM1, PLM2, PLM3 and PLM4 to replace the AND, OR, NAND and NOR gate respectively. Denote the control signal of the predecessor node as CP and the reference control signal as RS.

Step 2. Let I = 1, K = 0 and J = the maximum level of the network and CS of PLM1/PLM3 in level-1 be C_1 and CS of PLM2/PLM4/NOT in level-1 be C_2 where C_2 is the complement of C_1.

Step 3. Let SL = SA_i, then check if SL = Ø. If it is, then go to step 10; otherwise, go to next step.

Step 4. Pick one signal line of SL and trace back to its predecessors node and delete the line from SL. If the

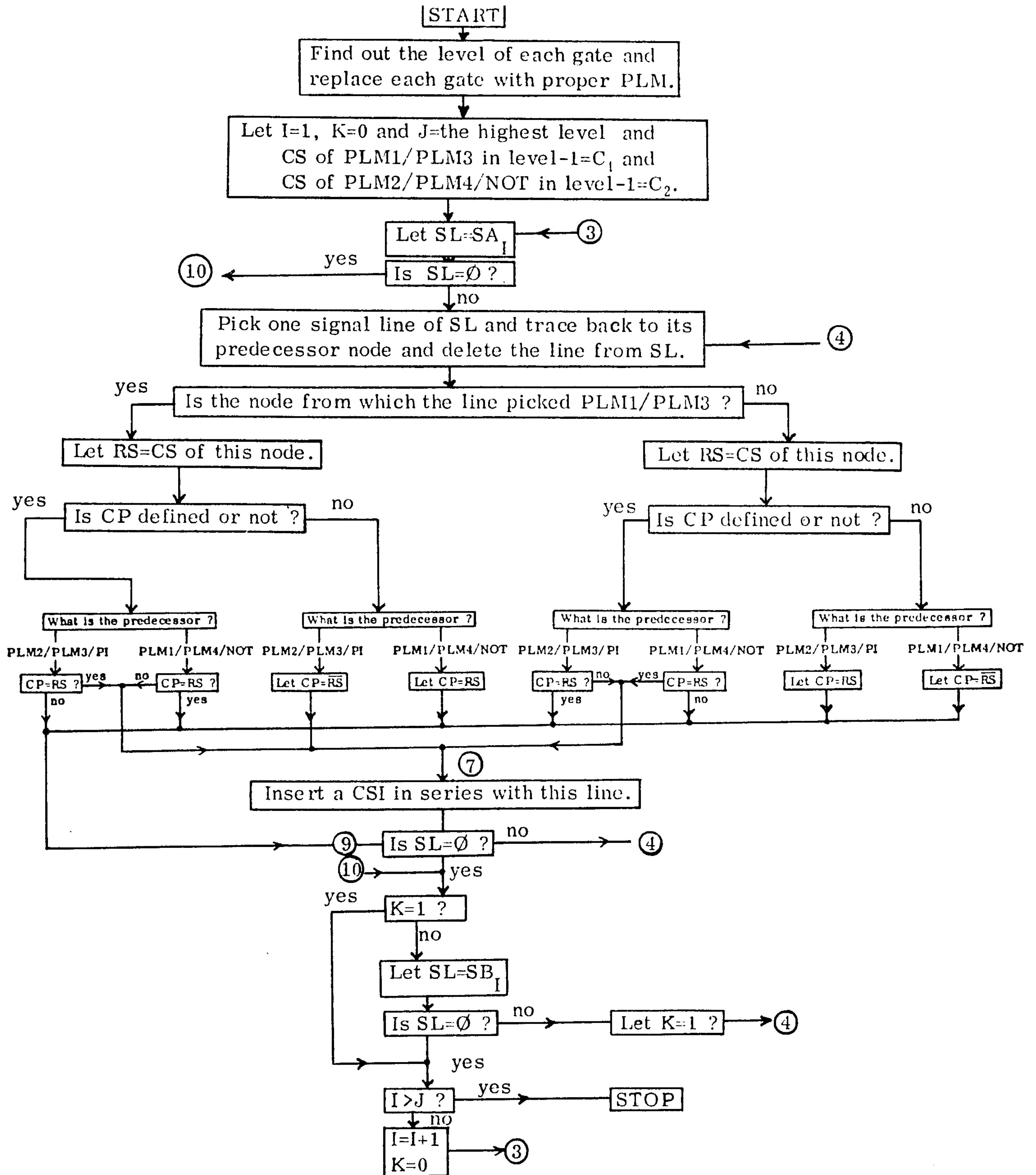


Figure 2.13
Flow Chart of The Systematic Design Algorithm for Combinational Network

node from which the signal line is picked is PLM1/PLM3, go to step 5; otherwise, go to step 12.

Step 5. Let RS be equal to the control signal of this PLM1/PLM3 node. Then check to see whether the control signal of its predecessor node is defined or not. If yes, go to step 6; otherwise, go to step 8.

Step 6. If the predecessor node is:

(1) PLM2/PLM3/PI, check if CP = RS. If it is, go to step 7; otherwise, go to step 9.

(2) PLM1/PLM4/NOT, check if CP = RS. If it is, go to step 9; otherwise, go to step 7.

Step 7. Insert a CSI in series with this input line. Then go to step 9.

Step 8. If the predecessor node is:

(1) PLM2/PLM3/PI, set CP to be the complement of RS.

(2) PLM1/PLM4/NOT, set CP to be equal to RS, then go to to step 9.

Step 9. Check to see whether the set SL is an empty set. If yes, go to step 10; otherwise, go back to step 4.

Step 10. Test for K = 1. If yes, go to step 11; otherwise, let SL = SB_i and check for SL = $\emptyset$. If it is, go to step 11; otherwise, set K = 1, then go back to step 4.

Step 11. Examine if I is larger than J. If yes, the procedure is completed; otherwise, let I = I + 1, and then go back to step 3.

node from which the signal line is picked is PLM1/PLM3, go to step 5; otherwise, go to step 12.

Step 5. Let RS be equal to the control signal of this PLM1/PLM3 node. Then check to see whether the control signal of its predecessor node is defined or not. If yes, go to step 6; otherwise, go to step 8.

Step 6. If the predecessor node is:

(1) PLM2/PLM3/PI, check if CP = RS. If it is, go to step 7; otherwise, go to step 9.

(2) PLM1/PLM4/NOT, check if CP = RS. If it is, go to step 9; otherwise, go to step 7.

Step 7. Insert a CSI in series with this input line. Then go to step 9.

Step 8. If the predecessor node is:

(1) PLM2/PLM3/PI, set CP to be the complement of RS.

(2) PLM1/PLM4/NOT, set CP to be equal to RS, then go to to step 9.

Step 9. Check to see whether the set SL is an empty set. If yes, go to step 10; otherwise, go back to step 4.

Step 10. Test for K = 1. If yes, go to step 11; otherwise, let $SL = SB_i$ and check for $SL = \emptyset$. If it is, go to step 11; otherwise, set K = 1, then go back to step 4.

Step 11. Examine if I is larger than J. If yes, the procedure is completed; otherwise, let I = I + 1, and then go back to step 3.

Step 12. Let RS be equal to the control signal of this PLM2/PLM4/NOT. Then check to see whether the control signal of its predecessor node is defined or not. If yes, go to step 13; otherwise, go to step 14.

Step 13. If the predecessor node is:

(1) PLM2/PLM3/PI, check if CP = RS. If it is, go to step 9; otherwise go to step 7.

(2) PLM1/PLM4/NOT, check if CP =RS. If it is, go to step 7; otherwise go to step 9.

Step 14. If the predecessor node is

(1) PLM2/PLM3/PI, set CP to be equal to RS.

(2) PLM1/PLM4/NOT, set CP to be the complement of RS.

Notice that the CS of all the inserted CSI is C_3. After the process is completed, it is possible that a portion of a signal path in the modified dual network has the CSI adjacent to a NOT gate as shown in Figure 2.14(a) which is actually equivalent to the CSI with C_4 as the control signal where $C_4 = \bar{C}_3$ as shown in Figure 2.14(b). If this is the case, then use the later structure to replace the former one. Now the procedure is actually completed. Then the fault set H of the network with control signals $C_1C_2C_3C_4$ can be detected with the most sensitive input pattern which is generated by the design algorithm and its complement.

Fault Detection Procedure

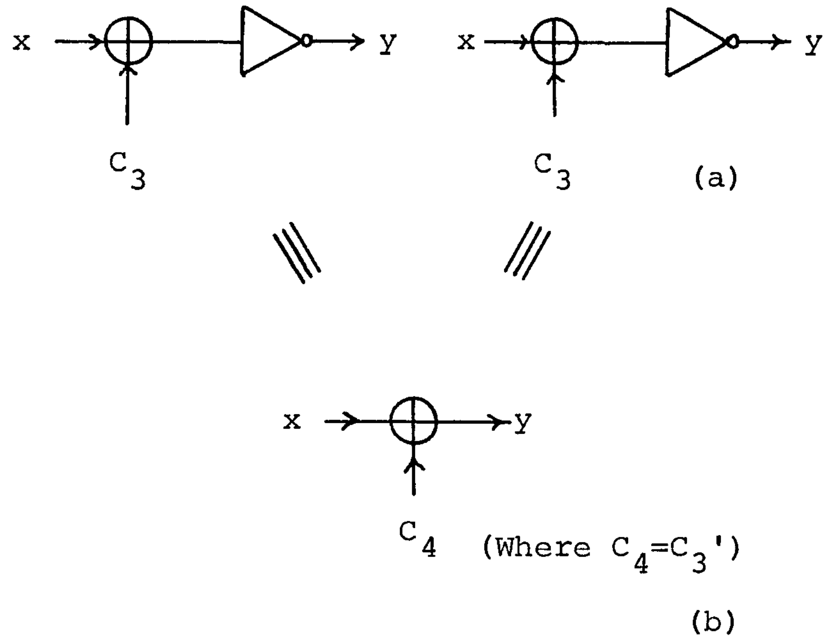


Figure 2.14
The Relation between the Control Signals C_3 and C_4

The fault detection procedure for the translated circuit is easy to follow since the MSIP is already generated by the systematic algorithm. Two tests can detect all the stuck faults in H of the given network.

<u>Test 1</u> Set $C_1C_2C_3C_4$ = 0110 and apply the MSIP X_a. If the observable primary output of PLM1/PLM2 is 1 and that of PLM3/PLM4 is 0, then the network does not have any fault in H_a of the network. Otherwise, it has at least one stuck-at-fault in H_a.

<u>Test 2</u> Set $C_1C_2C_3C_4$ = 1010 and apply the MSIP $X_b (X_b = \bar{X}_a)$. If the observable primary output of PLM1/PLM2 is 0 and that of PLM3/PLM4 is 1, then the network does not have any fault in H_b of the network. Otherwise, it has at least one stuck-at fault in H_b.

Since $H_a \cup H_b = H$ which is the set of all possible fault in the network, once the network passes these two tests, it will be guaranteed to be free of stuck faults. Then set $C_1C_2C_3C_4$ = 0001 to perform the network's normal function.

In the subsequent discussion, some examples are shown to demonstrate the proposed systematic design and detection algorithm.

<u>Example 5</u> Consider the network N1 shown in Figure 2.15(a), whose FN = [x_3, g_1, g_2], RN = [g_7, g_8] and H = [a/0,1, b/0,1, c/0,1, d/0,1, e/0,1, f/0,1, g/0,1, g_1/0,1, g_2/0,1, h/0,1, h_1/0,1, h_2/0,1, j/0,1, k/0,1, l/0,1, m/0,1, n/0,1, p/0,1].

Notice that the existence of FN(fanout node) and/or RN (reconvergent node) will possibly cause some stuck-at-fault undetectable, e.g., g/1 is undetectable in N1. This problem can be solved by using the proposed design technique. We first define the level of each gate according to definition and determine the sets SA_i and SB_i for all $1 \leq i \leq 3$, then convert N1 to an easily detectable network MRDN1 as shown in Figure 1.15(b) by applying the systematic procedure. We observe that the signal line indicated by the dotted box can be replaced by the corresponding box in Figure 1.15(c). The test input pattern $X = x_1x_2x_3x_4x_5 = C_2C_1C_1C_1C_1$ is generated. Two tests are applied as follows:

Test 1 Set $C_1C_2C_4 = 010$, and apply the test pattern X_a = 10000. If the PO, $Z_1Z_2 = 10$, then the network does not have any fault in H_a, where H_a = [a/0, b/1, c/1, d/1, e/1, f/1, g/0, g_1/0, g_2/0, h/1, h_1/1, h_2/1, j/0, k/0, l/0, m/0, n/0, p/1]. Otherwise, it has at least one fault in H_a and fault is detected.

Test 2 Set $C_1C_2C_4$ =100, and apply the test pattern X_b = 01111. If the PO, $Z_1Z_2 = 01$, then the network does not have any fault in H_b, where $H_b = H - H_a$. Otherwise, it has at least one fault in H_b and fault is detected.

If the network passes these two tests, then it is guaranteed to be free of stuck fault in H. We set $C_1C_2C_4$ =001, put the network back to its normal function.

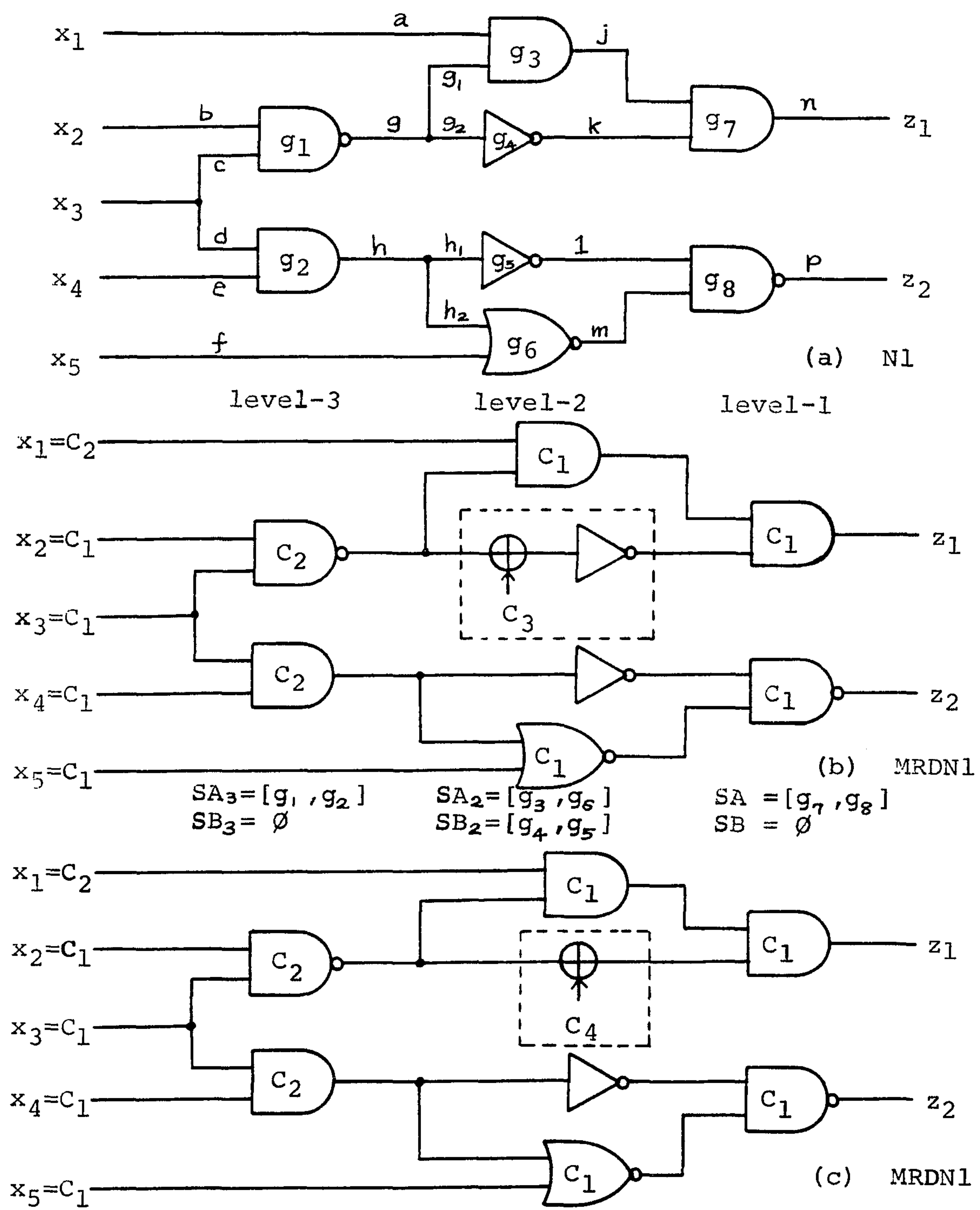


Figure 2.15
Network of Example 5

Notice that the undetectable g/1 fault in the conventional design network can now be detected by the proposed design technique.

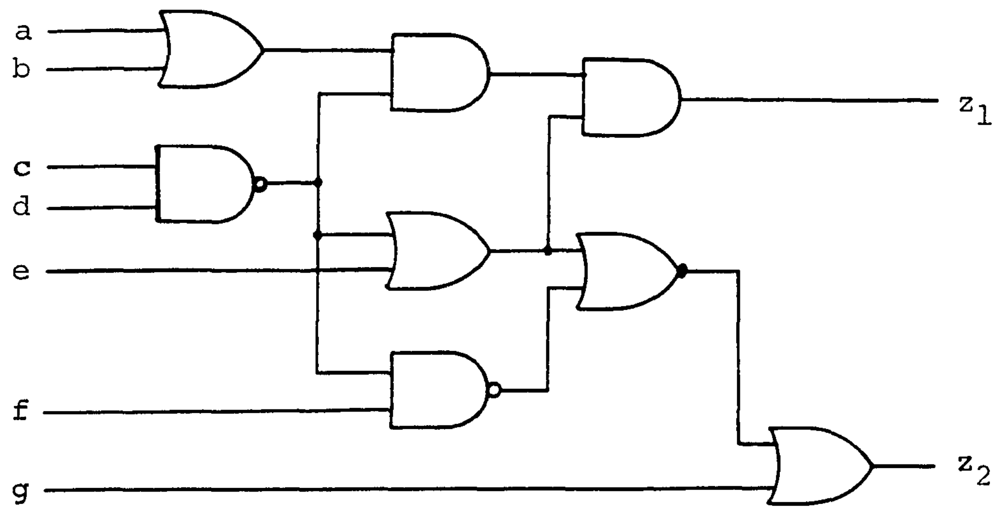
Example 6 Consider the network N2 shown in Figure 2.16(a). We first define the level of each gate and the set SA_i, SB_i for all $1 \leq i \leq 4$. Then an easily detectable network MRDN2 is generated as shown in Figure 2.16(b) by following the systematic design procedure. The generated test input pattern X = abcdefg = $C_2C_2C_1C_1C_2C_2C_2$ can detect the fault set H of N2 as follows:

Test 1 Set $C_1C_2C_3$ = 011 and apply the test input pattern X_a = 1100111, If the PO, Z_1Z_2 = 11, then the network does not have any fault in H_a where H_a = [a/0, b/0, c/1, d/1, e/0, f/0, g/0, h/0, h_1/0, h_2/0, h_3/0, i/0, j/0, j_1/0, j_2/0, k/1, l/1, m/0, n/0, Z_1/0, Z_2/0]. Otherwise, there exists at least one fault in H_a and fault is detected.

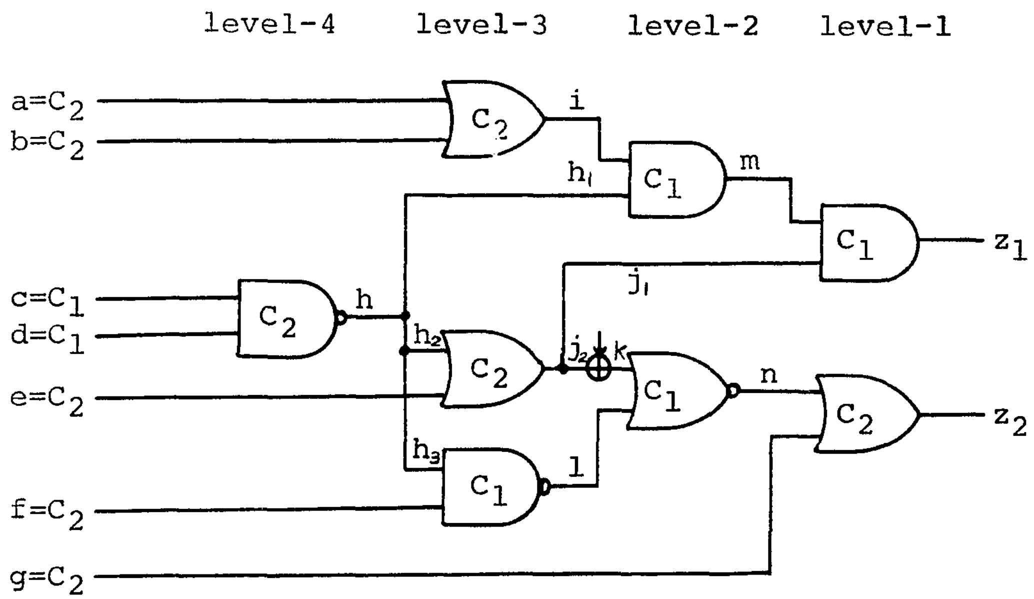
Test 2 Set $C_1C_2C_3$ = 101 and apply the test input pattern X_b = 0011000. If the PO, Z_1Z_2 = 00, then the network does not have any fault in H_b where $H_b = \bar{H}_a$ and $H_a \cup H_b = H$. Otherwise, there exists at least one fault in H_b and fault is detected.

Once the network passes the two tests, it is guaranteed that there is no stuck-at-fault in H, and the network can back to its normal operation mode by setting $C_1C_2C_3$ = 000.

The above discussions show that the proposed systematic design technique is easy to follow and is useful for designing



(a) N2



(b) MRDN2

Figure 2.16
Network of Example 6

an easily detectable combinational network for multiple fault detection.

CHAPTER 3

SYNCHRONOUS SEQUENTIAL NETWORK AND TESTABILITY ENHANCEMENT

3.1 Introduction

The derivation of fault detection test for sequential networks, in general, is a complex process due to the fact that the signals of the internal memory feedback lines are usually neither observable nor accessible. Therefore, the test for a sequential network, if it can be derived, turns out to be a long sequence of input patterns.

Methods for deriving the test input sequence for detecting faults in a sequential network have been extensively investigated [6], [31], [43], [44], [28], [21], [24], [25], [26], [32], [18], [22], [15], [12]. The existing techniques are designed to generate the test sequence, after the network is designed. During the functional design stage, however, the testability is, in general, not considered. As a result, the conventional techniques involve complex algorithms requiring, in general, a large amount of computer time and memory which makes them impractical for real time applications.

In this chapter, a new approach to the design of synchronous sequential networks is developed, which includes the test generation as a design requirement in addition to the functional specifications for the network. The network designed according to this new approach can be

tested with four test input patterns which will make the real time testing of the network realistic.

For the sake of completeness, some basic definitions on sequential networks are first reviewed in section 3.1.

In section 3.2, some conventional techniques for test generation are surveyed and the fault model of the four basic flip-flops are introduced to facilitate subsequent discussions.

Some specific structural aspects of a synchronous sequential network are investigated. In section 3.4, certain topological constraints are imposed on networks to allow us to develop a real time fault detectable synchronous sequential network design, in a sense that the faults in the network can be detected within few clock pulse periods.

A controllable memory module (CMM) is proposed which enables us to modify a given synchronous sequential network to satisfy the topological constraints developed in section 3.4. The testability of the modified network is shown to be greatly enhanced.

Finally, a systematic design algorithm and fault detection procedure are developed. The algorithm will convert any synchronous sequential network (implemented with the ordinary logic gates and the four basic master-slave flip flops) to a specific structure which will

satisfy the topological requirements for real time fault detection during the test modes.

The modified synchronous sequential network (MSSN) will require at most, six external control signals, and four tests are sufficient to detect the stuck-at-faults and malfunctions in the MSSN, independent of the complexity of the combinational block and the number of flip-flops in the network.

3.2 Notation and Definitions

A <u>sequential network</u> is defined to be a logical network with memory in which the present outputs depend on the previous states of the memory and/or the present primary inputs, and the memory state transitions depend on the previous states and present primary inputs of the network.

Any sequential network structure can be partitioned into two parts, namely, a combinational network and a block of memory elements. The canonical representation of sequential network is shown in Figure 3.1. The network has a finite number of <u>accessible</u> input variables $(x_1, x_2, --, x_n)$, a finite number of observable output terminals $(z_1, z_2, --, z_m)$, and a set of state variables $(y_1, y_2, --, y_q)$ representing the logical states of memory elements. A sequential network can be characterized mathematically be a quintuple $[PI, PO, S, \delta, \lambda]$ as follows:

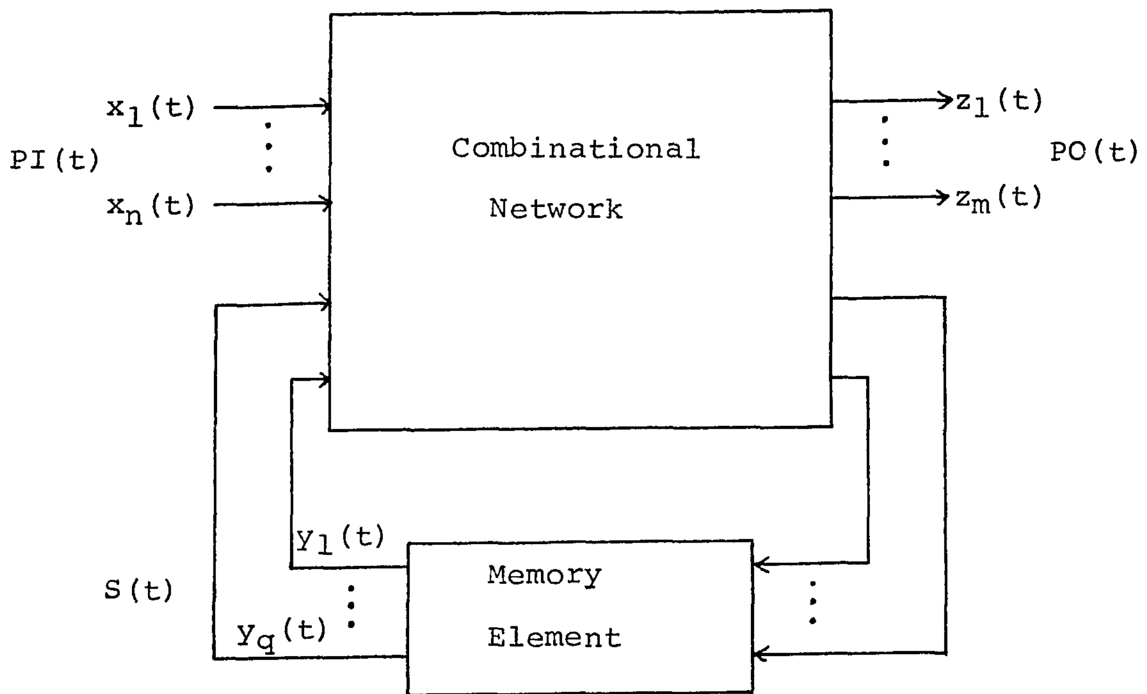


Figure 3.1
Canonical Model of Sequential Network

1. PI(t) is a set of primary (accessible) input patterns at time t such that $PI(t)=[X_0(t), X_1(t),..., X_k(t),..., X_{2^n-1}(t)]$ where $X_k(t)=[x_1(t),...,x_n(t)]_k$ and k is the equivalent binary integer defined by the input vector $X_k(t)$.
2. PO(t) is a set of primary (observable) output patterns at time t such that $PO(t)=[Z_0(t), Z_1(t),...,Z_k(t),..., Z_{2^m-1}(t)]$ where $Z_k(t)=[z_1(t),..., z_m(t)]_k$ and k is the equivalent binary integer defined by the output vector $Z_k(t)$.
3. S(t) is a set of state patterns at time t such that $S(t)=[S_0(t), S_1(t),..., S_k(t),..., S_{2^q-1}(t)]$ where $S_k(t) =[y_1(t),..., y_q(t)]$ and k is the equivalent binary integer defined by the state vector $S_k(t)$.
4. δ is the state transition function which maps an element in PIxS into an element in S, i.e., $S(t+1)= \delta[S(t), PI(t)]$.
5. λ is the output function which maps an element in PIxS into an element in S. If $Z(t)=\lambda[S(t), PI(t)]$, the network is called Mealy network. On the other hand, If $Z(t)= \lambda[S(t)]$, the network is referred to as a Moore Network.

The tramsition functions and output functions can be described either by a state table, or by a state diagram.

A <u>state table</u> consists of 2^n input columns, one for

each present input pattern, and 2^q state rows, one for each present state. For each combination of present input $X_i(t)$ and present state $S_j(t)$ (for all $0 \leq i \leq 2^n-1, 0 \leq j \leq 2^q-1$), the corresponding entry $[S_K(t+1), Z_1(t)]$ specifies the next state which the network will assume and the present output to be generated as shown in Fig. 3.2(a).

In a <u>state diagram</u>, each node corresponds to a state of the network, and each directed edge, to a state transition and output function. Each directed edge is labeled by the input which causes the transition, and by the output to be generated as shown in Fig. 3.2(b).

The state table and state diagram are, therefore, equivalent since both of them characterize the same information on network behavior.

A sequential network can be categorized as <u>synchronous</u> or <u>asynchronous</u> depending on whether the state transitions of memory elements are synchronized to a single clock source or not.

For a synchronous sequential network, the next state $S(t+1)$ and output $PO(t)$ are determined from the input values $PI(t)$ that are present when a clock pulse appears.

In asynchronous sequential network, the inputs are allowed to change only when the network has reached a stable state and the state transitions are initiated by changes in the input. Since the asynchronous network is not controlled by a clock, the transient conditions are generally

present following the change of inputs or state variables. In this process, a race is occured if two or more state variables need to change simultaneously. A race is said to be a critical race if the final stable state reached depends on the order in which the state variables change. Since this may lead a network to an undesired state, a critical race condition should be eliminated during the design stage.

In the subsequent discussion, we will limit our investigation to synchronous sequential networks since most digital systems of practical importance are operated in synchronous mode.

The memory element used in the synchronous sequential network is a bistable logic device called a flip-flop.

A master-slave flip-flop is a flip-flop whose state transitions occur in synchronism with the completion of the applied clock pulse. It is sometimes called a dual-rank flip-flop.

There are four types of master-slave flip-flops. Their logic function are defined as follows :

1. Set-Reset flip-flop, SRFF, is a flip-flop with two data inputs, R and S, and a clock input such that its state transitions satisfy the excitation requirement defined in Table 3.1(a).
2. Delay flip-flop, DFF, is a flip-flop with only one data input, D, and a clock input such that its state trans-

State change from Q(t)	to Q(t+1)	Required inputs S	R
0	0	0	-
0	1	1	0
1	1	-	0
1	0	0	1

(a) S-R flip-flop

State change from Q(t)	to Q(t+1)	Required input D
-	0	0
-	1	1

(b) D flip-flop

State change from Q(t)	to Q(t+1)	Required inputs J	K
0	0	0	-
0	1	1	-
1	1	-	0
1	0	-	1

(c) J-K flip-flop

State change from Q(t)	to Q(t+1)	Required input T
0	0	0
0	1	1
1	1	0
1	0	1

(d) T flip-flop

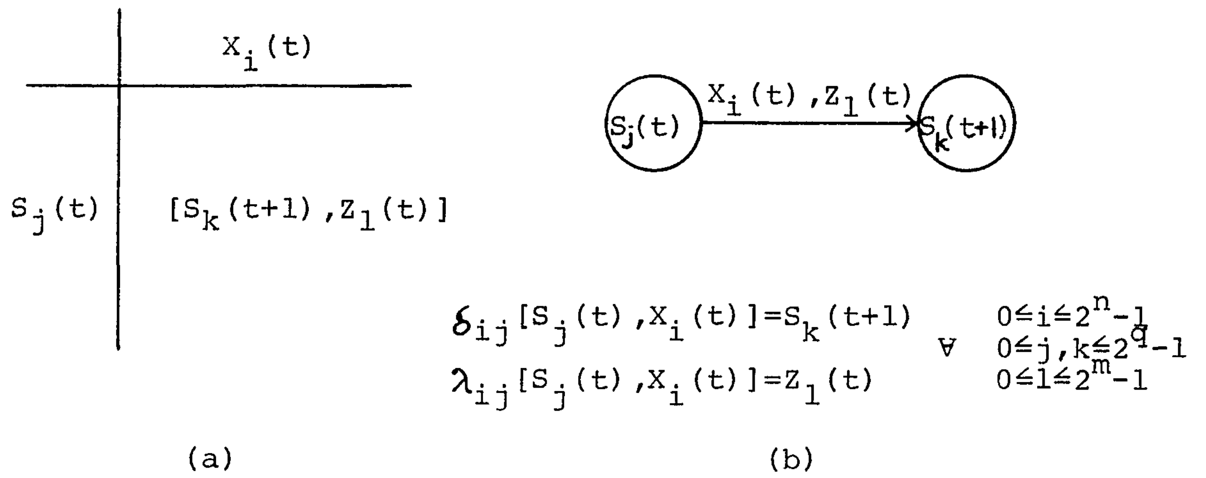
Table 3.1 Excitation requirement of the four basic flip-flops.

itions satisfy the excitation requirement defined in Table 3.1(b).

3. J-K flip-flop, JKFF, is a flip-flop with two data inputs, J and K, and a clock input such that its state transitions satisfy the excitation requirement defined in Table 3.1(c).
4. Trigger flip-flop, TFF, is a flip-flop with one data input, T, and a clock input such that its state transition satisfy the excitation requirement defined in Table 3.1(d).

From the definition, only the SRFF has the possibility to get into ambiguous state when SR=11 due to certain fault. It is, therefore, impossible to detect this kind of fault. Fortunately, the operation of SRFF is actually a special case of the operation of JKFF. We can always use JKFF to substitute SRFF to alliviate this difficulty. Therefore only three types of flip-flops will be considered in the subsequent discussion.

Often a master-slave type flip-flop has one additional input, namely, direct reset Rd. It is an asynchronous input. It can override the control of clock pulse and data input(s). Namely, when this input is present, the normal operation of the device is inhibited and the output goes to 0 state. The symbols of these three types of master-slave flip-flops are shown in Fig. 3.3. They will be used throughout the thesis.



$$\delta_{ij}[S_j(t),X_i(t)]=S_k(t+1) \quad \forall \quad 0\leq i\leq 2^n-1$$
$$\lambda_{ij}[S_j(t),X_i(t)]=Z_l(t) \qquad 0\leq j,k\leq 2^q-1, \quad 0\leq l\leq 2^m-1$$

(a) (b)

Figure 3.2
A General Representation of State Table and State Diagram

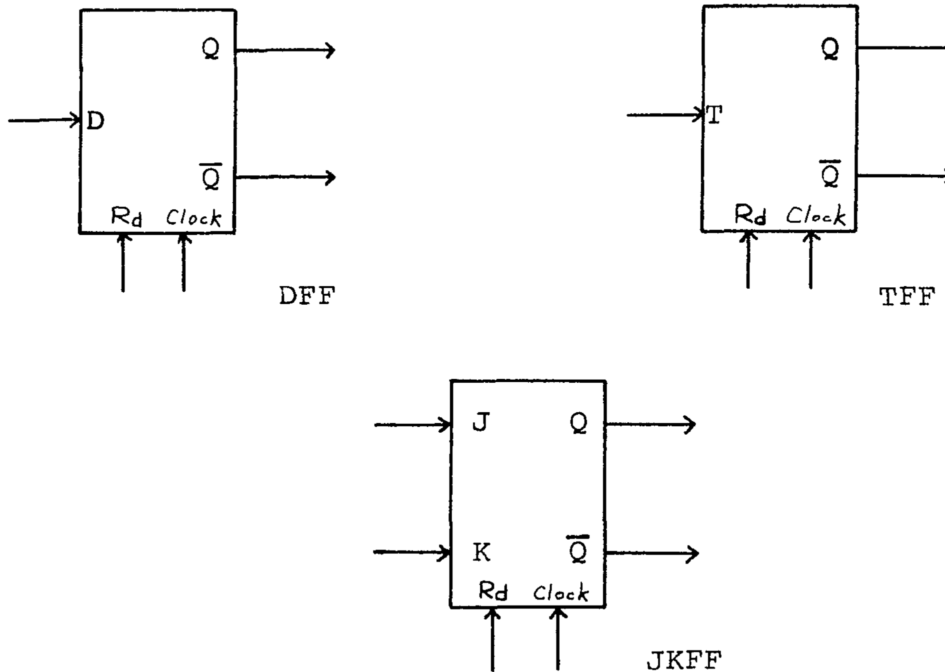


Figure 3.3
Logic Symbols of the Four Basic Master-Slave Flip-Flops

3.3 Fault Dectection Analysis and Fault Model

In a synchronous sequential network, the derivation of fault detection tests is extremely complex since the signal lines for the state variables are usually neither observable nor directly controllable. The fault present at time t may not affect the output at time t, hence we cannot in general, detect a permament fault by a single test pattern. It requires a sequence of test patterns to detect the fault in a sequential network, and the length of the test sequence is proportional to the number of memory elements in the network.

Methods for deriving test sequences for detecting stuck-at-fault in sequential networks, have been extensively investigated and developed in the past.

Galey [16] proposed to break all the feedback paths in order to make the network purely combinational so that the fault detection procedures for the combinational network can be applied. But it is impossible to break the feedback paths which are connected inside a flip-flop, which makes this approach inapplicable in most networks.

Williams and Angell [49] extended the idea of breaking the feedback paths, and, in addition, reconstructing the flip-flops to form a shift register so that the state can be easily set or checked by the use of a test point. The cost analysis [49] indicates that there may be some application of this approach in the future, when the yields

obtainable for LSI chips have improved.

Another method is to bring all the feedback path out of the terminals so that the state of the network can be monitored at any time, however, it will cause a severe reduction in circuit-to-pin ratio and require excess test points.

Hennie [21] proposed an automated method which has the advantage in that the method does not require the faulty state table generated by a given fault. This method was later used by Kime [24]. In this approach, each transition of the state table is attempted, and its correctness is checked by a distinguishing sequence (also called diagnosing sequence). A distinguishing sequence is an input sequence which, when applied to a network, results in a different output sequence for each distinct initial state of the network. The determination of such a distinguishing sequence is a well-known process. However, for networks having many states, this determination process will require both a complex program and large amounts of computer time and memory. Furthermore, Hennie states that in general, a test sequence (checking experiment) having $mn \times (2L + 2n - 1) + n(n-1) + (n+1)L$ tests will be required for a network in which m is the number of inputs, n, the number of states, and L, the length of the distinguishing sequence. However, the test sequence will be reduced to $mn \times (2L+n) + (n+1)L$ under the optimum condition.

Poage and McCluskey [32] proposed a procedure for deriving minimal-length test sequence which requires all the faulty state tables. A combined state table consisting of the state tables of all the faulty networks with that of fault-free network need be generated and this combined state table will identify the states for which a given fault will produce different output patterns. This procedure also requires a large amount of computer time and memory to generate and compare these state tables.

Seshu [43], [44] developed another automated procedure which generates a testing sequence by simulation and local optimization. The good network and all the faulty networks are simulated. At each step of the testing sequence, all possible changes of inputs are tried and the one which detects the most faults is selected as the next test combination. The process is then repeated to find the next combination. However, this method may not generate tests for some detectable faults in the network.

An application of this procedure has been described by Manning [28]. Manning's procedure does not require the knowledge of the network's state table. The procedure initially finds, a large number of faults per test. However, after a majority of the faults have been detected, the procedure loses the direction.

Hsieh [22] introduced the input/output sequence by using different literals for each transition output to

derive a shorter checking experiment.

Kohavi [25], [26] modified Henni's procedure by using different distinguishing sequence for different states (called variable length distinguishing sequence) to reduce the length of checking experiment.

The techniques described above, in general, require a complex algorithm to generate the nearly complete test set for detecting the faults in a sequential network, and require a hard core computer to perform the test sequence.

The technique developed in the preceding chapter can be extended to design an easily detectable synchronous sequential network with the use of specialized hardware and topological constraints of the network. The method of generating the fault detection test sequence applicable in the real time enviroment will be developed based on the fault model defined in the subsequent discussions.

Each flip-flop has its own distinct excitation requirements as described earlier. To determine the accuracy of a flip-flop operation, it is not necessary to identify or locate what is wrong inside the flip-flop. The important aspect is to know whether or not the flip-flop does function properly according to its desired excitation requirement shown in Table 3.1. If the flip-flop can not function correctly for a certain excitation, then we say that the flip-flop has malfunction. The malfunction of the flip-flop can be classified, in general, into one of the six categories :

1. Type A Malfunction is defined to be the flip-flop fails to remain at its previous 0 state.
2. Type B Malfunction is defined to be the flip-flop fails to remain at its previous 1 state.
3. Type C Malfunction is defined to be the flip-flop fails to change to 0 state.
4. Type D Malfunction is defined to be the flip-flop fails to change to 1 state.
5. Type E Malfunction is defined to be the flip-flop fails to copy its 0 input.
6. Type F Malfunction is defined to be the flip-flop fails to copy its 1 input.

Based on these definitions, JKFF,TFF will have four types of malfunctions, (i.e., type A,B,C and D) and DFF will have only two types of malfunctions (i.e., type E and F). If the flip-flop does not have the malfunctions described above, then it is ensured to function properly under all the possible inputs unless there exists an intermittent (transient) fault inside the flip-flop.

In the following discussion, let FFi/A(B,C,D,E,or F) denote the flip-flop i has type A(B,C,D,E or F) malfunction respectively, and the fault in a flip-flop is restricted to the single permanent fault. Fault f1 and f2 in the flip-flop are said to be equivalent if their existence will cause the same type of malfunction.

Theorem 3.1 The function of a Delay flip-flop, DFF, can be detected by two tests (D=0 and D=1) if the flip-flop has a direct reset input and state output q is observable.

Proof: Consider the excitation table shown in Table 3.1(b). The state of the DFF can be direct reset to the 0 state (q=0) by applying an Rd=1 pulse. Then the test D=1 should force the state change to the 1 state (q=1), if there is no DFF/F. Otherwise the state output is still 0 (q=0) and the DFF/F is detected. If the DFF passes the first test, then the second test D=0 should force the state change back to the 0 state if there is no DFF/E. Otherwise, the state output is still 1 (q=1) and the DFF/E is detected. If the DFF passes the second test, then it is guaranteed to function properly since the excitation table is validated by the two tests.

Notice that the fault Rd/1 will force the state q=0 regardless of the logic value of input D. However, test D=1 will detect it since the state output must be 1 (q=1) for normal function. So the fault Rd/1 is actually equivalent to DFF/F. On the other hand, the fault Rd/0 will not reset the DFF to 0 state, but it has no actual effect on the DFF under single fault assumption.

Theorem 3.2 The function of a JK flip-flop, JKFF, can be checked by the four tests (JK=0-, 1-, -0, -1), if the JKFF has a direct reset input Rd and the state output q is observable.

Proof: The state of a JKFF can be directly reset to the 0 state (q=0) by applying an Rd=1 pulse. Then from the excitation table shown in Table 3.1 (c), the first test JK=0- should force the FF to remain at q=0 state, if there is no JKFF/A. Otherwise, the state output q=1 will indicate the existence of JKFF/A. If the FF passes the first test, then the second test JK=1- should force FF to change to q=1 state, if there is no JKFF/D. Otherewise, the state output q=0 will indicate the existence of JKFF/D. If the FF passes the second test, then the third test JK=-0 should force FF to remain at q=1 state, if there is no JKFF/B. Otherwise, the state output q=0 will indicate the existence of JKFF/B. Finally, if the FF passes the third test, then the fourth test JK=-1 should force the FF to change to q=0 state, if there is no JKFF/C. Otherwise, the state output q=1 will indicate the existence of JKFF/C. Once the JKFF passes the proposed four tests, then the proper function of JKFF is guaranteed since the excitation table is now validated.

Notice that the fault Rd/1 will force the FF to q=0 state regardless of the input pattern JK. However, it will be detected by the test JK=10. So that the fault Rd/1 is equivalent to JKFF/D. The fault Rd/0 has no effect on the proposed test sequence under the single fault assumption.

Similar analysis can be applied for the TFF. The following corollaries are concluded :

Corollary 3.2.1 The proper function of a Trigger flip-flop, TFF, can be checked by the four tests (T=0, 1, 0, 1), if the TFF has a direct reset input Rd and the state output is observable.

As a result, we have the following test sequences for detecting the malfunction of the three types of flip-flops when an Rd=1 pulse is applied to reset each flip-flop to the 0 state :

1. Test sequence for D flip-flop is : D=1, 0 (or 0, 1).
2. Test sequence for JK flip-flop is : JK=01, 10, 10, 01.
3. Test sequence for T flip-flop is : T=0, 1, 0, 1.

If we can provide the corresponding test sequence to each flip-flop of the network simultaneously, then the malfunction of the network will be detected by at most four tests as concluded later. Therefore, it can be considered real time fault detection. But the requirement of these test sequences inject certain topological constraints on the network which will be discussed in the next section.

3.4 Network Topology and Controllable Memory Modules

In this section a synchronous sequential network is analyzed structurally. Some definitions are introduced in order to facilitate the discussion.

1. Each gate in the combinational block defines a _signal node_ and the set of all signal nodes is denoted as _SN_.
2. Each input variable defines a _primary input node_ and the set of all primary input nodes is denoted as _PIN_.
3. Each observable output terminal defines a _primary output node_ and the set of all primary output nodes is denoted as _PON_.
4. Each memory element (flip-flop) defines a _memory node_ and the set of all memory nodes is denoted as _MN_.
5. The signal line between two signal nodes is simply called the _signal line_ and the set of all signal lines is denoted as _SL_.
6. The signal line from a primary input node to a signal node or a memory node is called the _primary input line_ and the set of all primary input lines is denoted as _PIL._
7. The signal line from a signal node or a memory node

to a primary output node is called the primary output line and the set of all primary output lines is denoted as POL.

8. The signal line from a memory node to a signal node is called the memory state line and the set of all primary state lines is denoted as MSL.
9. The signal line from a memory node to a memory node is called the memory memory line and the set of all memory lines denoted as MML.
10. The signal line from a signal node to a memory node is called the memory input line and the set of all memory input lines is denoted as MIL.

Based on the definitions, any synchronous sequential network can be represented by a directed graph which consists of a set of nodes N and a set of directed edges E, such that N= PIN U PON U SN U MN and E= SL U PIL U POL U MSL U MIL U MML.

For the synchronous sequential network shown in Fig. 3.4 its corresponding directed graph is shown in Fig. 3.5, in which,

$$PIN = [x_1, x_2, x_3]$$

$$PON = [q_1, q_2]$$

$$SN = [g_1, g_2, \ldots, g_8, g_9]$$

$$MN = [DFF_1, DFF_2]$$

$$SL = [4, 6, 10, 11, 12, 13, 14, 15]$$

$$PIL = [1, 2, 3, 5, 9]$$

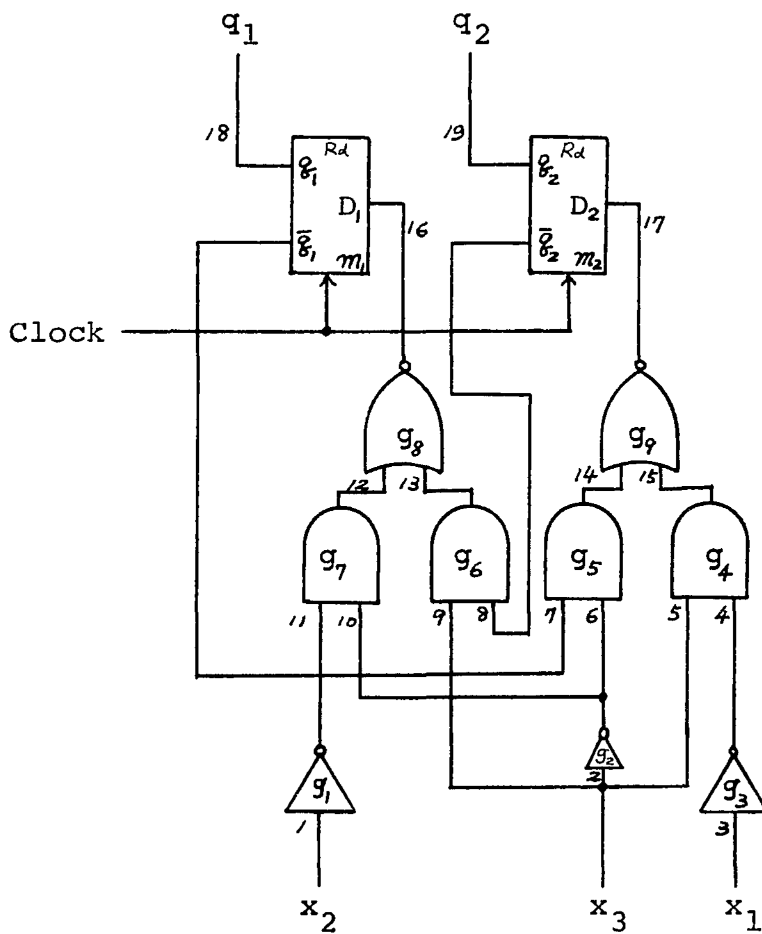


Figure 3.4
A Two Bit Shift Right/Shift Left Register

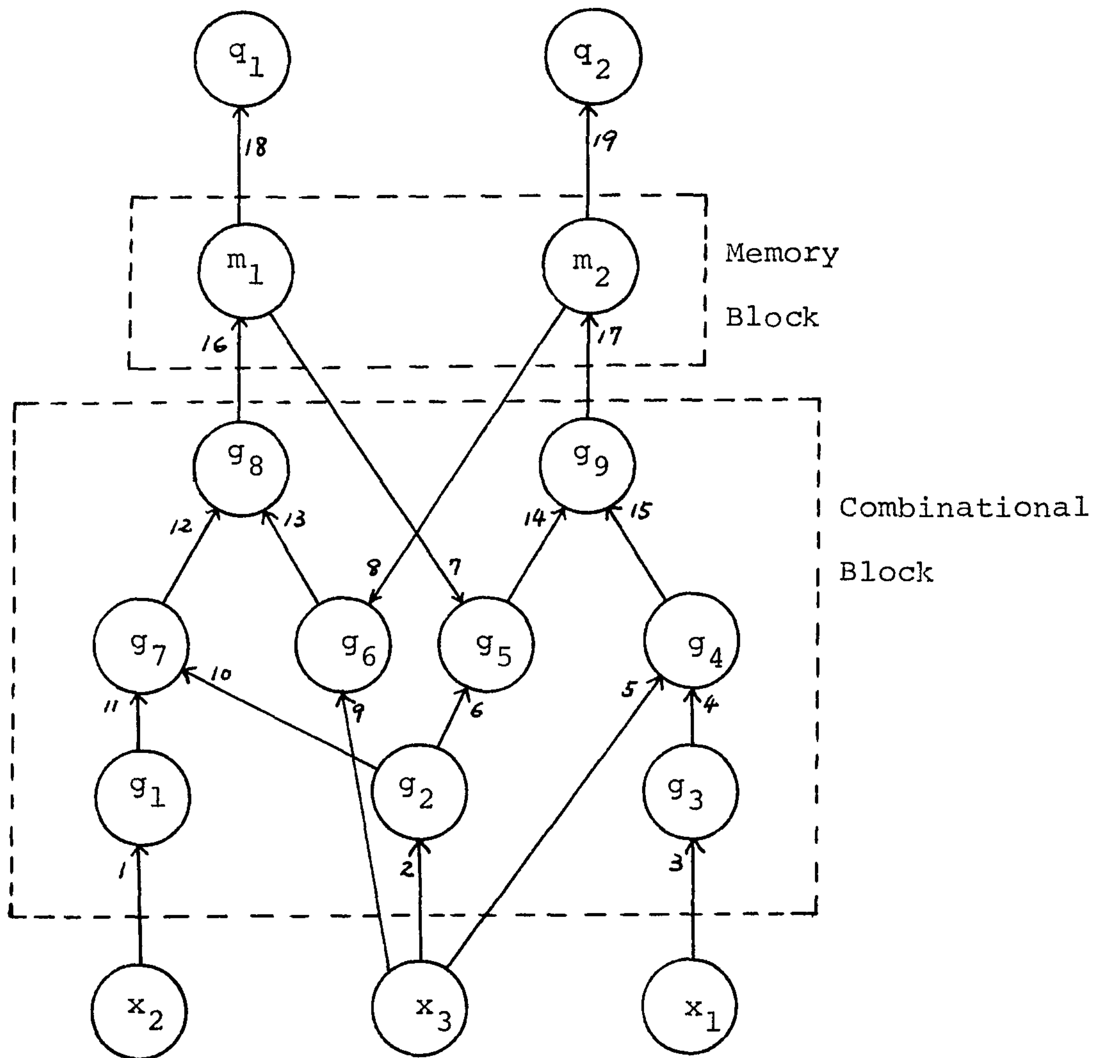


Figure 3.5
The Directed Graph of the Network of Figure 3.4

POL = [18,19]

MSL = [7,8]

MIL = [16,17]

MML = $\emptyset$

The characteristic of certain topological structure of a synchronous sequential network is analyzed in the following discussions.

Consider the specific topological structure of a synchronous network shown in Fig. 3.6. Its characteristic can be described by the following theorem.

Theorem 3.3 The stuck-at-fault of the combinational block and the malfunctions of the memory elements of a synchronous sequential network without memory state line (MSL $=\emptyset$) and memory memory line (MML $=\emptyset$), and with completely observable state outputs can be detected by the test sequence developed in section 3.3, if its combinational block is implemented with the programmable logic module (PLM's) and the necessary controlled signal invertors (CSI's).

Proof: If MSL $=\emptyset$, then the accessible primary inputs will be the only inputs to the combinational block. Therefore, from the theorem 2.7, this combinational block can be constructed by the proposed PLM's and CSI's such that the output of the combinational block always carry the error information during the two test modes by two test input patterns. In addition, the outputs of the combinational block can be controlled to have any desired logic value by the systematic

design algorithm. Consequently, it is always possible to supply the desired test sequence for each flip-flop since MML $=\emptyset$. The error information on a signal path which leads to the flip-flop's input and the function of the corresponding flip-flop can be detected by the test sequence and the error information can be observed at the memory output lines.

This property is illustrated by the following example.

Example 1

Consider the sequential network shown in Fig. 3.7(a) where

PIN=[A,B,C,D]

PON=[Z]

SN=[g_1,g_2,g_3]

MN=[DFF]

SL=[e,f]

PIL=[a,b,c,d]

POL=[h]

MSL=$\emptyset$

MML=$\emptyset$

MIL=[g]

The combinational block indicated by the box in Fig. 3.7(a) can be converted to the modified combinational block as shown in Fig. 3.7(b) according to the systematic design algorithm developed in Chapter 2. If we apply the Rd=1 pulse to rest the DFF to 0 state (Q=0), then the stuck-at-faults in the combinational block and the malfunctions of

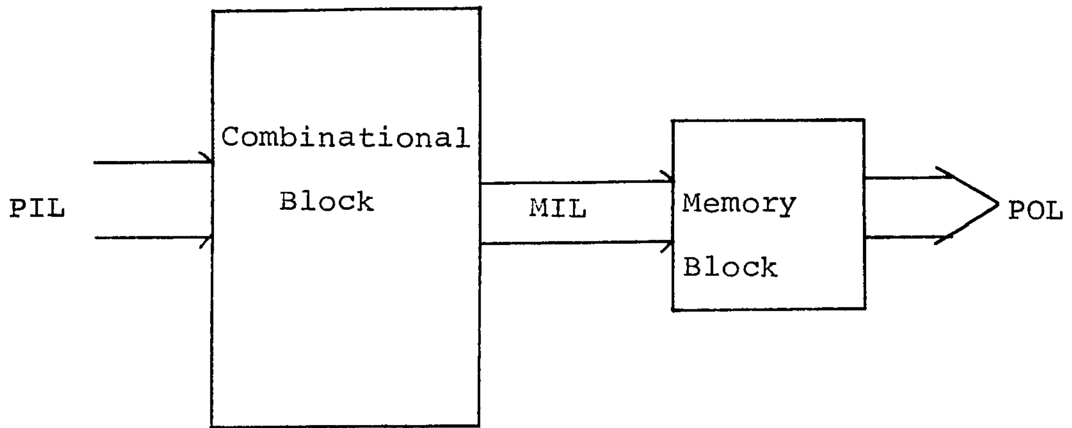


Figure 3.6
A Special Structure of Synchronous Sequential Network for Theorem 3.3

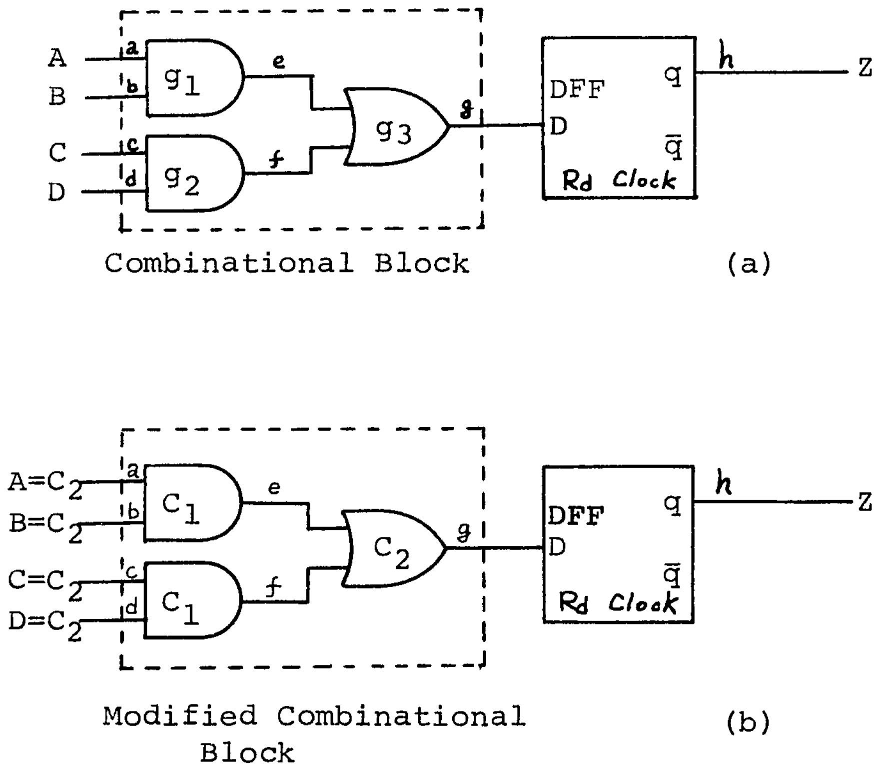


Figure 3.7
A Synchronous Sequential Network Illustrates Theorem 3.3

DFF will be detected by the test sequence abcd=1111,0000 as follows:

<u>Test 1</u> Set C_1 C_2=01 and apply test abcd=1111. Then the error information in Ha **where** Ha=[a/o,b/o,c/o,d/o,e/o, f/o,g/o] will propagate to memory input line g. If there exists no fault in Ha then g will provide the test D=1 to detect the type F malfunction in the DFF(DFF/F) by observing the memory output line h, (h=o indicates an error). If there exists faults in Ha, then g=o. This will provide the test D=o. In this case the DFF will remain at o state, so the memory output line h=o. Therefore, an error is indicated. Ha and DFF/F gives the same indication on the observable output h. They are actually equivalent.

If there exists no stuck-at-faults in the Ha and no DFF/F, then the DFF will be in the 1 state (i.e., h=1). Then go to the next test.

<u>Test 2</u> Set C_1C_2=10 and apply test abcd=0000. Then the error information caused by the stuck-at-fault in Hb where Hb=[a/1,b/1,c/1,d/1,e/1,f/1,g/1] will propagate to the memory input line g. If there exists no fault in Hb, then g will provide the test D=0 to detect the type E malfunction in the DFF (DFF/E) by observing the memory output line h, h=1 indicates the existence of DFF/E. If there exists faults in Hb, then g=1. In this case, the DFF will still stay at 1 state, i.e. h=1. So the memory output line h still indicates the error information. Therefore, Hb and DFF/E

can be detected by this test mode.

Once the network passes these two tests, it is guaranteed to be free of stuck-at-fault and malfunctions, and the network is supposed to function correctly unless there exists certain intermittent faults.

Theorem 3.4 Given a synchronous sequential network whose structure (as shown in Figure 3.8) satisfies the following conditions:

Its combinational block can be partitioned into two distinct blocks -- block A and block B such that:

1. All the inputs to block A are primary input line (PILa),
2. All the outputs of block A are memory input lines (MIL),
3. All the inputs to block B are primary input lines (PILb) and memory state lines (MSL), and
4. All the outputs of block B are primary output lines (POL) where PILa $\cap$ PILb=$\emptyset$ and PILa U PILb=PIL.

Then the stuck-at-faults of the combinational block and malfunctions of the memory elements of the network can be detected by the test sequence developed in section 3.3, provided that the combinational block are converted to their corresponding modified combinational block with the proposed programmable logic module (PLM's) and Controlled Signal Inverter (CSI's)

Proof: From theorem 3.3, it is verified that the error information of the stuck-at-faults and malfunction of the

sub-network indicated by the dotted box in Fig. 3.8 can always propagate to the MSL by the proposed test sequence. In addition, the logic value of each line in MSL can be controlled by the logic value specified by the test input pattern for each line in PILa. Therefore, we can specify the proper logic value of each line in PILb so as to make the input pattern formed by the logic value of each line in PILb and MSL the most sensitive input pattern to sense the error information on the MSL and PILb, and also detect the stuck-at-faults of block B by observing the logic value at each line in POL. The proof is then completed.

Example 2

Consider the sequential network shown in Figure 3.9(a) where

PIN=[A,B,C,D,E]

PON=[Z]

SN=[g_1,g_2,g_3,g_4] where SNa=[g_1,g_2,g_3], SNb=[g_4]

MN=[DFF]

SL=[e,f]

PIL=[a,b,c,d,h] where PILa=[a,b,c,d], PILb=[h]

POL= [j]

MSL=[i]

MIL=[g]

MML= $\emptyset$

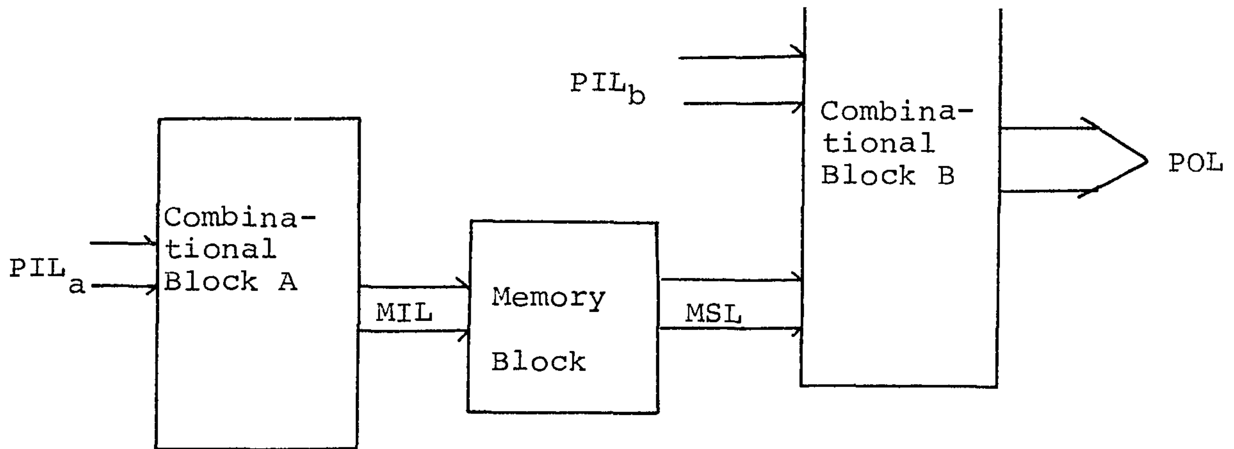


Figure 3.8

A Specific Configuration of Synchronous Sequential Network for Theorem 3.4

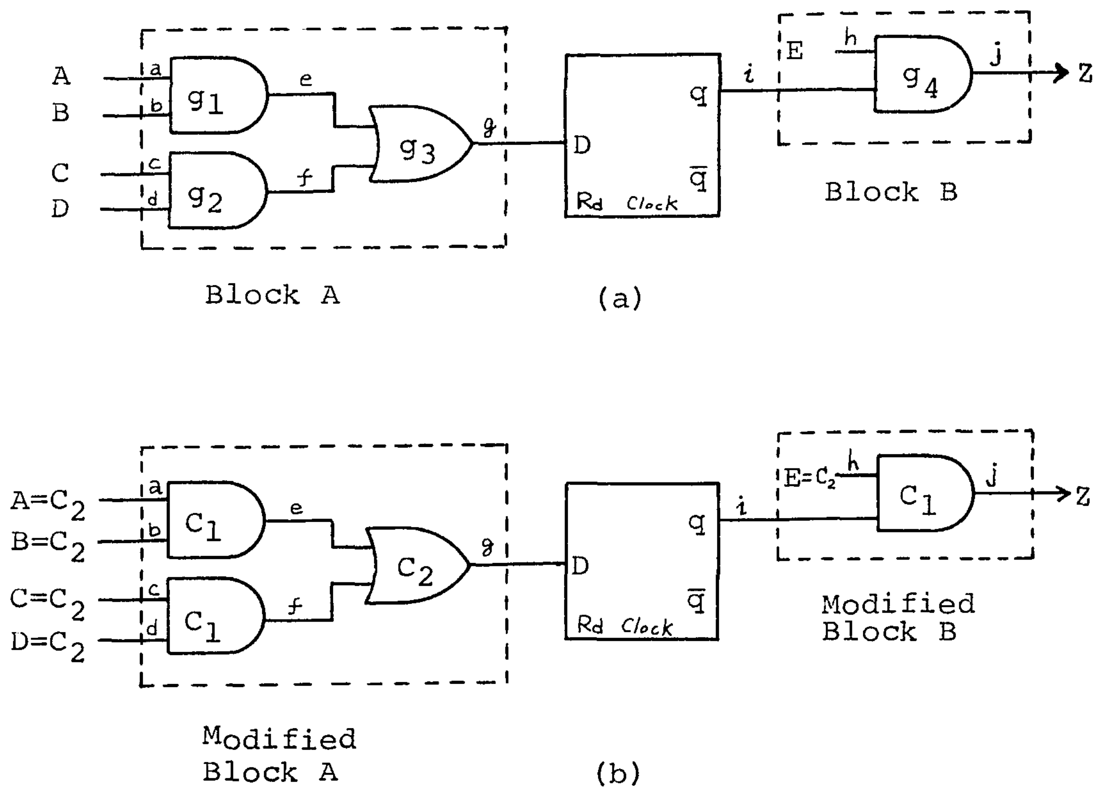


Figure 3.9

A Synchronous Sequential Network Illustrates Theorem 3.4

The combinational block A and B can be converted to the modified combinational block A and B as shown in the figure 3.9(b) by the systematic design alogrithm. If we apply the Rd=1 pulse to reset the DFF to 0 state (Q=0), then the stuck-at-faults in the combinational block A and B and the malfunction of the DFF will be detected by the test sequence abcdh=11111,00000 described as follows:

Test 1 Set C_1C_2=01 and apply test abcdh=11111, then the fault information produced by the stuck-at-fault in Ha where Ha=[a/o,b/o, c/o,d/o,e/o,f/o,g/o,h/o,i/o,j/o] and the type F malfunction in DFF can be detected by observing the output Z. If Z=o, then the existence of stuck-at-faults in Ha and/or DFF/F is indicated. Otherwise, the network passes this test and go to the next test.

Test 2 Set C_1C_2=10 and apply test abcdh=00000. Similarly, the fault information produced by the stuck-at-faults in Hb where Hb=[a/1,b/1,c/1,d/1,e/1,f/1,g/1,h/1,i/1,j/1] and the type E malfunction in DFF can be detected by observing the output Z. If Z=1, then it indicates the existence of some stuck-at-faults and/or DFF/E malfunction. Otherwise, the network passes this test and is ensured to function properly.

Unfortunately, not every synchronous sequential network whose structure will satisfy the conditions posed by the Theorems 3.3 or 3.4.

To alleviate this difficulty, a new type of memory element is proposed and defined as follows:

A <u>controllable memory module</u> CMM is defined as a memory element with direct reset capability whose state outputs are convertible by two external control signals.

A general model of CMM is shown in Figure 3.10. If the flip-flop used in the module is DFF (or TFF, JKFF), then the input to the module is D (or T, JK), and the module is denoted as DCMM (or TCMM, JKCMM) respectively. The logic symbols of the three basic CMMs are shown in Figure 3.11. The outputs of CMM are y_i and $\bar{y}_i$ where $y_i = q_i \oplus Cq_i$ and $\bar{y}_i = \bar{q}_i \oplus C\bar{q}_i$. Cq_i and $C\bar{q}_i$ are the two external control signals. The CMM will operate like the pure flip-flop unit in the module when the two control signals are set to 0.

In the following discussion, we treat the faults in CMM based on its functional point of view.

Consider the DCMM as shown in Figure 3.11(a). The type F malfunction (DCMM/F) can be caused by D/0, q/0, Cq/1 (Cq/0) and y/0 (y/1) and the internal equivalent faults. DCMM/F can be detected by first applying the direct reset to the module and then applying the test D=1 and Cq=0 (Cq=1). The type E malfunction (DCMM/E) can be caused by D/1, q/1, Cq/1 (Cq/0) and y/1 (y/0) and the internal equivalent faults. DCMM/E can be detected by first applying the direct reset to the module and then applying the test D=0 and Cq=0 (Cq=1). So the malfunction of DCMM can be detected by the test sequence DCa= 10 (or 11) and 00 (or 01).

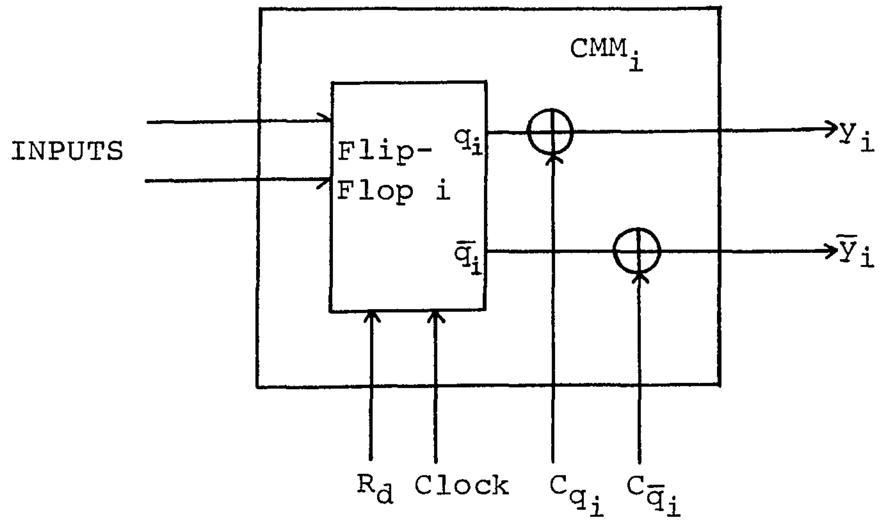


Figure 3.10
A General Model of Controllable Memory Module (CMM)

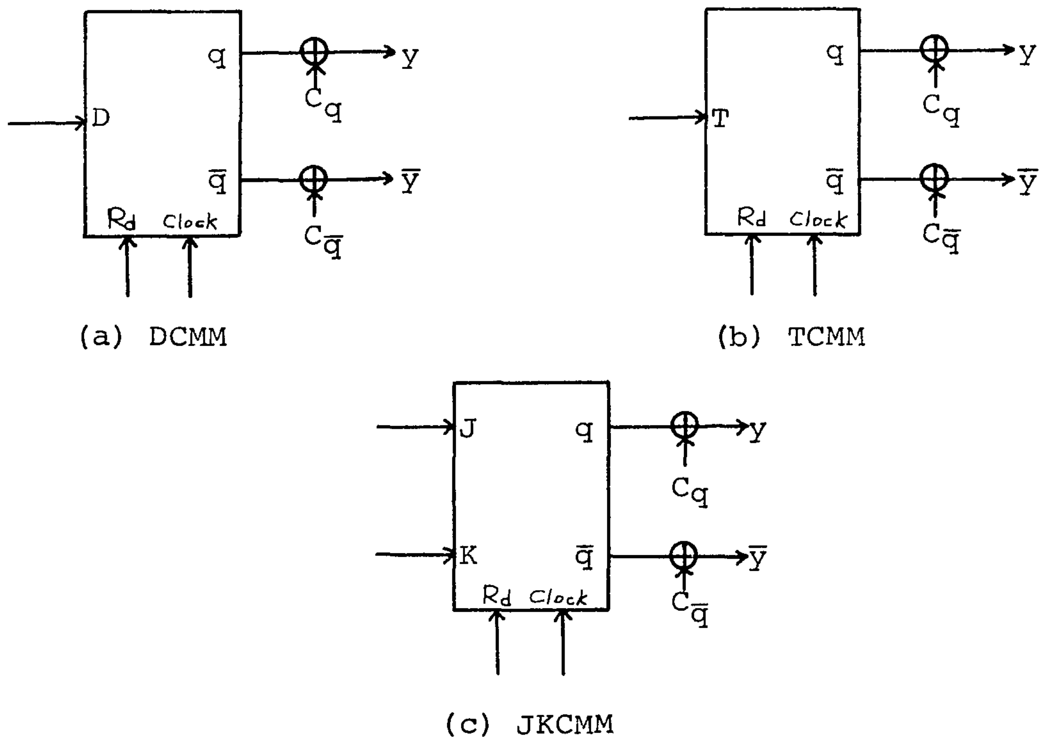


Figure 3.11
Logic Symbols of the Four Basic CMM's

Consider the JKCMM as shown in Figure 3.11(c). The type A malfunction (JKCMM/A) can be caused by J/1, q/1,Cq/1 (Cq/0) and y/1 (y/0) and the equivalent internal faults. The type B malfunction can be caused by K/1, q/0, Cq/1 (Cq/0) and y/0 (y/1) and the equivalent internal faults. The type C malfunction can be caused by K/0, q/1, Cq/1 (Cq/0) and y/1 (y/0) and the equivalent internal fualts. The type D malfunction can be caused by J/0, q/0, Cq/1 (Cq/0) and y/0 (y/1) and the equivalent internal faults. It is found that the type A, D, B, C malfunctions of JKCMM can be detected sequentially by first directly resetting the module and then applying the test sequence JKCq= 0-0(0-1), 1-0(1-1), -00(-01) and -10(-11) in which - means don't care (i.e., either 0 or 1).

By the similar analysis, it is concluded that the type A, D, B and C malfuntions of TCMM can be detected sequentially by first directly resetting the module and applying the test sequence TCq= 00(01), 10(11), 00(01) and 10(11).

Notice that the control signal Cq can be selected independently. However, the desired observable output states depend on the selected logic value of Cq.

The utilization of proposed CMM's to enhance the testability of a synchronous sequential network will be discussed in the following section.

3.5 Design with Testability Enhancement

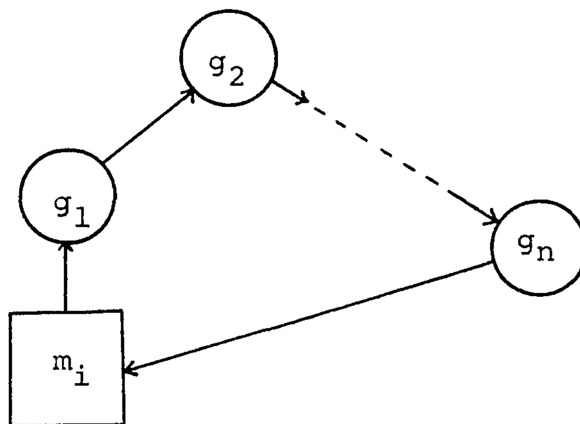
Real time fault detection of a synchronous sequential network can be achieved if it is possible to provide the proposed test sequence to each flip-flop of the network simultaneously.

A signal path is said to be a state monitor signal path (smsp). If the path starts from a memory node, then passes through a set of signal nodes and ends with a primary output node. The set of all possible smsp of a given network is denoted as SMSP.

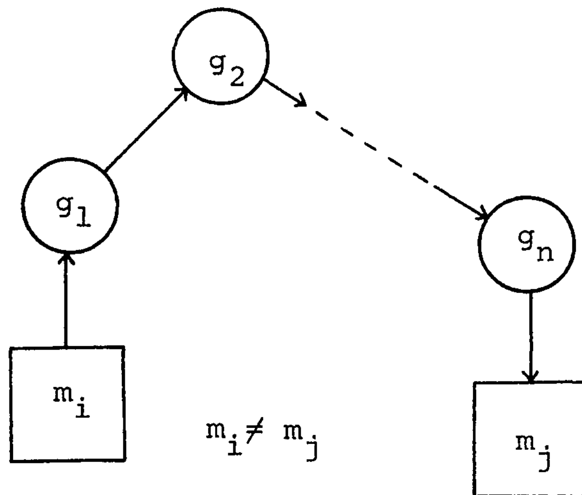
A signal path is said to be a memory to memory signal path (mmsp), if the path starts from a memory node m_i, then passes through a set of none fanout nodes and ends with a memory node m_j for some $m_i, m_j \in M \cdot N$ (the set of all memory node) in which i may or may not equal to j. The set of all the possible mmsp of a given sequential network is denoted as MMSP. Two basic structures of mmsp are shown in Figure 3.12(a) and (b), where $0 \leq n \leq N$ and N is any positive integer.

If a synchronous sequential network contains a mmsp, then the network will not satisfy the conditions of theorem 3.3 or 3.4. Therefore, it will be impossible to supply the proposed test sequence to each flip-flop simultaneously.

The proposed CMM's provide a method of supplying the proper input pattern to the cominational network so as to produce the desired test sequence to each flip-flop simultaenously for a given synchronous sequential network.



(a)



(b)

Figure 3.12
Two Basic Structures of the Memory to Memory Signal Path (mmsp)

A synchronous sequential network is called the modified synchronous sequential network (MSSN) if a proper CMM is used to substitute for each flip-flop in the memory block and the PLM's and CSI's are utilized to implement the combinational block.

Theorem 3.5 Any synchronous sequential network with no memory to memory signal path (mmsp) and contains a state monitor signal path (smsp) for each memory state line of MSL. Then the network can be converted to MSSN so that the stuck-at-faults and malfunctions can be detected by the test sequences for the three types of flip-flops developed in section 3.3.

Proof: In MSSN, two control signals of CMM provide the facility to supply portion of the MSIP for the combinational block in conjunction with the proper primary input pattern. Therefore, the logic value of each memory input line of MIL can be controlled so as to provide the proposed test sequence for each CMM simultaneously. In addition, the fault information on each memory state line of MSL will not be delayed by the clock, since MMSP= $\emptyset$. Hence the fault information will be sensed by the combinational block immediately. Because there exists a smsp for each memory state line of MSL, the fault information of all the memory state line will propagate to the primary observable outputs. Therefore, any stuck-at-fault and malfunction will be indicated when test sequence is applied.

The important property of the above theorem will be demonstrated by the following example.

Example 3

Consider a synchronous sequential network shown in Figure 3.13(a) where

PIN=[A,B]

PON=[Z]

SN=[g_1,g_2]

MN=[JKFF]

SL=[d]

POL=[g]

PIL=[a,f]

MSL=[b,c]

MIL=[e]

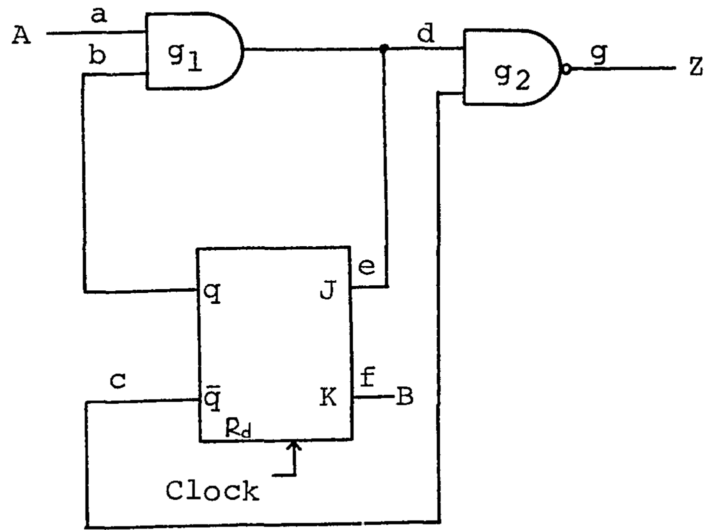
MML=∅

MMSP=∅

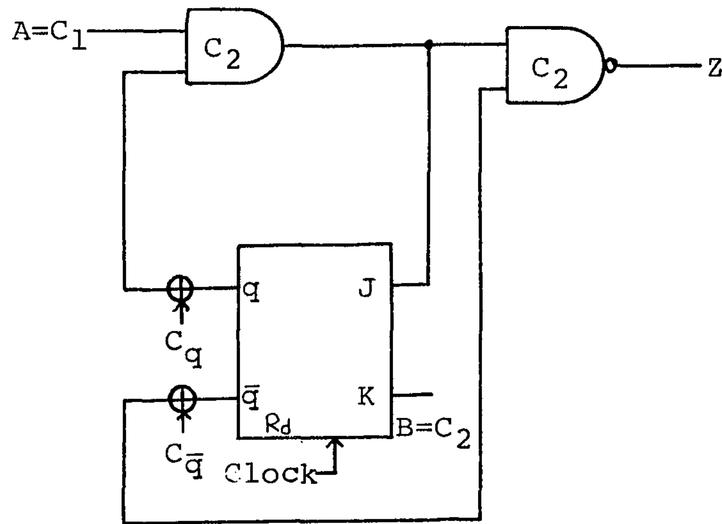
SMSP=[bdg,cg]

It is converted to the MSSN as shown in Figure 3.13(b). The control signals for PLM's and JKCMM are also indicated in the figure. If we apply the Rd=1 pulse to reset the JKCMM,(q=0) then the stuck-at-faults in the combinational block and malfunction of the JKCMM will be detected by the test sequence ab=01,10,10 and 01 described as follows:

Test 1 Set $C_1C_2CqC\bar{q}$=0101 and apply test ab=01, then the stuck-at-faults in Ha where Ha=[a/1,b/1,c/1,d/1,e/1,g/0] can be detected by observing the primary output Z (Z=0 indicates



(a)



(b)

Figure 3.13
A Synchronous Sequential Network with No mmsp

an error). If Z=1, then apply the clock pulse to detect the type A malfunction of JKCMM (JKCMM/A). If there exists JKCMM/A, then q =1. This fault information will propagate to Z and force Z=0 to indicate an error. Otherwise, q=0, proceed to next test.

Test 2 Set $C_1C_2CqC\bar{q}$=1010 and apply test ab=10, then the stuck-at-faults in Hb where Hb=[a/o,b/o,c/o,d/o,e/o,g/1] can be detected by observing the output Z (Z=1 indicates an error). If Z=0, then apply the clock pulse to detect the JKCMM/D malfunction. If there exists JKCMM/D, then q=0. This fault information will propogate to Z and keep Z=0 to indicate an error. Otherwise, q=1 (notice that this will force Z=1), proceed to next test.

Test 3 Set $C_1C_2CqC\bar{q}$=1001 and apply test ab=10, then the stuck-at-faults in Hb will be indicated by Z=1. If Z=0, then apply the clock pulse to detect the JKCMM/B malfunction. If there exists JKCMM/B, then q=0. This fault information will propogate to Z and force Z=1 to indicate an error. Otherwise, q=1, proceed to next test.

Test 4 Set $C_1C_2CqC\bar{q}$=0110 and apply test ab=01, then the stuck-at-faults in Ha will be indicated by Z=0. If Z=1, then apply the clock pulse to detect the JKCMM/C malfunction. If there exists JKCMM/C, then q=1. This fault information will propagate to Z and keep Z=1 to indicate an error. Otherwise, q=0, and the test sequence is completed.

Once the MSSN passes the above four tests, it is ensured to be free of stuch-at-faults and malfunctions. The MSSN will perform the normal (original) function by setting control signals $C_1C_2C_qC_{\bar{q}}$=0000.

In general, most of the synchronous sequential networks will satisfy the topological requirements of theorem 3.5. Consequently, a wide class of synchronous sequential network can be designed by the proposed technique to enhance the testability of the network in real time applications. Another important aspect of this theorem is that the test sequence is independent of the number of flip-flop used and the type of flip-flop used in the network.

A restricted class of synchronous sequential network which contains some memory to memory signal path (mmsp) will not satisfy the topology of theorem 3.5. The existence of an mmsp will cause the fault information of its memory state line to be delayed. It will provide the incorrect test sequence for the next test. Therefore, the fault indication in the next test will lead to a wrong conclusion. This effect will be demonstrated by the following example:

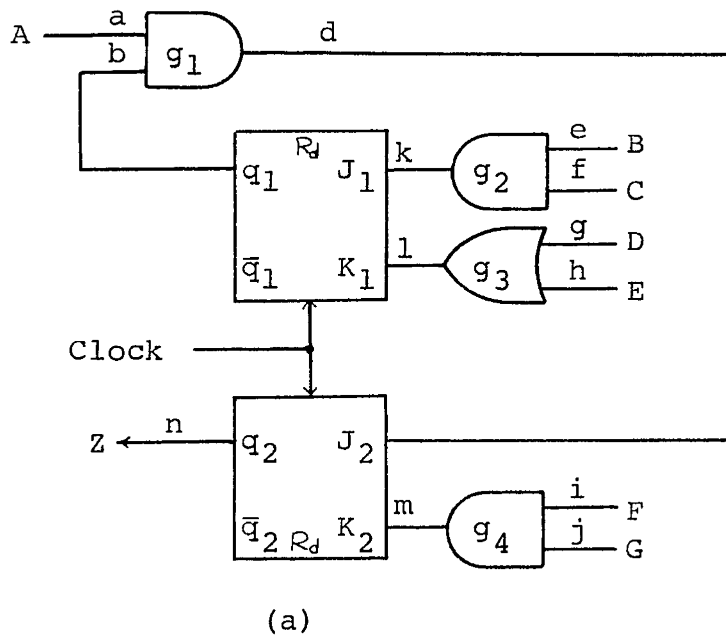
Example 4

Consider the modified synchronous sequential network shown in Figure 3.14(b). Where

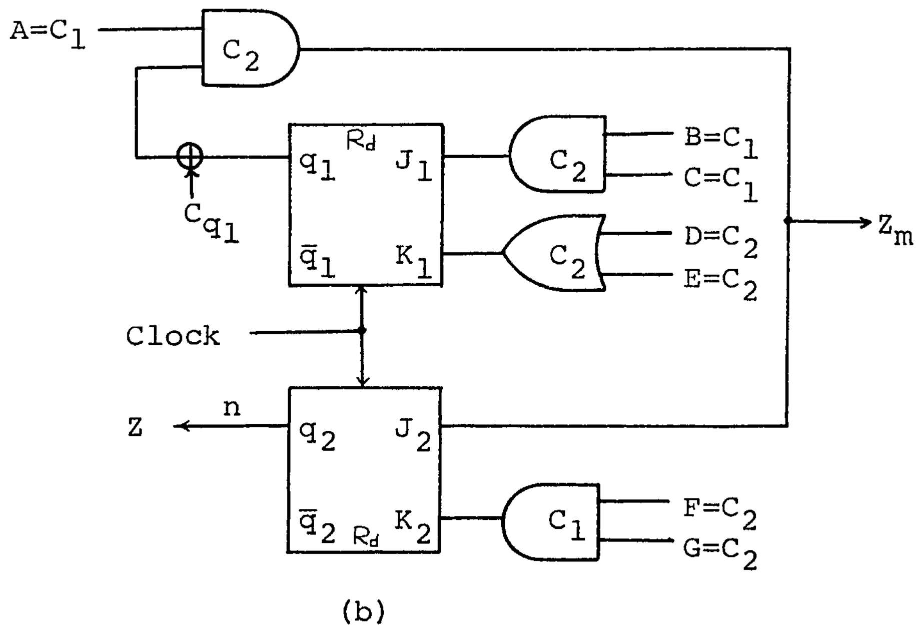
MSL=[b]

MMSP=[bd]

SMSP=[n]



(a)



(b)

Figure 3.14
A Synchronous Sequential Network with mmsp

Assume the Rd=1 pulse is applied to reset the JKCMM and JKFF. Then set $C_1C_2Cq_1$=010 and apply the test ABCDEFG=0001111. If there exists JKCMM/A, then q_1=1. This fault information will not be detected. For the next test, set $C_1C_2Cq_1$=101 and apply ABCDEFG=1110000. If there exists no JKFF/D, but the fault JKCMM/A will make J_2=0 (J_2=1 is the desired test now). As a result, it causes the output Z to indicate an error for JKFF/D malfunction which is erroneous.

This difficulty can be avoided by adding an additional monitor output Zm as shown in Fig. 3.14(b). Then the JKCMM/A will be indicated by Zm=1 right after the clock pulse is applied.

The above discussion demonstrates that the problem posed by the existence of mmsp can be solved by inserting the monitor output Zm at the troublesome memory state lines. Hence, the following theorem can be concluded.

Theorem 3.6 Any synchronous sequential network can be converted to the modified synchronous sequential network MSSN so that the stuck-at-faults and malfunctions of the network can be detected by the test sequences of the four flip-flops in section 3.3 if a monitor output is inserted at each memory state line (msl) which has memory to memory signal path (mmsp).

Proof: If a monitor output is added at each msl which has mmsp then the fault information carried on this msl can be observed at the monitor output immediately (i.e., there is no

information delayed by the clock). If the msl which has no mmsp, then it must have a state monitor signal path (smsp), so the fault information on this msl can be observed at the corresponding primary output by Theorem 3.5. Therefore, the statement of the theorem is valid.

There are many ways to set the control signals so that they will convert the modified network to satisfy the topological constraints of Theorem 2.5 and the test sequence requirements developed in section 3.3. Figure 3.15 shows another control signal setting of the same MSSN of the above example. It is obvious that the latter control signal setting will cost more hardware.

An inherent property of the MSSN is that the control signals of CMM's can be determined at the final stage of design since it will neither increase the cost nor save any hardware. This property will be used to develop the systematic design procedure in the next section.

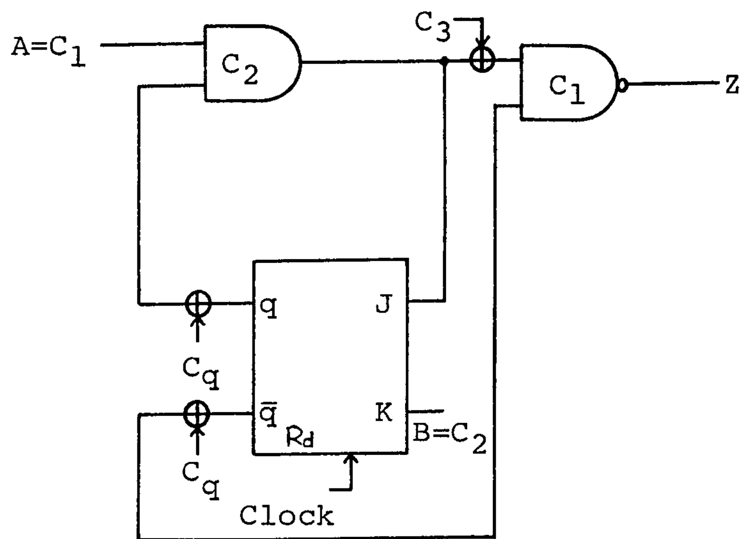


Figure 3.15

Another Control Signal Setting of Network Shown in Figure 3.13

3.6 Systematic Design Algorithm for Synchronous Sequential Network

The structure of a general synchronous sequential network can be represented as shown in Figure 3.16. The combinational block can be partitioned into two blocks -- combinational blocks A(CB_A) and combinational block B (CB_B) such that the input of CB_A consists of PIL_a and MSL and the output of CB_A is MIL, and the input of CB_B consists of PILb and MIL and MSL and the output of CB_B is POL (where $PIL_a \cup$ PILb =PIL). The inputs to the memory block are MIL and the outputs of the memory block are MSL.

Let the set of all the signal nodes in CB_A be denoted as SN_A and that in CB_B as SN_B.

A signal node in SN_A is called the <u>axis signal node</u> (asn) if there exists a signal line from this node to a memory node. The set of all the asn in SN_A is denoted as ASN.

In the algorithm, the level of each signal node in SN_B is determined according to the level of node G,L(G), defined in section 2.6. The level of each signal node in SN_A is determined in the same manner by treating each asn as the primary output node (i.e., L(g)=1 for all the g in ASN).

Let the set of all OR, NOR, AND, NAND gates in the level i of CB_A, CB_B be denoted as AAi, BAi respectively, and all the signal lines incident into AAi, BAi is denoted as the set SAAi, SBAi respectively.

Let the set of all NOT gates in level i of CB_A, CB_B be denoted as AB_i,BB_i respectively, and all the signal

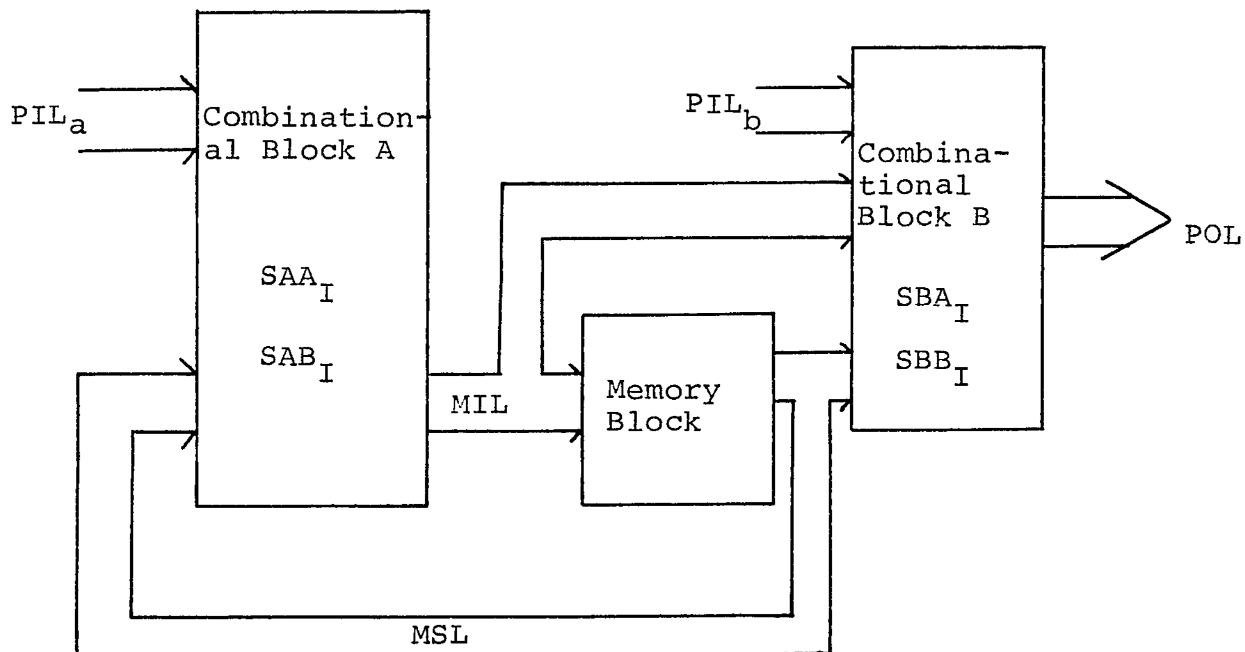


Figure 3.16
The Topology of a General Synchronous Sequential Network

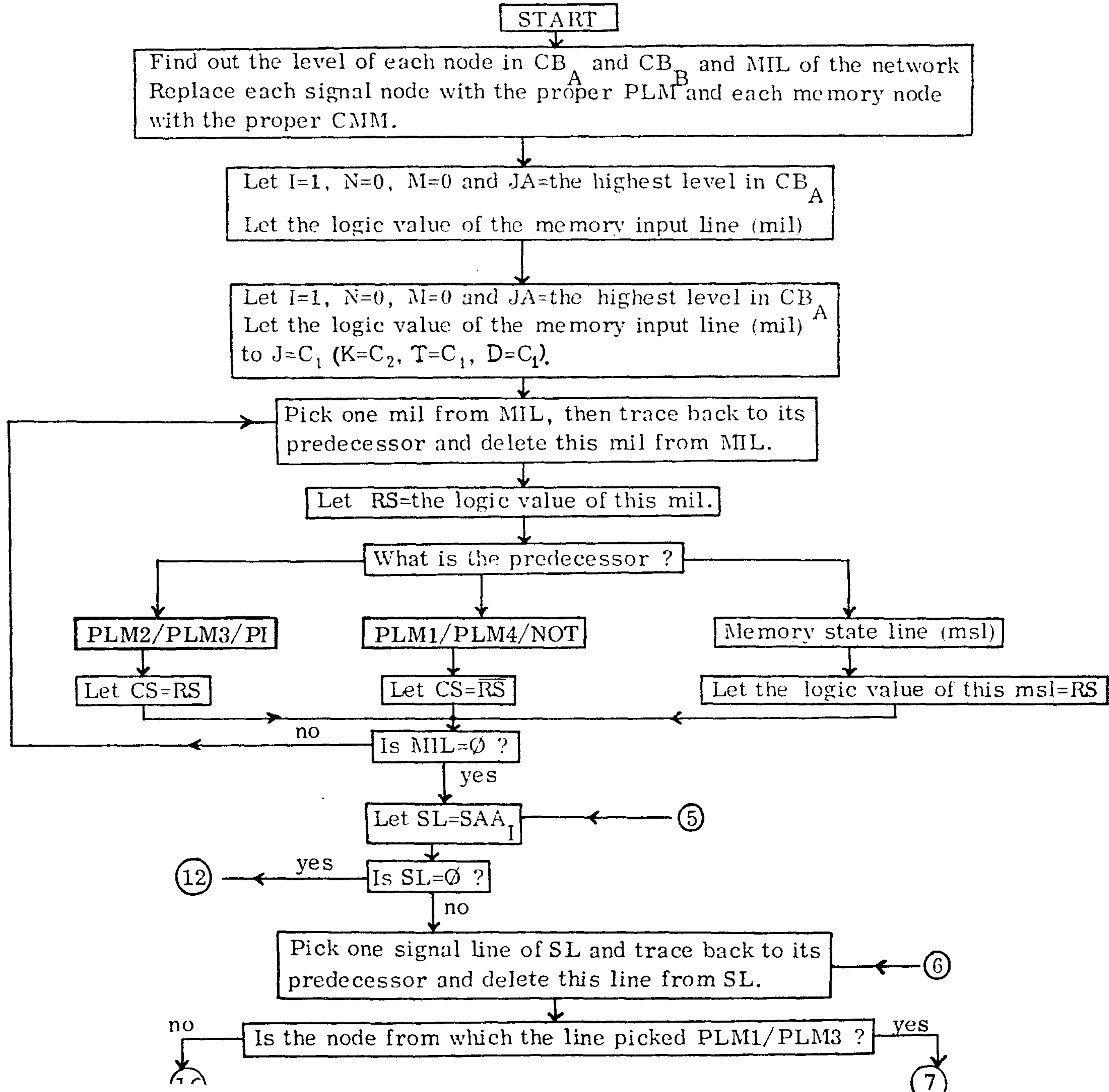
lines incident into AB_i, BB_i is denoted as SAB_i, SBB_i respectively.

An algorithm of the systematic design procedure for synchronous sequential network is developed based on the results of the previous section. It will make the network satisfy the topological constraints posed by theorem 3.5 and theorem 3.6 so that four tests will detect the stuck-at-faults and malfunctions of the network. Since four test require four clock pulses, therefore, the network can be considered as a real-time-fault detectable network. The flow chart is shown in Figure 3.17 and the details of the procedure are described below.

Systematic Design of Algorithm

Step 1. Determine the level of each signal node in CB_A and CB_B separately and the set of MIL of all the memory input lines of network. Then substitute each signal node with a proper PLM and each memory node with a proper CMM (i.e., AND $\longrightarrow$ PLM1, OR $\longrightarrow$ PLM2, NAND $\longrightarrow$ PLM3, NOR $\longrightarrow$ PLM4, DFF $\longrightarrow$ DCMM, TFF $\longrightarrow$ TCMM, and JKFF $\longrightarrow$ JKCMM). Denote the control signal of the predecessor node or the logic value of a memory state line (msl) as CP and the reference control signal as RS.

Step 2. Let I=1, N=O, M=0, and JA=the maximum level of the sub network CB_A. Let the logic value of the memory input line (mil) to $J=C_1$ ($K=C_2$, $T=C_1$, $D=C_1$).


START
Find out the level of each node in CB_A and CB_B and MIL of the network
Replace each signal node with the proper PLM and each memory node with the proper CMM.
Let I=1, N=0, M=0 and JA=the highest level in CB_A
Let the logic value of the memory input line (mil)
Let I=1, N=0, M=0 and JA=the highest level in CB_A
Let the logic value of the memory input line (mil)
to J=C1 (K=C2, T=C1, D=C1).
Pick one mil from MIL, then trace back to its predecessor and delete this mil from MIL.
Let RS=the logic value of this mil.
What is the predecessor ?
PLM2/PLM3/PI
PLM1/PLM4/NOT
Memory state line (msl)
Let CS=RS
Let CS=RS
Let the logic value of this msl=RS
no
Is MIL=Ø ?
yes
Let SL=SAA_I
5
yes
12
Is SL=Ø ?
no
Pick one signal line of SL and trace back to its predecessor and delete this line from SL.
6
no
Is the node from which the line picked PLM1/PLM3 ?
yes
7

(16)

Let RS=CS of this node

Is CP defined or not ?

yes

no

What is the predecessor ?

PLM2/PLM3/PL/msl

PLM1/PLM4/NOT

CP=RS ?

yes

no

CP=RS ?

yes

no

What is the predecessor ?

PLM2/PLM3/PL/msl

PLM1/PLM4/NOT

Let CP=RS

Let CP=RS

(7)

Let RS=CS of this node

Is CP defined or not ?

yes

no

What is the predecessor ?

PLM2/PLM3/PL/msl

PLM1/PLM4/NOT

CP=RS ?

no

yes

CP=RS ?

no

yes

What is the predecessor ?

PLM2/PLM3/PL/msl

PLM1/PLM4/NOT

Let CP=RS

Let CP=RS

Insert a CSI in series with this line.

Is SL=Ø ?

no → (6)

yes

N=1 ?

yes

← (12)

no

Let $SL=SAB_I$

Is SL=Ø ?

no → Let N=1 → (6)

yes

I >JA?

no → I=I+1, N=0 → (5)

yes

Is M=1 ?

yes → STOP

no

M=M+1

Let I=1, N=0 and JA= the highest level in CB_B.
Let CS of PLM1/PLM3=C_1 (or C_2).
Let CS of PLM2/PLM4/NOT = C_2 (or C_1).
Let $SAA_J=SBA_J$, $SAB_J=SBB_J$ $\forall$ $1 \leq J \leq JA$.

(5)

Figure 3.17
Flow Chart of The Systematic Design Algorithm for Synchronous Sequential Network

Step 3. Pick one mil from MIL, then trace back to its predecessor and delete the line from MIL. Let RS= the logic value of this mil. Then check its predecessor.

Step 4. If the predecessor is

(1) PLM2/PLM3/PI, set its control signal CS=RS,

(2) PLM1/PLM4/NOT, set its control signal CS=$\overline{RS}$,

(3) memory state line (msl), let the logic value of this msl=RS.

Then check if MIL=Ø. If it is, go to next step. Otherwise go back to step 3.

Step 5. Let SL=SSA_I, then check if SL=Ø. If it is, go to step 12. Otherwise go to next step.

Step 6. Pick one signal line of SL and trace back to its predecessor and delete this line from SL. If the node from which the signal line is picked is PLM1/PLM3 node, go to step 7. Otherwise, go to step 16.

Step 7. Let RS be equal to the control signal of this PLM1/PLM3 node. Then check to see whether the control signal of its predecessor node is defined or not. If yes, go to step 8. Otherwise go to step 9.

Step 8. If the predecessor is

(1) PLM2/PLM3/PI/msl, check if CP=RS. If it is, go to step 10. Otherwise go to step 11.

(2) PLM1/PLM4/NOT, check if CP=RS. If it is, go to step 11. Otherwise go to step 10.

Step 9. If the predecessor is

(1) PLM2/PLM3/PI/ms1, set CP to be the complement of RS,

(2) PLM1/PLM4/NOT, set CP to be equal to RS. Then go to step 11.

Step 10. Insert a CSI in series with this signal line. Then go to step 11.

Step 11. Check to see whether the set SL is empty. If yes, go to next step. Otherwise, go back to step 6.

Step 12. Test for N=1. If yes, go to step 13. Otherwise, let $SL=SAB_I$ and check for $SL=\emptyset$. If it is, go to step 13. Otherwise, set N=1, then go back to step 6.

Step 13. Examine if I is larger than JA. If yes, go to the next step. Otherwise, let I=I+1, and then go back to step 5.

Step 14. Test for M=1. If yes, the procedure is completed. Otherwise, let M=1, and then proceed to next step.

Step 15. Let I=1, N=0, and JA= the highest level of the subnetwork CB_B and let CS of PLM1/PLM3 equal to C_1 (or C_2) and CS of PLM2/PLM4/NOT equal to C_2 (or C_1). Then let $SAA_J = SBA_J$, $SAB_J = SBB_J$ for all $1 \leq J \leq JA$. Then go back to step 5.

Step 16. Let RS be equal to the control signal of this PLM2/PLM4/NOT. Then check to see whether the control signal of its predecessor is defined or not. If yes, go to next step. Otherwise, go to step 18.

Step 17. If the predecessor is

(1) PLM2/PLM3/PI/msl, check if CP=RS. If yes, go to step 11. Otherwise, go to step 10.

(2) PLM1/PLM4/NOT, check if CP=RS. If yes, go to step 10. Otherwise, go to step 11.

Step 18. If the predecessor is

(1) PLM2/PLM3/PI/msl, set CP to be equal to RS.

(2) PLM1/PLM4/NOT, set CP to be the complement of RS.

Then go back to step 11.

After the process is completed, the desired logic value of each msl in MSL is defined. The corresponding control signal $Cq(C\bar{q})$ can be determined by the following relations:

$Cq = q \oplus$ the logic value of its msl

$C\bar{q} = \bar{q} \oplus$ the logic value of its msl.

Since there are only two possible logic values (C_1 or C_2) on each msl, therefore the control signal setting for any type of controllable memory module (CMM) will have four possible cases: $C_1 \oplus q$, $C_2 \oplus q$, $C_1 \oplus \bar{q}$, $C_2 \oplus \bar{q}$. Notice that $C_1 \oplus q = C_2 \oplus \bar{q}$ and $C_2 \oplus q = C_1 \oplus \bar{q}$ since $C_1=\bar{C}_2$, so we let control signal Ca denote $C_1 \oplus q$ or $C_2 \oplus \bar{q}$ and let control signal Cb denote $C_1 \oplus \bar{q}$ or $C_2 \oplus q$. Then the resulted network will have at most six control signals (C_1, C_2, C_3, C_4, Ca and Cb), independent of the size of the combinational block and the number of the flip-flops in the network. Besides, the resulting

test input pattern and control signal settings will provide the required test sequence (developed in section 3.3) to each flip-flop simultaneously when the algorithm is finished. If the monitor output is inserted into each memory state line (msl) which has memory to memory signal path (mmsp). The resulting modified synchronous sequential network (MSSN) can be detected by the four tests required by the test sequence developed in section 3.3, based on Theorem 3.6. The real time fault detectable synchronous sequential network is therefore achieved by the proposed systematic design technique.

Systematic Fault Detection Procedure

Fault detection for the modified synchronous sequential network (MSSN) implemented by the design algorithm is easy to apply due to the fact that the test input pattern and all the control signals generated by the algorithm will provide the desired test sequence for detecting the stuck-at-faults and malfunctions of the MSSN by four tests when the Rd=1 pulse is applied to reset all the controllable memory module (CMM) to 0 state. Let the test input pattern (X) correspond to $C_1 C_2$=01, $C_1 C_2$=10 be equal to Xa, Xb respectively. Also, let the stuck-at-faults that can be detected by Xa, Xb be denoted as Ha, Hb respectively. Then the four tests can be performed systematically as described below.

Test 1. Set $C_1C_2C_3C_4CaCb$=01 10 01 and apply the test input pattern X=Xa. If the observed output pattern does not match with the desired one after the clock pulse, then the unmatched output(s) indicate(s) that there exists at least one stuck-at-fault in Ha and/or type A (and/or E) malfunction in the corresponding signal path. Otherwise, the MSSN passes this test, and then go to next test.

Test 2. Set $C_1C_2C_3C_4CaCb$=10 10 10 and apply the test input pattern X=Xb. After the clock pulse, set C_1C_2=01 and X=Xa, then observe the output pattern. If it does not match with the desired output pattern, then the unmatched output(s) indicate(s) that there exists at least one type D (and/or F) malfunction and/or stuck-at-faults in Hb in the corresponding signal path. Otherwise, the MSSN passes this test, and then go to the next test.

Test 3. Set $C_1C_2C_3C_4CaCb$= 10 10 01 and apply the test input pattern X=Xb. If the observed output pattern does not match with the desired one after the clock pulse, then the unmatched output(s) indicate(s) that there exists at least one stuck-at-fault in Hb and/or type B malfunction in the corresponding signal path. Otherwise, the MSSN passes this test, and then go to next test.

Test 4. Set $C_1C_2C_3C_4CaCb$=01 10 10 and apply the test input pattern X=Xa. After the clock pulse, set C_1C_2=10 and X=Xb, then observe the output pattern. If it does not match with the desired output pattern, then the unmatched output(s) indicate(s) that there exist at least one type C malfunction and/or stuck-at-fault in Ha in the corresponding signal path. Otherwise, the MSSN passes this test and the test sequence is completed.

Once the MSSN passes the described four tests, then it is ensured to function correctly since H(H=HaUHb) and all possible malfunctions of the CMM are checked. We then set the control signals $C_1C_2C_3C_4CaCb$=00 01 00 to put the MSSN back to its normal operation (function).

In the following discussion, some examples are shown to demonstrate the proposed systematic design algorithm and the fault detection procedure.

Example 5

Consider a two bits shift-right/shift-left register as shown in Figure 3.4. It is converted to the MSSN as shown in Figure 3.18 according to the design algorithm. The structural characteristics of this network are

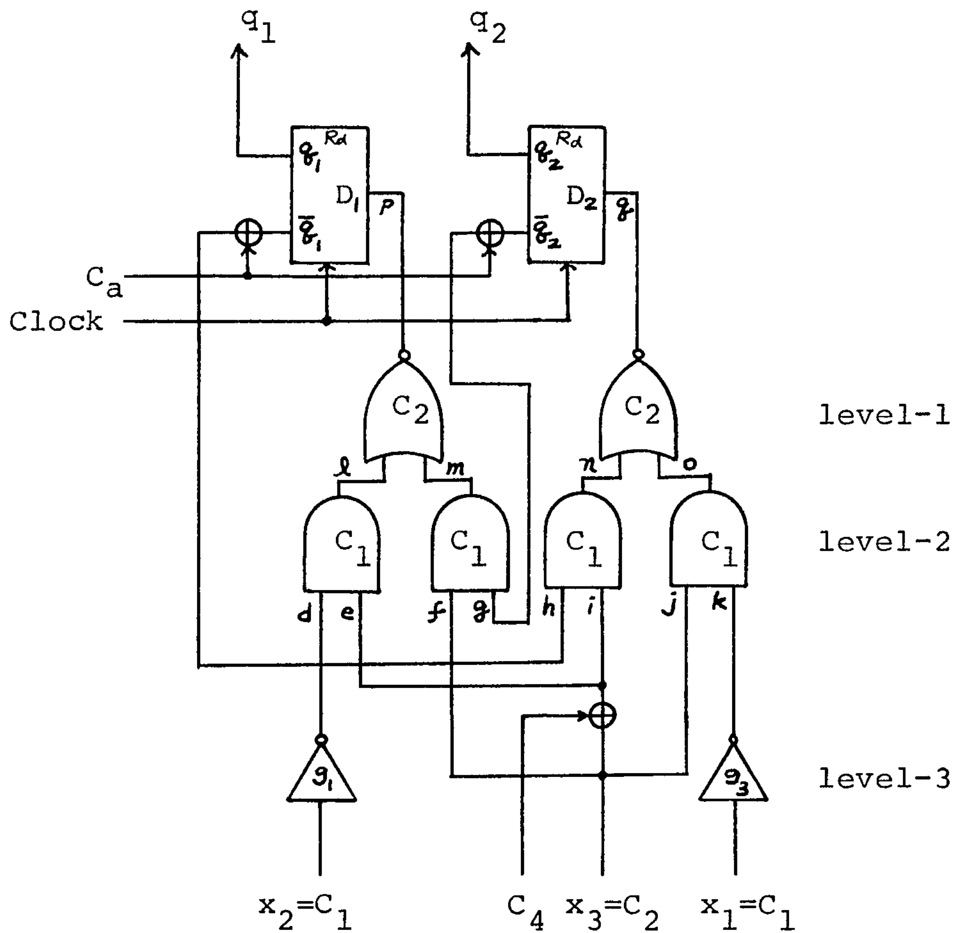


Figure 3.18
Modified Two Bit Shift Right/Shift Left Register

SN_A=SN=[g_1, g_2, ..., g_8, g_9]

SN_B=∅

ASN=[g_8, g_9] L(g_8)=L(g_9)=1

MIL=[p, q]

MSL=[g, h]

MMSP=∅

We first determine the level of each gate in SN_A and then, the sets SAAi and SABi for all $1 \leq i \leq 3$ The level and control signal setting is determined and indicated as shown in the figure by following the design algorithm. If we apply the Rd-1 pulse to reset each DCMM, then the generated test input pattern $X=x_1x_2x_3=C_1C_1C_2$ can provide the desired test sequence to detect the faults in the network. It is explained as follows:

Test 1. Set $C_1C_2C_4$Ca=0100 and apply X=001. This will provide the test D_1D_2=00 simultaneously. The information of stuck-at-faults in Ha where Ha=[a/1, b/0, c/1, d/0, e/0, f/0, g/0, h/0, i/0, j/0, k/0,l/0, m/0, n/0, n/0, o/0, p/1, q/1] will propagate to D_1 and D_2 and force D_1D_2=11. The $DCMM_1$/E and $DCMM_2$/E will also be indicated by y_1y_2=11. If y_1y_2=00, then there is no faults in Ha and $DCMM_1$/E and $DCMM_2$/E. The network passes this test mode and proceed to next test.

Test 2. Set $C_1C_2C_4$Ca=1001 and apply test X=110. This will provide the test D_1D_2=11 simultaneously. The information of stuck-at-faults in Hb where Hb=[a/0, b/1, c/0,

d/1, e/1, f/1, g/1, h/1, i/1, j/1, k/1, l/1, m/1, n/1, o/1, p/0 , q/0] will propagate to D_1 and D_2 and will be sensed by $DCMM_1$,and $DCMM_2$. The state outputs y_1y_2=00 indicates the existence of faults in Hb and $DCMM_1/F$ and $DCMM_2/F$. If y_1y_2=11, then there is no faults in Hb and $DCMM_1/F$ and $DCMM_2/F$.

Once the network passes these two tests, then it is guaranteed to function propertly. The network is then put back to its normal operational mode by setting $C_1C_2C_4Ca$=0010.

Example 6

Consider a 00-01-10-00 sequence detector which is implemented with JKFFs as shown in Fig. 3.19(a). The structured characteristics of this network are

SNA=[g_1, g_2, g_3, g_4, g_5, g_6, g_8, g_9]

SN_B=[g_7]

ASN=[g_3. g_4. g_5. g_6]

MIL= [a,b,c,d]

MSL=[e,f,g]

MMSP= Ø

The network is converted to MSSN as shown in Fig. 3.19(b). The level of each signal node is first determined. Then we can determine the sets SAAi, SABi, SBAi, SBBi as follows:

SAA_1=[g_3, g_4, g_5, g_6]

SAB_1= Ø

SAA_2=[g_1, g_2]

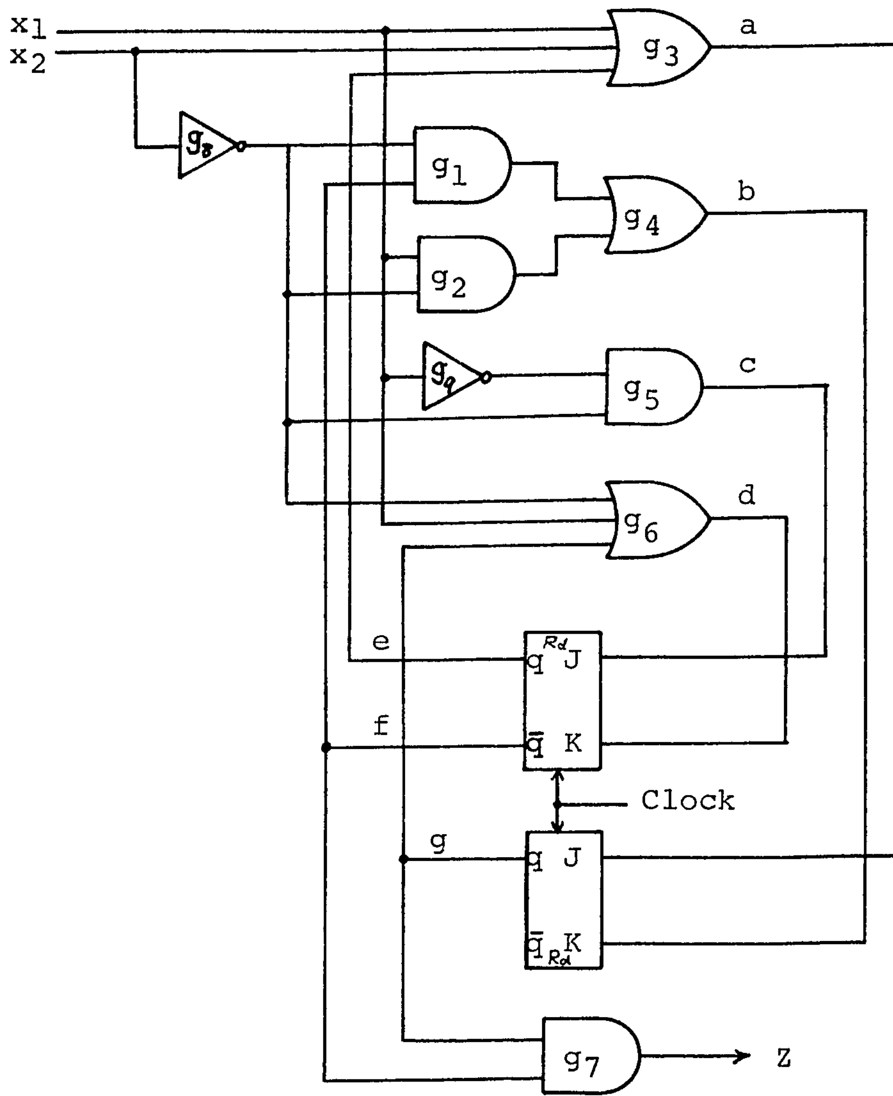
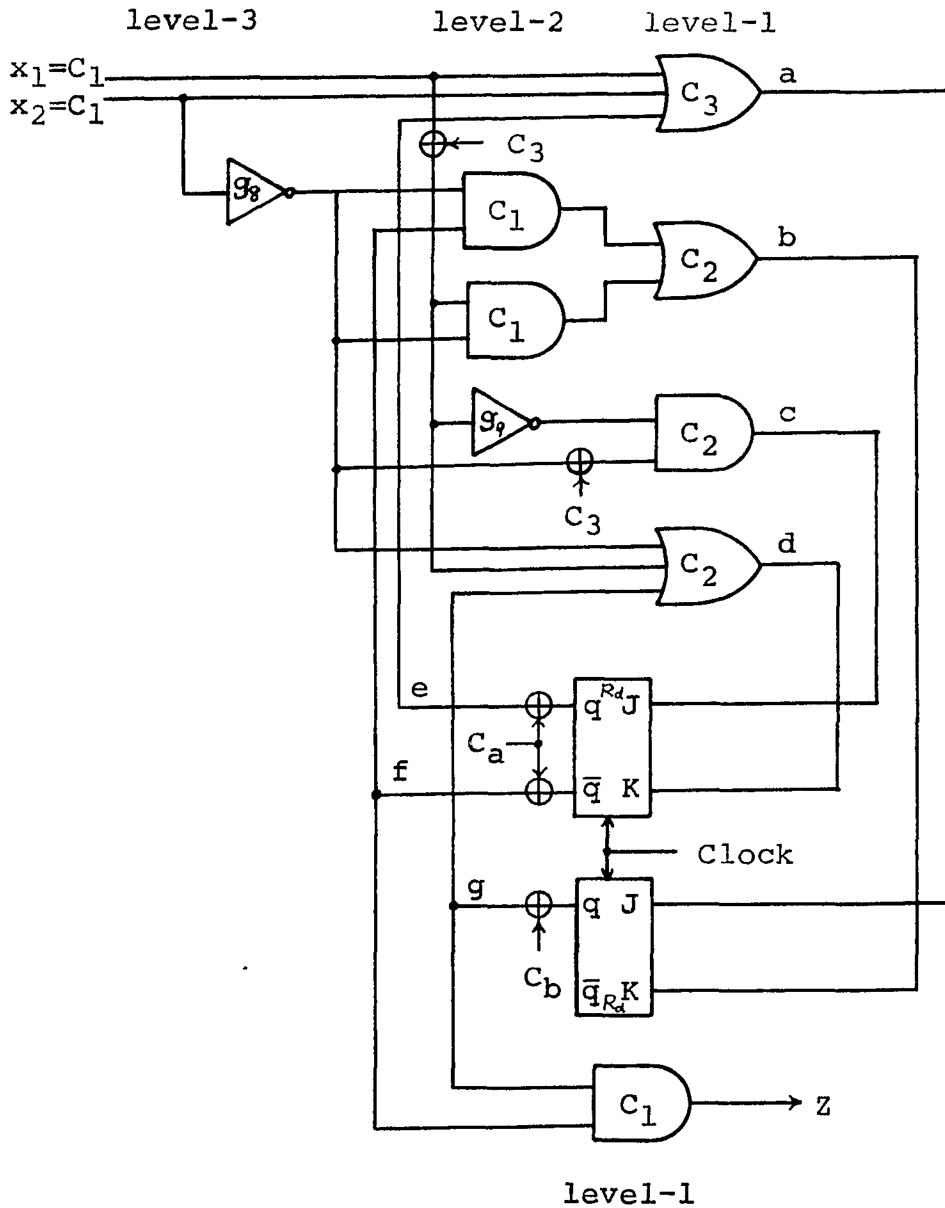


Figure 3.19 (a)



(b)

Figure 3.19

An 00-01-10-00 Sequence Detector and Its Modified Version

$SAB_2=[g_9]$

$SAA_3=\emptyset$

$SAB_3=[g_8]$

$SBA_1=[g_7]$

$SBB_1=\emptyset$

The control signal settings are determined according to the design algorithm and the results are shown in the figure. If we apply the Rd=1 pulse to reset each JKCMM, then the generated test input pattern $X=x_1x_2=C_1C_1$ can provide the proposed test sequence to each JKCMM simultaneously to detect the stuck-at-faults and malfunction of the given network as described below.

<u>Test 1.</u> Set $C_1C_2C_3CaCb=01101$ and apply test X=00. This will supply tests $J_1K_1=01$, $J_2K_2=01$ simultaneously. Then the existence of stuck at faults in Ha and $JKCMM_1/A$, $JKCMM_2/A$ will be indicated by the output Z=0. If Z=1, the network passes this test mode and proceed to next test.

<u>Test 2.</u> Set $C_1C_2C_3CaCb=10110$ and apply test X=11. This will supply tests $J_1K_1=10$, $J_2K_2=10$ simultaneously. Then the existence of stuck-at-faults in Hb and $JKCMM_1/D$, $JKCMM_2/D$ will be indicated by the output Z=0 (when C_1 is set to 0). If Z=1, the network passes this test mode and proceed to next test.

Test 3. Set $C_1C_2C_3CaCb=10101$ and apply test $X=11$. This will provide the tests $J_1K_1=10$, $J_2K_2=10$ simultaneously. The existence of stuck-at-faults in Hb and $JKCMM_1/B$, $JKCMM_2/B$ will be detected by the indication of $Z=1$. If $Z=0$, the network passes this test mode and proceed to next test.

Test 4. Set $C_1C_2C_3CaCb=01110$ and apply test $X=00$. This will provide the tests $J_1K_1=01$, $J_2K_2=01$ simultaneously. The existence of stuck-at-faults in Ha and $JKCMM_1/C$, $JKCMM_2/C$ will be detected by the indication of $Z=1$ (when C_1 is set to 1). If $Z=0$ then the network passes this test mode.

Once the network passes the above four tests, the function of each JKCMM is validated since the proposed test sequence is applied. Therefore the network is guaranteed to function correctly. We then set the network back to its normal function mode by setting $C_1C_2C_3CaCb=00000$.

The examples demonstrate that the design algorithm is easy to apply and can lead to the implementation of a real time fault detection synchronous sequential network.

CHAPTER 4

APPLICATIONS OF TESTABILITY ENCHANCEMENT IN SYSTEM DESIGN

4.1 Introduction

Although extensive research investigations have produced various fault detection techniques for digital networks in recent years [21], [24], [25], [26], [32], [18], [22], [15] and [12]. The problem of system level fault detection has not been considered widely and remain unsolved.

A general approach to the system level design has been the utilization of the triple modular redundancy (TMR) technique which eliminates the need for fault detection. However, the problem inherent in a TMR system is that once two of the three system components fail to function correctly, the system will produce incorrect data which will be interpreted as correct data in the absence of fault information for the components. In some systems, this type of failure may lead to catastrophic system failure. In such application enviroments, it is essential to monitor continously the accuracy of system operation through real time generation and detection of fault information.

In this chapter, the design techniques developed in earlier sections will be extended to the system level design such that the resulting system can be tested in real time for functional failures and insure functional accuracy in

system performance.

In this chapter, a simplified but typical stored program digital processor is introduced. Each functional block in the system is designed by the techniques developed in the preceding chapters. The fault detection of each block is discussed in detail. Finally, a self-contained fault detectable system is organized and fault detection procedure for the system is developed. The example demonstrates that the design algorithm developed in the previous chapters can easily be extended to design a real time fault detectable digital system which will require at most 4M tests for detecting the stuck-at-faults and the malfunctions of the entire system where M is the number of functional blocks contained in the system, and this upper bound is independent of the complexity of the individual functional block.

4.2 System Structure Modeling and Analysis

The structure of any digital system can be partitioned into several blocks according to its specific function. In this section, a specific structure of a stored program digital processor is introduced as an example. The system consists of seven major functional blocks, namely, the accumulator register, the program counter, the instruction register, the control sequence generator, the memory register and the memory data register.

4.2.1 Description of Functional Blocks

1) An n-bit accumulator register, denoted as AR(n), is a synchronous sequential network which comprises a parallel register and a combinational network. The AR(n) is to serve as a central processing register that performs an elementary operation of the incoming data from other register blocks and the data previously stored in its internal register.

2) An m-bit program counter register, denoted as PC(m), is a set of m flip-flops used to indicate the address of the next instruction to be fectched.

3) An n-bit instruction register, denoted as IR(n), is a set of n flip-flops containing the information on current instruction. Its most significant q-bits carry the operation code of the current instruction which is to be decoded by the operation decoder. Its remaining bits specify the address part of the current instruction and is used to fetch the corresponding operand from the memory block if the specified operation is a memory reference type instruction.

4) A control sequence generator, denoted as CSG, consists of a state generator and a combinational network. The outputs of CSG is controlled by the output of the operation decoder and the ouput of the state generator. The CSG provides all the

control gating signals in proper time duration and sequence in the system to perform the necessary register transfer and data modification operations.

5) A k words of n-bit memory block, denoted as M(k,n), consists of k memory registers (each consists of n flip-flops), a memory address decoding network and an output gating network. The M(k,n) operates in random access mode and is used to store the program and data.

6) An m-bit memory address register, denoted as MAR(m), is a set of n flip-flops to retain the address of a memory register whose content is to be fetched or modified.

7) An n-bit memory data register, denoted as MDR(n), is a set of n flip-flops used to maintain the contents fetched fromor written into the memory block.

4.2.2. System Organization and Operation

Based on the defined functional blocks, the system can be organized by the use of data bus and control bus. The architectual configuration of the system is shown in Figure 4.1.

The data bus is a bidirectional bus which is a set of wires that connect each register block of the system and provoide the signal path for transfering the binary data

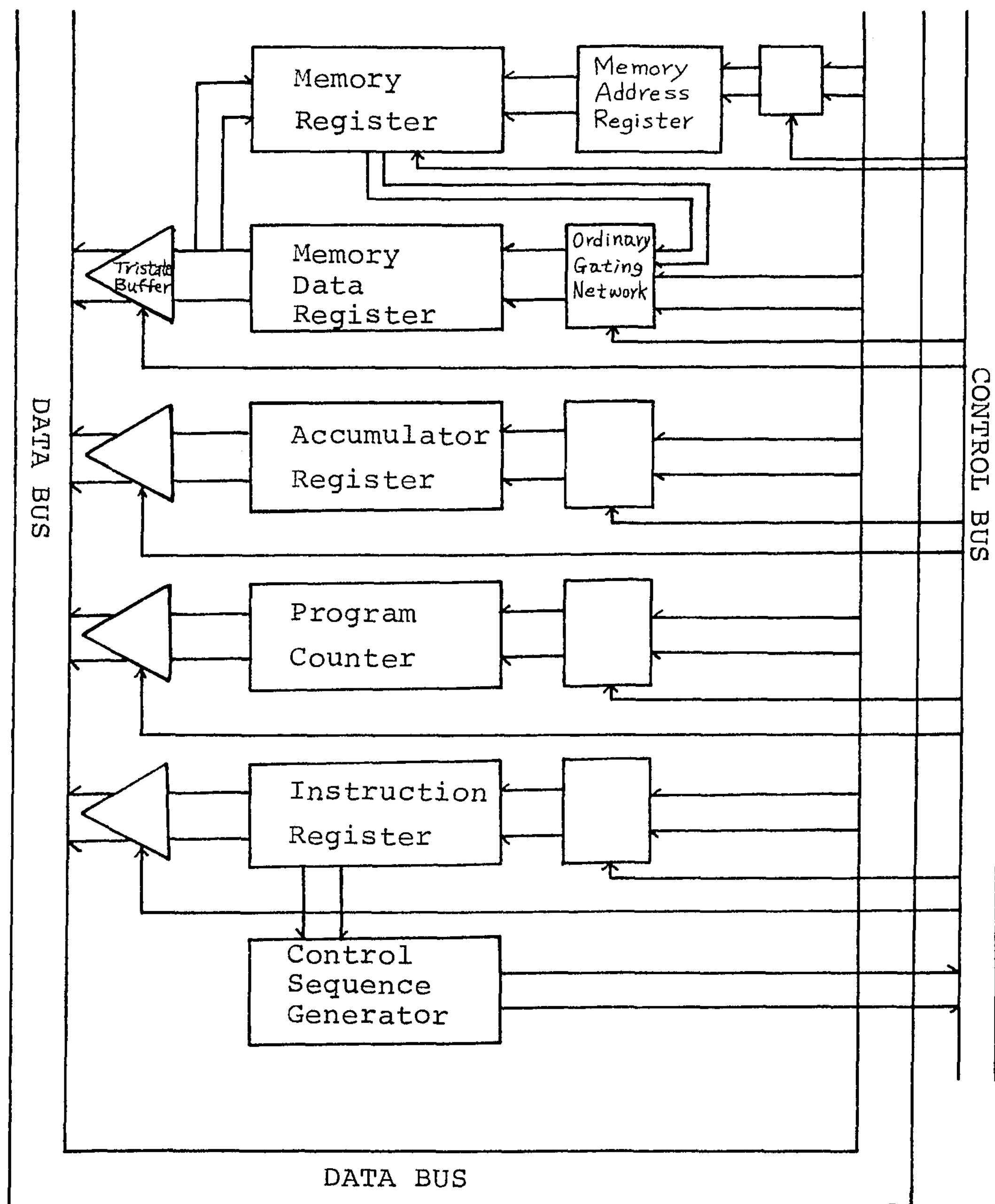


Figure 4.1

A Simplified Stored Program Digital Processor

between the blocks.

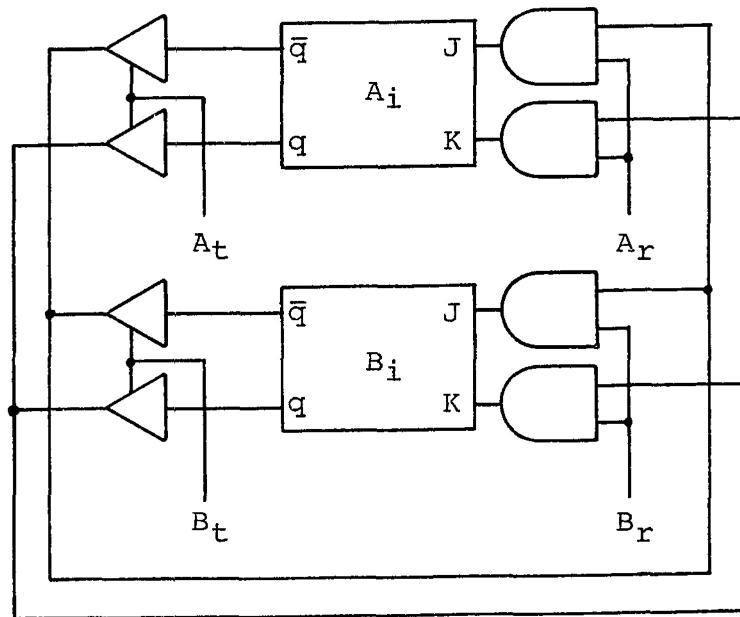
The *control bus* is a set of wires that provide the signal path to send the corresponding control gating signals from the control sequence generator to each functional block of the system.

The use of a data bus restricts us to connect only one register block's output to the bus at any given time. However, the information on the data bus can be storbed into one or more registers as desired at any given time.

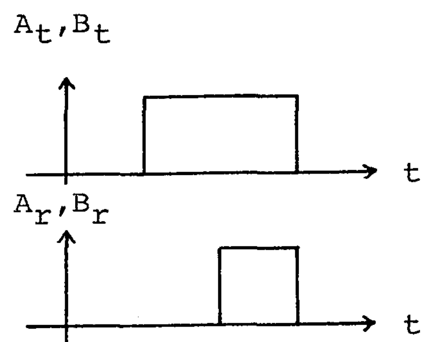
Therefore, the tristate gating network will be used in this example to control the connection between the output of register block and the data bus, and the ordinary gating network is used to pass the data on the data bus to the input of each register block under the control of gating signal from the control sequence generator. A typical configuration of the input and output gating is shown in Figure 4.2(a) and the timing relation between the control commands are shown in Figure 4.2(b).

The following notations are defined to facilitate the latter discussion.

1) M <A> denotes the *memory location* specificed by contents of the register A;
2) (M <A>) denotes the *contents* of the memory location specified by the register A;
3) (A)→B denotes the operation of transfering the



(a)



(b)

Figure 4.2

A Typical Configuration of Bus Structure and Its Timing Relationship

contents of register A into register B;

4) op(IR) denotes the contents of the operation part of IR (instruction register).

Then the time sequence of the system operation can be specified by the state diagram shown in Fig. 4.3. The system has five machine states which form a complete machine cycle. The details of the system operation during each machine state are described below:

1) During state S_1 (g_1=1), the system transfers the contents of program counter (PC) into memory address register (MAR).

2) During state S_2 (g_2=1), the system fetches the contents of memory location specified by MAR into memory data register (MDR) and increments the contents of PC by 1.

3) During state S_3 (g_3=1), the system transfers the contents of MDR into instruction register (IR) and also transfers the address portion of MDR into MAR.

4) During state S_4 (g_4=1), the system performs a transfer or an unary operation, depending upon the operation code part of the instruction register.

   (a) If op(IR)=0(or 1,2,3), transfer the contents of the memory location specified by MAR into MDR.

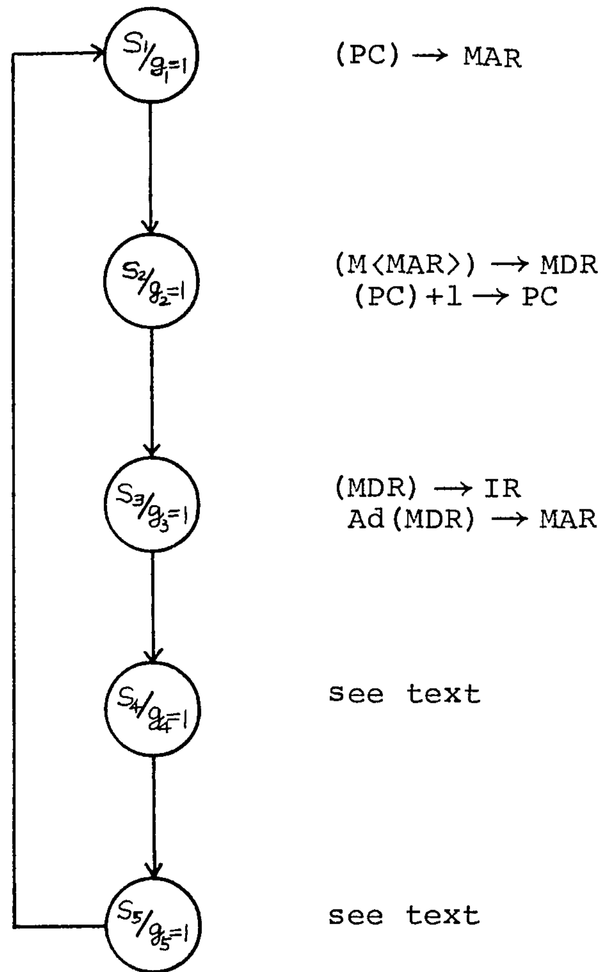


Figure 4.3

State Diagram Illustrates the Sequential Operations of the System

(b) If op(IR)=4, transfer the contents of accumulator register (AR) into MDR.

(c) If op(IR)=5, transfer the address part of MDR into PC. (This is called an unconditional jump).

(d) If op(IR)=6 and (AR)=0, transfer the address part of MDR into PC. (This is called a conditional jump).

(e) If op(IR)=7 and (AR)=0, increment the content of PC by 1. (It is called the conditional skip).

(f) If op(IR)=8, clear the AR.

(g) If op(IR)=9, complement the AR.

(h) If op(IR)=10, increment the AR.

(i) If op(IR)=11, shift the contents of AR to the right by one bit.

(j) op(IR)=12, shift the contents of AR to the left by one bit.

5) During state S_5 (g5=1), the system executes the following binary operation:

(a) If op(IR)=0, add the contents of MDR and AR, then transfer the result into AR.

(b) If op(IR)=1, logically AND the contents of MDR and AR, then transfer the results into AR.

(c) If op(IR)=2, logically OR the contents of MDR and AR, then transfer the result into AR.

(d) If op(IR)=3, logically exclusive-OR the contents of MDR, and AR, then transfer the result into AR.

(c) If op(IR)=4, write the contents of MDR into M<MAR> .

After state S_5, the system goes back to state S_1 and starts the new machine cycle. The description of the above system operations actually defines the design specification of the control gating network to provide the proper command signals in proper time duration for each system operation. Figure 4.4 shows the implementation of the control gating network of the proposed system in which the subscript t means the control signal for transmitting data and the subscript r means the control signal for receiving data.

Since each functional block in the proposed system is a synchronous sequential network, the design techniques developed in the previous chapters can be applied to implement each block so that its stuck-at-faults and malfunctions can be detected in real time.

4.3 Design of Real Time Fault Detection Functional Block

In this section, the structure of each block is analyzed in detail and the design algorithms are applied to implement the real time fault detection functional block and the systematic fault detection procedures are described.

In the following discussion, the Bi_1 (Bi_0) is used to

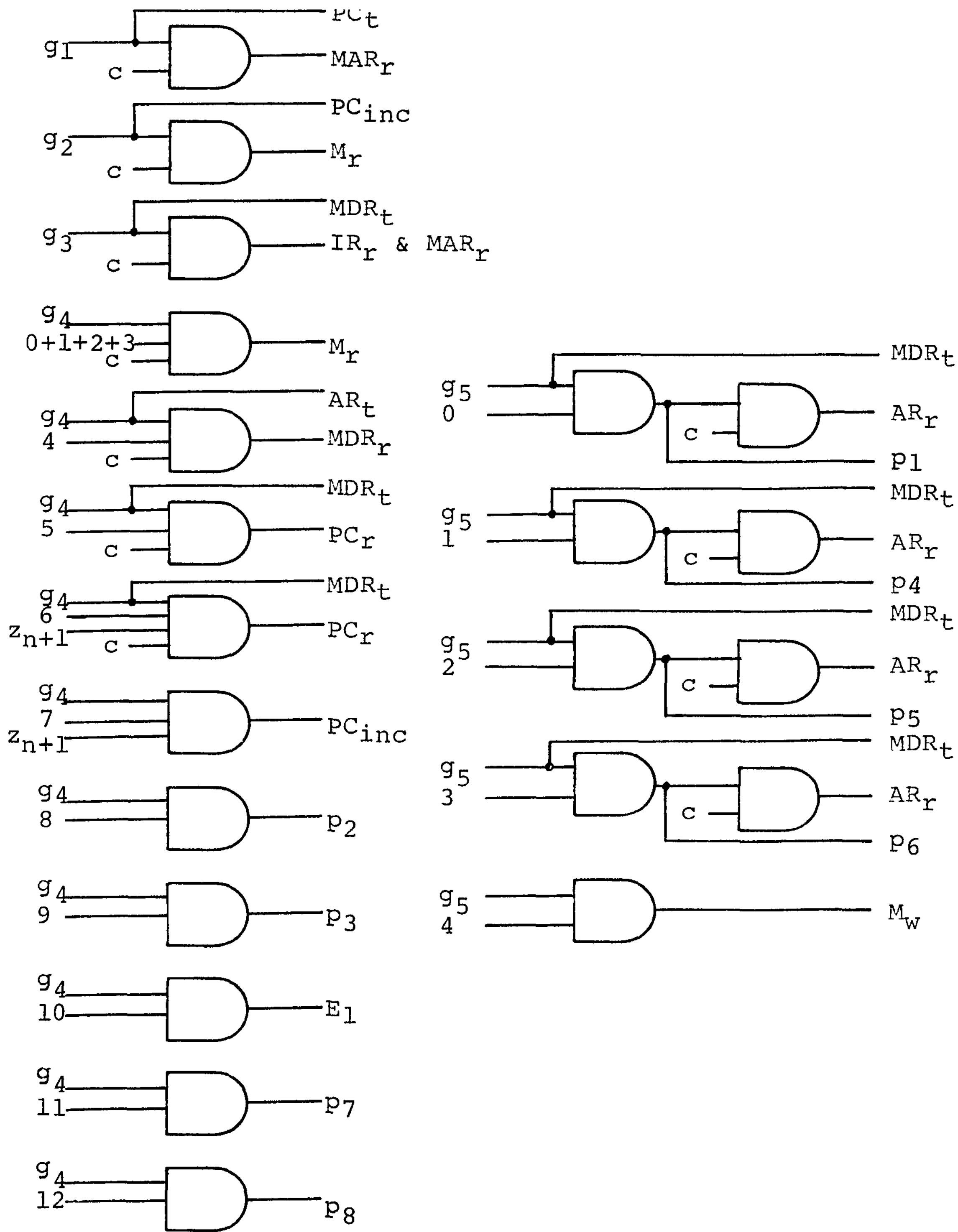


Figure 4.4
A Typical Control Gating Network in Control Sequence Generator

denote the logic value of the bus line which connects the $\bar{q}(q)$ output and J(K) input of JK flip-flop i respectively.

4.3.1 Accumulator Register

An accumulator register AR(n) can be represented by the block diagram shown in Figure 4.5. It consists of a combinational network and a parallel register A. The inputs of AR(n) comprises the data output from register B and a set of control commands from the control sequence generator. The outputs of AR(n) is simply the state ouputs of the internal register A. Figure 4.6 shows the logic diagram of a typical stage (i) of AR(n) which will perform a list of elementary operation defined in Table 4.1 upon the control commands from the control sequence generator. The AR(n) is formed by n such cascaded stages. Each stage is a synchronous sequential network consisting of one JK flip-flop and a combinational network where the input pattern Xi=Bi z_i c_i p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 Ei A_{i-1} A_{i+1}, and the output pattern Zi= Ai $z_{i+1}c_{i+1}$ E_{i+1}. Therfore, we can apply the design algorithm to convert this stage to a real-time detectable network. First, the level of each gate is determined, then the control signal of each gate is decided. Then the test input pattern $Xi=C_1C_2C_1C_1C_1C_1C_1C_1C_1C_1C_1C_1C_1C_1$ and the desired output pattern $Zi=C_1C_2C_1C_1$ are generated. Finally, the control signals of JKCMM are determined. The resulting modified network is shown in Fig. 4.7.

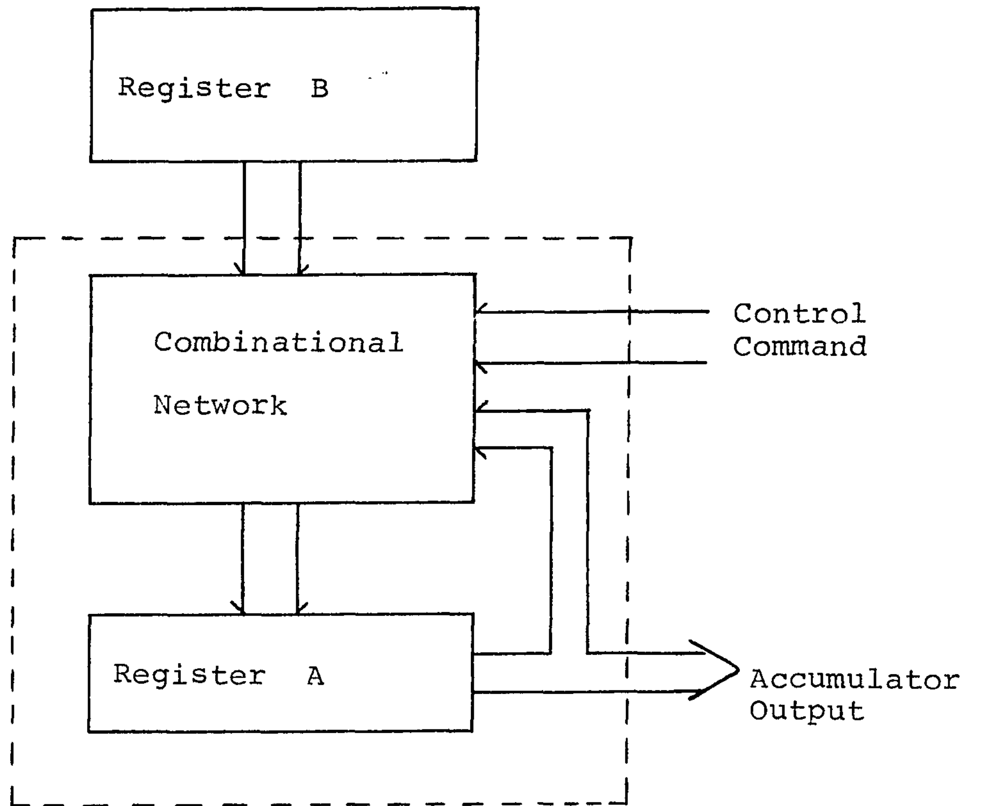


Figure 4.5
Block Diagram of an Accumulator Register

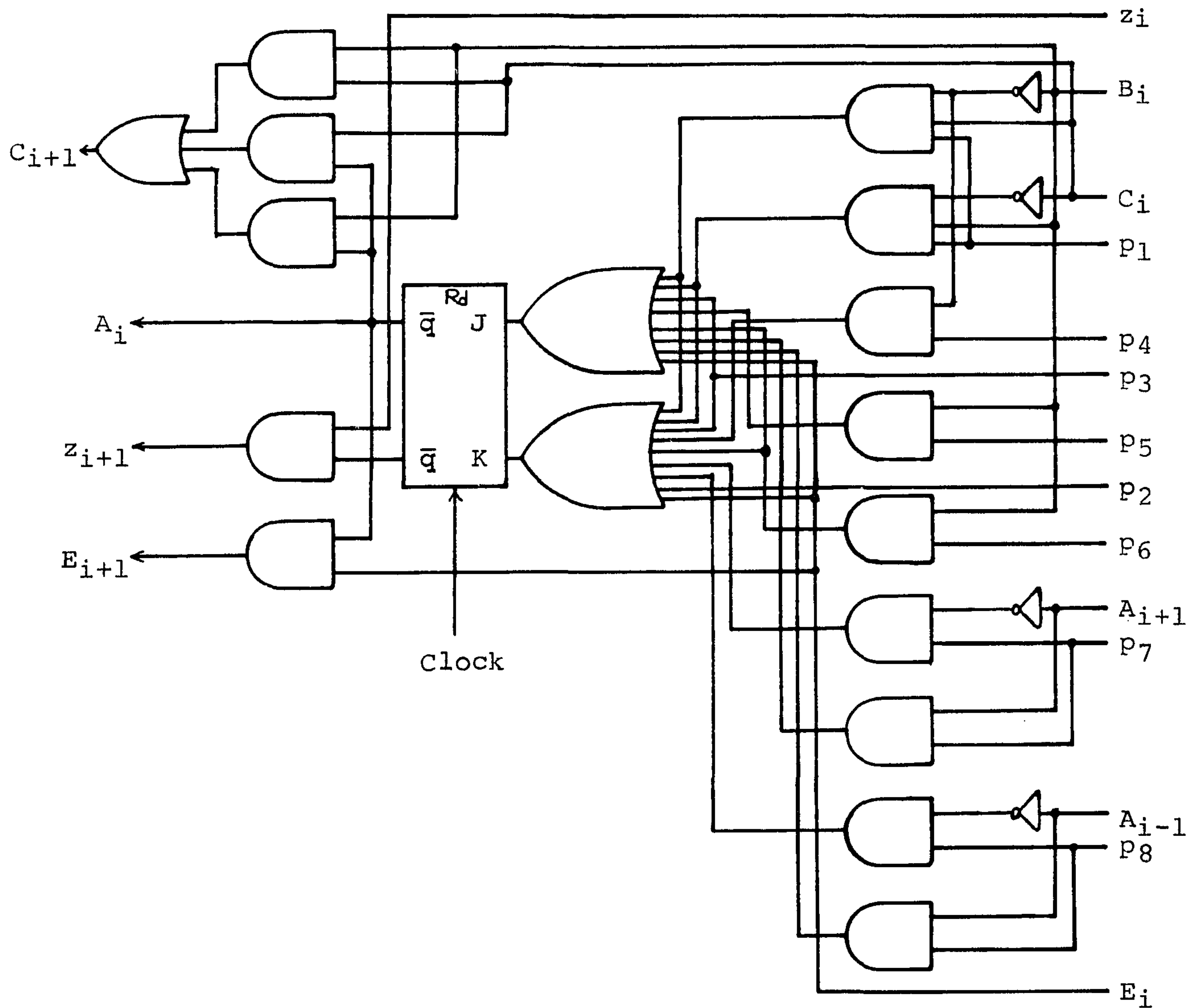


Figure 4.6
An Implementation of Typical Stage of Accumulator Register

Control Signal	Operation	Symbolic Designation
p_1	Arithmetic Addition	$(A)+(B) \longrightarrow A$
p_2	Clear	$0 \longrightarrow A$
p_3	Complement	$(\bar{A}) \longrightarrow A$
p_4	Logical AND	$(A) \wedge (B) \longrightarrow A$
p_5	Logical OR	$(A) \vee (B) \longrightarrow A$
p_6	Exclusive-OR	$(A) \oplus (B) \longrightarrow A$
p_7	Shift-Right	$(A_{i+1}) \rightarrow A_i \quad i=1,2,\ldots,n-1.$
p_8	Shift-Left	$(A_{i-1}) \rightarrow A_i \quad i=2,3,\ldots,n.$
E_1	Increment	$(A)+1 \longrightarrow A$
z_1	Check if Zero	if $(A)=0$, then $z_{n+1}=1$.

Table 4.1
The Elementary Operations of Accumulator Register

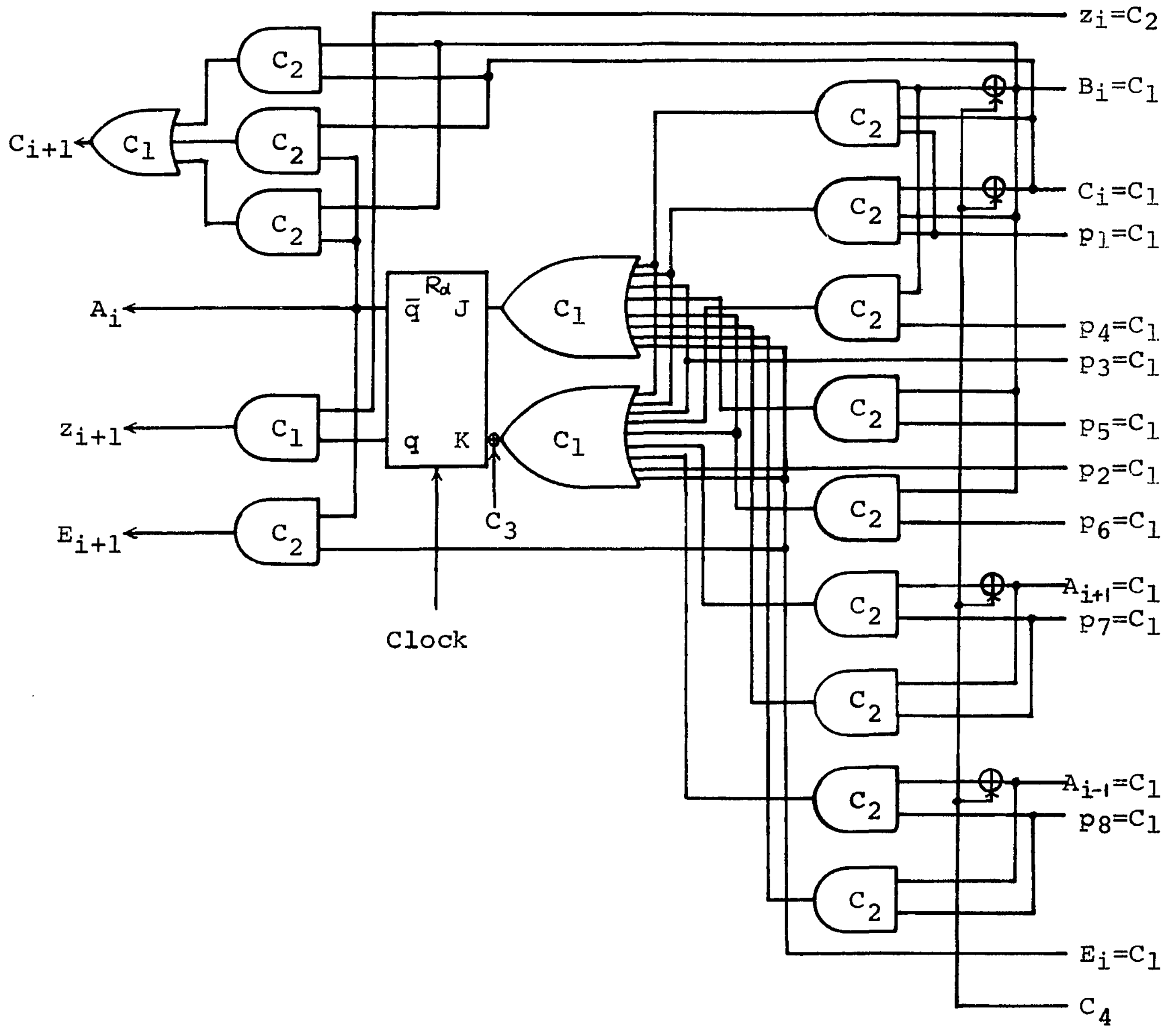


Figure 4.7
A Modified Version of Network Shown in Figure 4.6

Assume that the input pattern Xi is directly accessible and the output pattern Zi is directly observable. The stuck-at-fault and malfunctions of the modified network can now be detected by following the systematic fault detection procedure when the Rd=1 pulse is applied to reset the JKCMM to 0 state:

<u>Test 1</u> Set $C_1C_2C_3C_4C_a$=01100 and apply test input pattern Xi= Xa=010 0000 0000 000. Therefore the test JK=01 and the most sensitive input pattern (MSIP) for each gate are provided. After the system clock pulse, the stuck-at-fault in Ha and type A malfunction can be indicated by observing the output pattern Zi. If Zi $\neq$0100, then there exists at least one stuck-at-fault in Ha and/or type A malfunction in the corresponding signal path. Otherwise, the modified network passes this test and goes to next test.

<u>Test 2</u> Set $C_1C_2C_3C_4C_a$=10101 and apply test input pattern Xi= Xb=101 1111 1111 111. Therefore the test JK=10 and the most sensitive input pattern (MSIP) for each gate are provided. After the system clock pulse, set C_1C_2=01 and Xi=Xa, and then observe the output pattern Zi. If Zi$\neq$0100, then there exists at least one stuck-at-fault in Hb and/or type D malfunction in the corresponding signal path. Otherwise, this test is passed and then proceeds to next test.

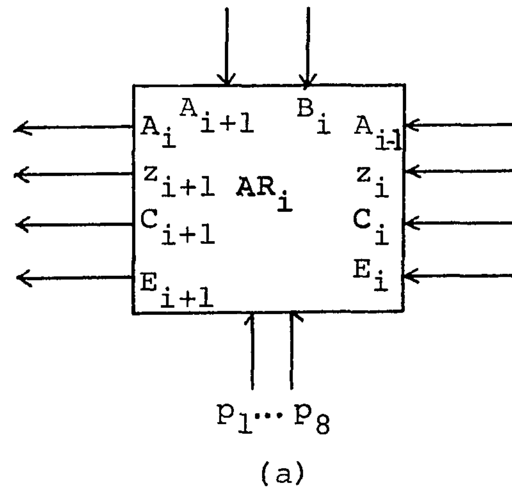
<u>Test 3</u> Set $C_1C_2C_3C_4C_a$=10100 and apply test input pattern Xi=Xb. Therefore the test JK=10 and the MSIP for each gate

are provided. After the system clock pulse is applied, the stuck-at-faults in Hb and type B malfunctions can be detected by observing the output pattern Zi. If Zi$\neq$1011, then there exists at least one stuck-at-fault in Hb and/or type B malfunction in the corresponding signal path. Otherwise, the modified network passes this test and forwards to the last test.

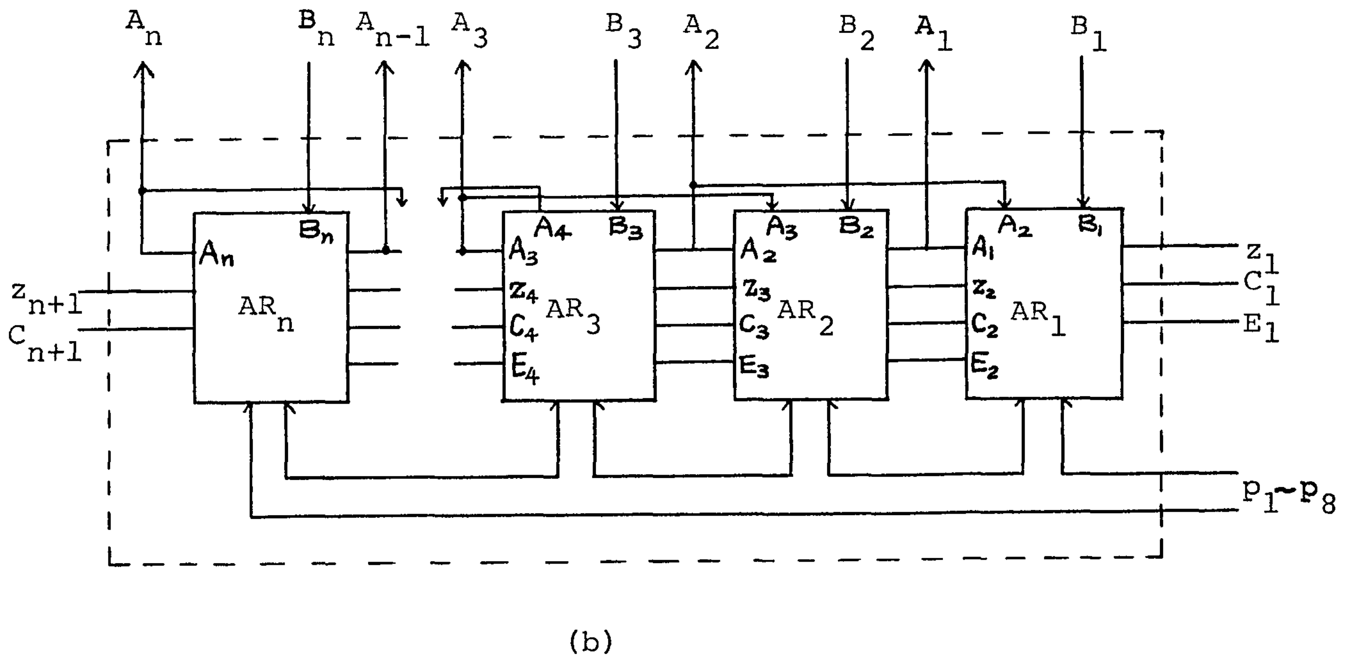
Test 4 Set $C_1C_2C_3C_4C_a$=01101 and apply test input pattern Xi=Xa. Therefore the test JK=01 and the MSIP for each gate are provided. After the system clock pulse is applied, set C_1C_2=10 and Xi=Xb, and then observe the output pattern Zi. If Zi$\neq$1011, then there exists at least one stuck-at-fault in Ha and type C malfunction in the corresponding signal path. Otherwise, the modified network passes this last test and the required test sequence is completed.

Consequently, the modified network is guaranteed to be free of stuck-at-faults and malfunctions, and should perform its normal function properly when the control signals $C_1C_2C_3C_4C_a$ is set to 00010 unless one or more intermittent faults are present.

The logic diagram of Figure 4.6 can be represented by a simplified block diagrm as shown in Figure 4.8(a). Then the complete AR(n) can be constructed by connecting n identical stage in cascade as shown in Figure 4.8(b). The input pattern X of AR(n) is $B_1B_2...B_n, z_1c_1, p_1p_2p_3p_4p_5p_6p_7p_8E_1$ and



(a)



(b)

Figure 4.8
Another Description of the Block Diagram of Accumulator Register

the ouput pattern Z of AR(n) is $A_1\ A_2..An,\ z_{n+1}\ c_{n+1}$.

If we assume that the input pattern X is directly accessible and the output pattern Z is directly observable, then the test input pattern $X=C_1C_1\cdots C_1, C_2C_1, C_1C_1C_1C_1C_1C_1C_1C_1C_1$, and the desired output pattern $Z=C_1C_1..C_1, C_2C_1$ can be generated by the proposed design technique. Then, the modified AR(n) can be detected by four tests if the Rd=1 pulse is first applied to reset each JKCMM to 0 state. The four tests are performed as follows:

Test 1 Set $C_1C_2C_3C_4C_a$=01100 and apply the test input pattern X=Xa=00..0,10, 0000 00000. This will provide the MSIP for each gate and the tests Ji Ki=01 (for all i) simultaneously After the system clock pulse is applied, if the observed output pattern ≠00..0,10, then the existence of at least one stuck-at-fault in Ha and/or type A malfunction in the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified AR(n) passes this test and goes to next test.

Test 2 Se $C_1C_2C_3C_4C_a$=10101 and apply the test input pattern X=Xb=11··· 1,01,1111,11111. This will provide the MSIP for each gate and the tests JiKi=10 (for all i) simultaneously. After the system clock pulse is applied, set C_1C_2=01 and X=Xa, then observe the output pattern Z. If Z≠ 00...0,10, then it indicates the existence of at least one stuck-at-fault in Hb and/or type D malfunction in the signal path which corresponds to the error output bit(s). Otherwise, the modified AR(n)

pass this test and goes to next test.

Test 3 Set $C_1C_2C_3C_4C_a$=10100 and apply the test input pattern X=Xb. This will provide the MSIP for each gate and the tests JiKi=01 (for all i) simultaneously. After the system clock pulse is applied, if the observed output pattern Z$\neq$ 11..1, 01, then the existence of at least one stuck-at-fault in H_b and/or type B malfunction in the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified AR(n) passes this test and goes to next test.

Test 4 Set $C_1C_2C_3C_4C_a$=01101 and apply the test input pattern X=Xa. This will provide the MSIP for each gate and the test JiKi=10 (for all i) simultaneously. After the system clock pulse is applied, set C_1C_2=10 and X=Xb, then observe the output pattern Z. If Z $\neq$11...1,01, then there exists at least one stuck-at-fault in Ha and/or type C malfunction in the signal path which corresponds to the error output bit(s). Otherwise, the modified AR(n) passes this final test and the required test sequence for detecting the stuck-at-faults and malfunctions of the modified AR(n) is completed.

Consequently, the proper function of the modified AR(n) is verified. We then set the control signals $C_1C_2C_3C_4C_a$=00010 to make the modified AR(n) perform its normal function.

Notice that the portion of the required test input patterns, B_1 B_2..B_n, is readily available at the data bus

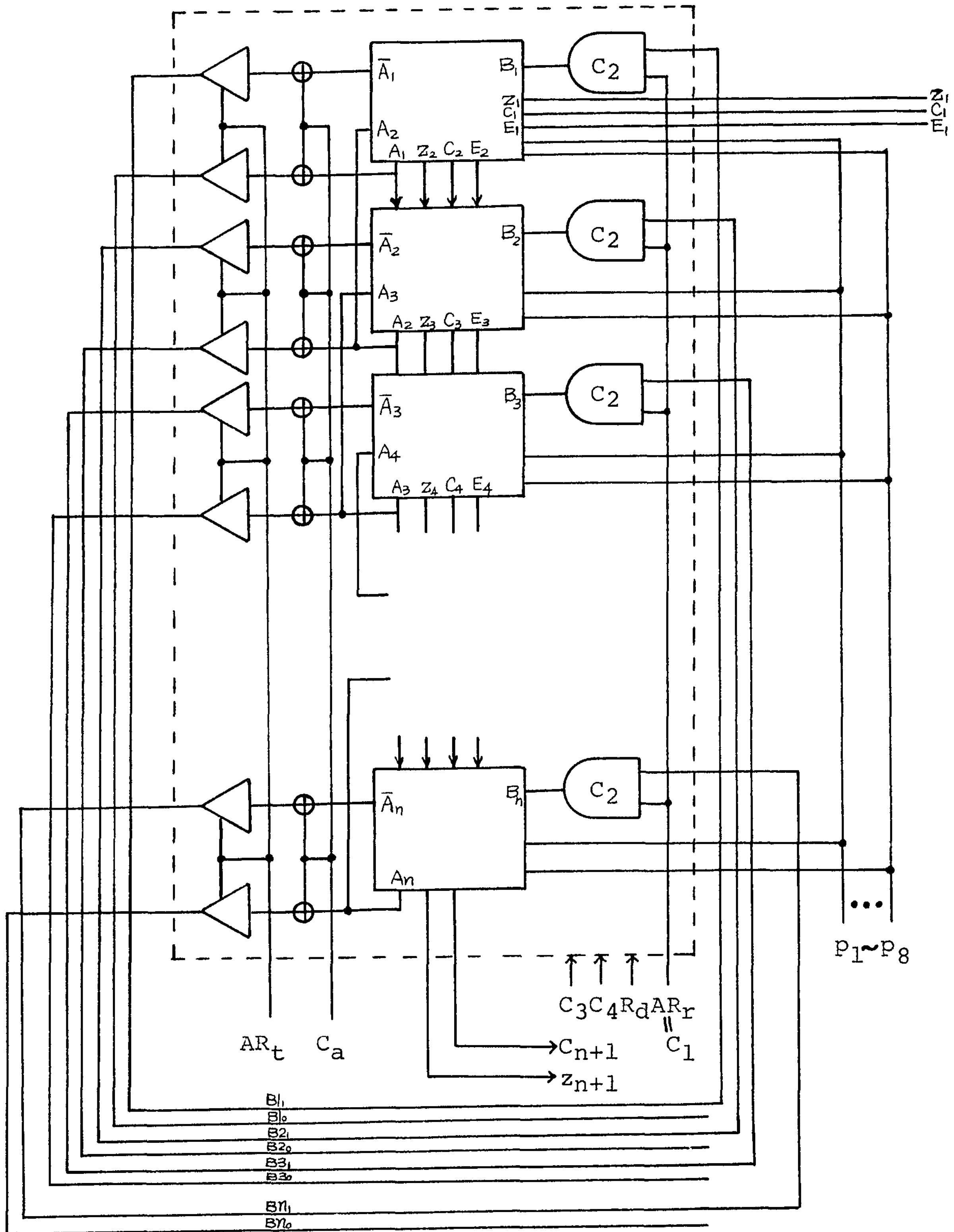


Figure 4.9
A Modified Accumulator Register in the Bus Structure

from the output of the modified AR(n) as shown in Figure 4.9. Therfore, this portion of test input pattern does not need be supplied externally. In addition, the portion of the output pattern, $A_1 A_2 \cdot\cdot An$, is connected to the data bus during the testing of the modified AR(n). Hence the data bus can serve as the monitor for the output $A_1A_2 \cdot\cdot An$. But the data of output $z_{n=1}$ $c_{n=1}$ will need two additional monitor lines to monitor the outputs of Zn+1 and Cn+1. The testing procedure is similar to the one just described except that the test input pattern X is modified to X' (where $X = z_1 c_1 p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8 E = \bar{1} C_2 C_1 C_1 C_1 C_1 C_1$ $C_1 C_1 C_1 C_1 C_1$, $Xa' =$ 10,0000,00000 and $X_b' = 01,1111,11111$) and the control command $AR_t AR_r$ is set to 10,11; 11 and 10 during test 1, 2, 3 and 4 respectively.

4.3.2 Program Counter

The network configuration which satisfies the design specification of the system operation on the program counter PC(m) discussed in section 4.2 is shown in Figure 4.10(a). The PC(m) will hold the address of the next instruction to be fetched and will increment by one when the control command $PC_{inc} = 1$. The PC(m) is a synchronous sequential network, the design algorithm is therefore applied and the resulting modified PC(m) is shown in Figure 4.10(b) in which the proper control signal settings are determined and the test pattern for $PC_r PC_{inc} = C_2 C_1$ is generated. Assume that

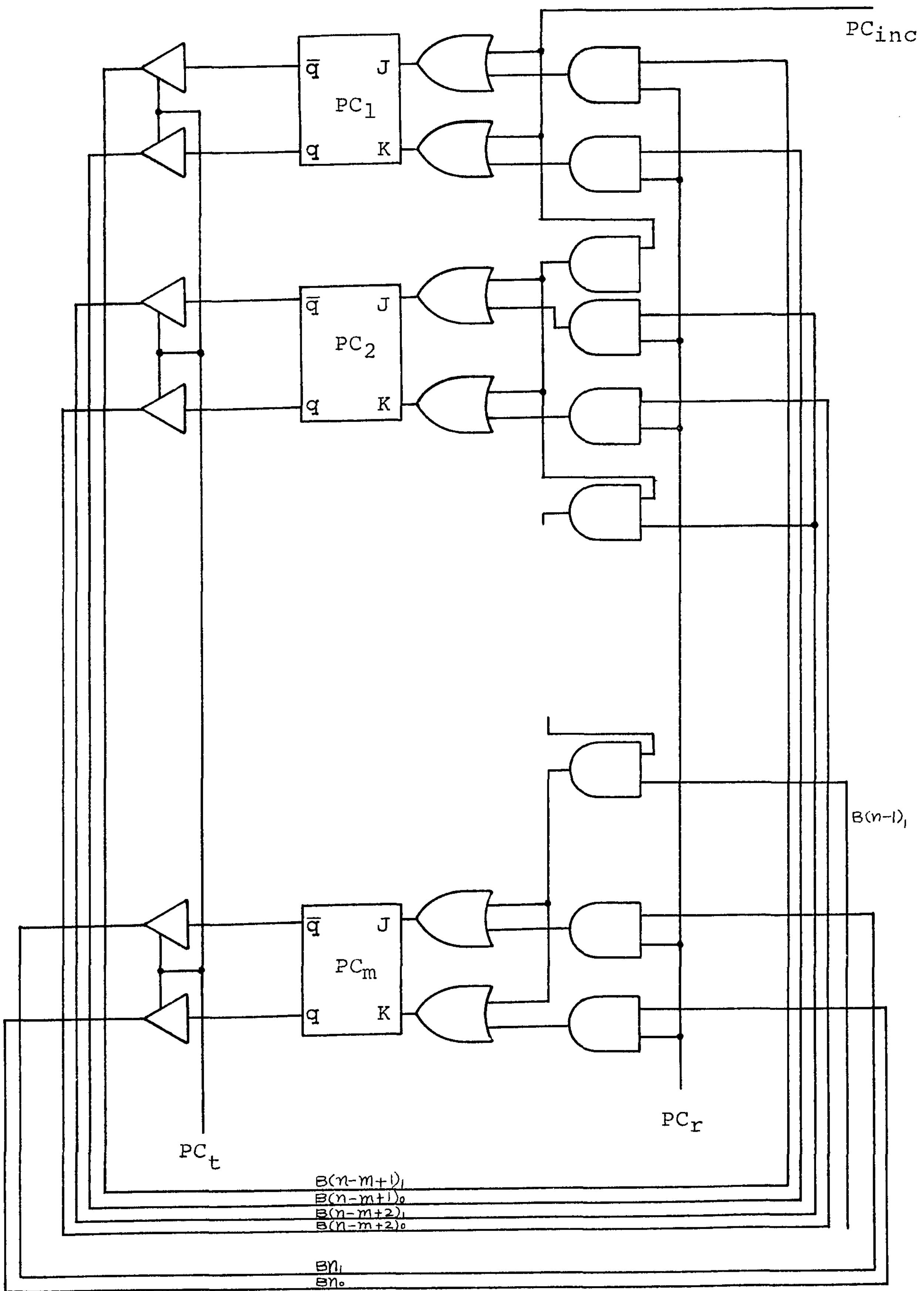


Figure 4.10 (a)

PC$_{inc}$=C$_1$

PC$_1$ PC$_2$ PC$_m$

PC$_t$ C$_a$ C$_3$ PC$_r$=C$_2$

B(n-m+1)$_1$

B(n-m+1)$_0$

B(n-1)$_1$

Bn$_1$

Bn$_0$

(b)

Figure 4.10

A Typical Structure of Program Counter and Its Modified Version

the status on the data bus is directly observable. If we apply the Rd=1 pulse to reset all JKCMM in PC(m) to 0 state, then the stuck-at-faults and malfunctions in PC(m) can be detected by following the systematic fault detection procedure as follows:

Test 1 Set $C_1C_2C_3C_a$=0110 and apply $PC_tPC_rPC_{inc}$=110. This will provide the tests JiKi=01 for all i and the most sensitive input pattern (MSIP) for each gate in this test mode simultaneously. After the system clock pulse is applied, the stuck-at-fault in Ha and type A malfunctions in the modified PC(m) can be detected by observing the status of the data bus. If the observed output pattern on data bus does not match with the desired output pattern, Bi_1Bi_0=01 for all i, then there exists at least one stuck-at-fault in Ha and/or type A malfunction in the signal path which corresponds to the unmatched bit. If the observed pattern matches with the desired one, the modified PC(m) passes this test and proceeds to next test.

Test 2 Set $C_1C_2C_3C_a$=1011 and apply $PC_tPC_rPC_{inc}$ =101. Then the tests JiKi=10 for all i and the MSIP for each gate in this test mode can be provided simultaneously. After the system clock pulse is applied, the stuck-at-faults in H_b and type D malfunctions in the modified PC(m) can be detected by observing the status of the data bus. If the observed output pattern on data bus does not match with the

desired output pattern, $Bi_1Bi_0=01$ for all i, then there exists at least one stuck-at-fault in Hb and/or type D malfunction in the signal path which corresponds to the unmatched bit. If the observed pattern matches with the desired one, the modified PC(m) passes this test and then, proceed to next.

Test 3 Set $C_1C_2C_3C_a=1010$ and apply $PC_tPC_rPC_{inc}=101$. This will provide the test JiKi=10 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, the stuck-at-faults in Hb and type B malfunctions in the modified PC(m) can be detected by observing the status of the data bus. If the observed output pattern on data bus does not match with the desired output pattern, $Bi_1\,Bi_0=10$ for all i, then there exists at least one stuck-at-fault in Hb and/or type D malfunction in the signal path which corresponds to the unmatched bit. If the observed pattern matches with the desired one, the modified PC(m) passes this test and proceeds to next test.

Test 4 Set $C_1C_2C_3C_a=0111$ and apply $PC_rPC_tPC_{inc}=110$. Then the tests JiKi=01 for all i and the MSIP for each gate in this test mode are provided simultaneously. After the system clock pulse is applied, the stuck-at-fault in Ha and type C malfunctions in JKCMM of the modified PC(m) can be detected by observing the status of the data bus. If the observed output pattern on data bus does not match with the

desired output pattern, $Bi_1Bi_0=10$ for all i, then there exists at least one stuck-at-fault in Ha and/or type C malfunction in the signal path which corressponds to the unmatched bit. If the observed pattern matches with the desired one, then the modified PC(m) passes this test and the required test sequence is completed.

Once the modified PC(m) passes the required four tests, the proper function of the modified PC(m) is then verified and the modified PC(m) is ensured to perform its normal function correctly when we set $C_1C_2C_3C_a=0000$.

4.3.3 Instruction Register

A typical structure of the instruction register IR(n) is shown in Figure 4.11(a) in which the outputs of IR(n) are connected to the data bus by a set of tristate gates controlled by the command signal IRt, and also directly gated into the operation decoder. The data on the bus is gated into the input of IR(n) by a set of ordinary gates controlled by the command signal IRr. The IR(n) is also a synchronous sequential network. Assume that the status of the data bus and the output of the operation decoder (OD) are directly observable, and the control command IR_t and IR_r are directly accessible. Then the IR(n) can be translated to a modified network which is capable of real time fault detection by utilizing the systematic design algorithm. The resulting modified IR(n) and its corresponding control signal settings are shown in Figure 4.11(b).

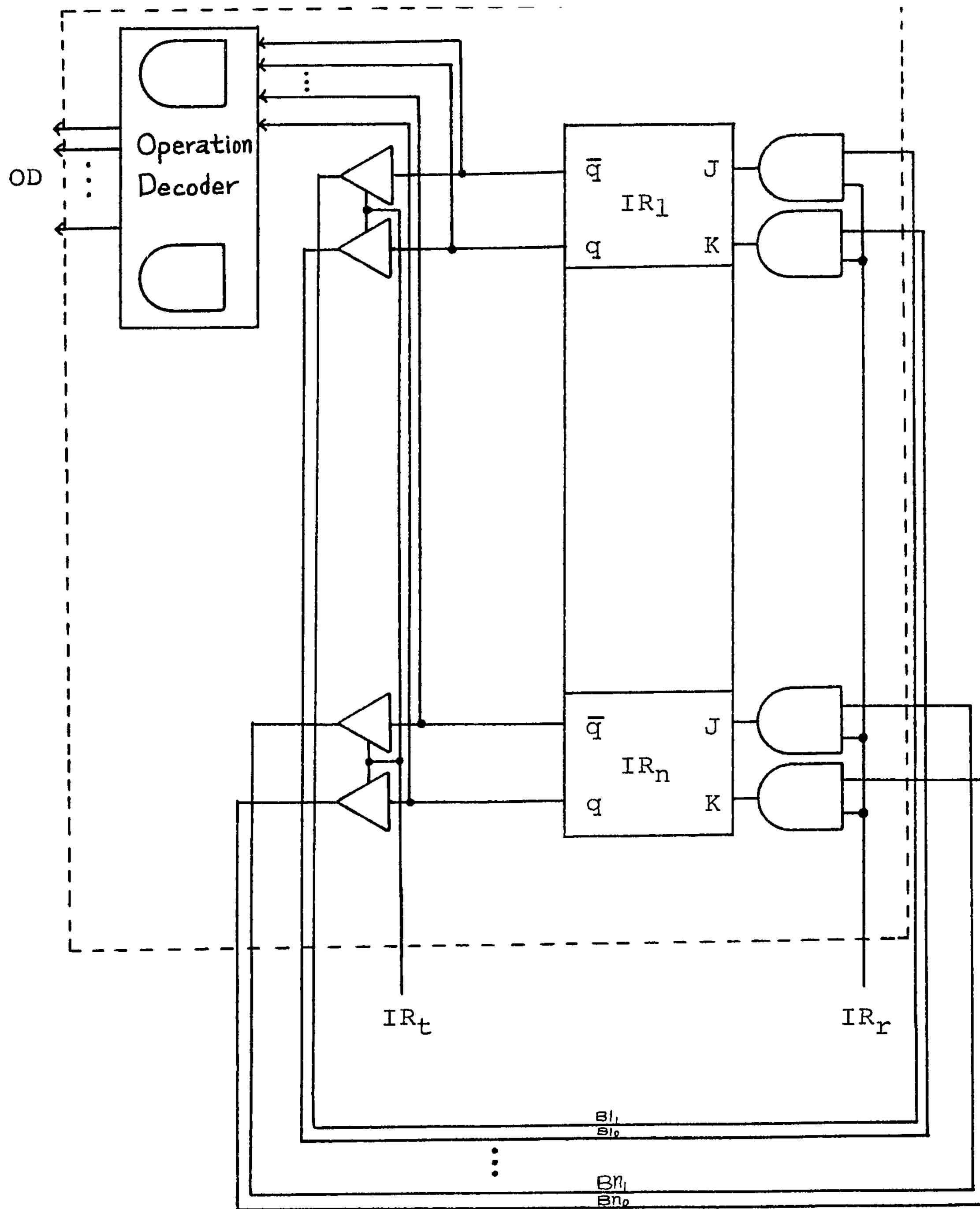


Figure 4.11 (a)

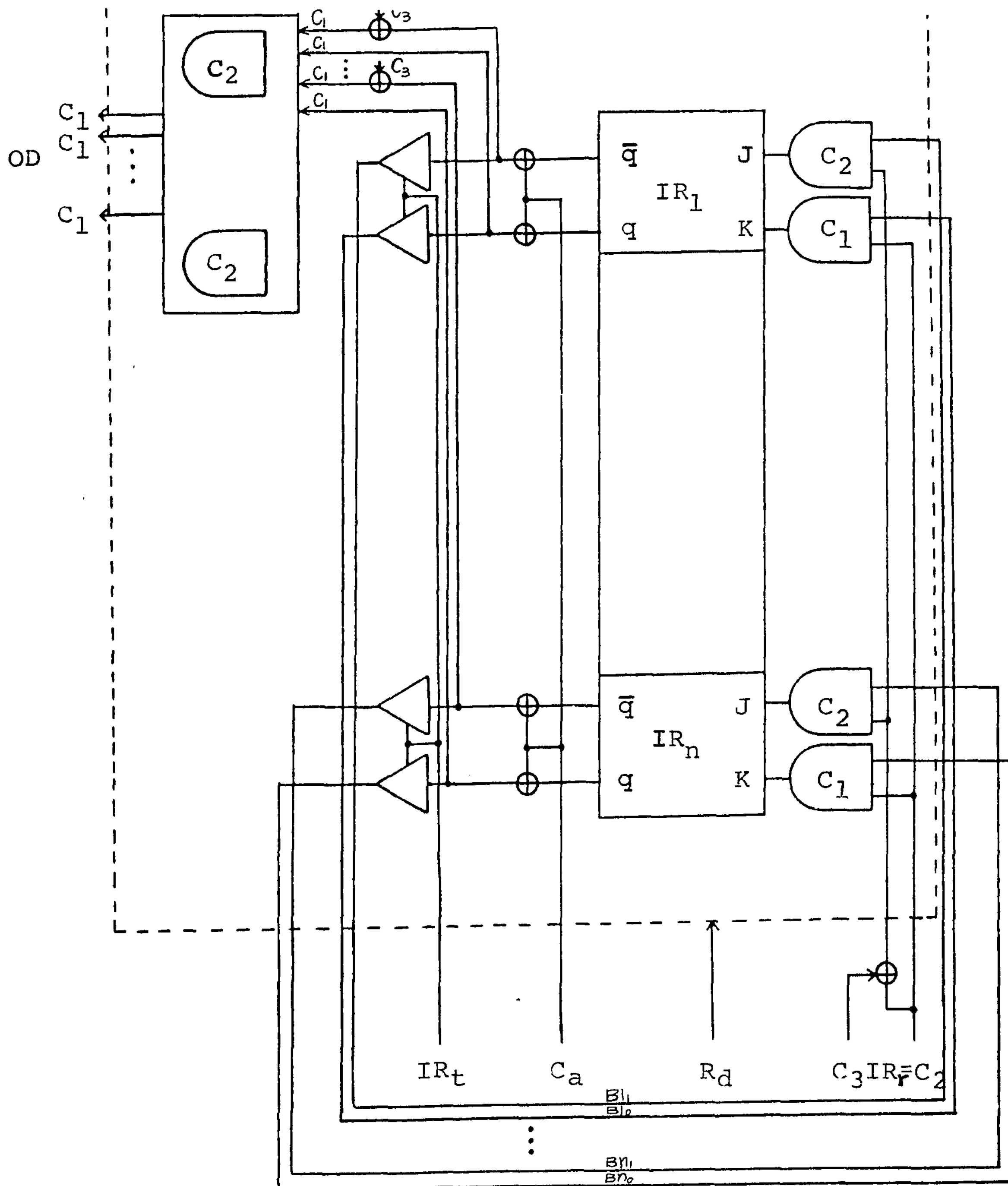


Figure 4.11

A Netwrok Configuration of Instruction Register and Its Modified Version

The test input $IRr=C_2$ and the desired output pattern $OD= C_1C_1..C_1$ are generated as shown. After the Rd=1 pulse is applied to reset each JKCMM in the modified IR(n) to 0 state, then the stuck-at-faults and malfunctions of the modified IR(n) can be detected by the following four tests:

Test 1 Set $C_1C_2C_3C_a=0110$ and apply $IR_t\ IR_r=11$. This will provide the tests JiKi=01 for all i and the most sensitive input pattern (MSIP) for each gate in this test mode simultaneously. After the system clock pulse is applied, the stuck-at-faults in Ha and type A malfunctions in JKCMM of the modified IR(n) can be detected by observing the status of the data bus and the output pattern OD. If the observed output pattern $OD \neq 00...0$, and/or the status of data bus $Bi_1\ Bi_0 \neq 01$ for some i, then the presence of at least one stuck-at-fault in Ha and/or type A malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified IR(n) passes this test and proceeds to next test.

Test 2 Set $C_1C_2C_3C_a=1011$ and apply $IR_t\ IR_r=10$. This will provide the tests JiKi=10 for all i and the most sensitive input pattern (MSIP) for each gate in this test mode simultaneously. After the system clock pulse is applied, set $C_1C_2=01$, then observe the status of the data bus and the output pattern (OD). If the observed output pattern $OD \neq 00..0$,

and/or the status of data bus $Bi_1\ Bi_0 \neq 01$ for some i, then the presence of at least one stuck-at-fault in Hb and/or type D malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified IR(n) passes this test and proceeds to next test.

<u>Test 3</u> Set $C_1C_2C_3C_a=1010$ and apply $IR_t\ IR_r=10$. This will provide the tests JiKi=10 for all i and the most sensitive input pattern (MSIP) for each gate in this test mode simultaneously. After the system clock pulse is applied, the stuck-at-fault in Hb and type B malfunctions in JKCMM of the modified IR(n) can be detected by observing the status of the data bus and the output pattern (OD). If the observed output pattern $OD \neq 11..1$ and/or the status of the data bus $Bi_1\ Bi_0 \neq 10$ for some i, then the presence of at least one stuck-at-fault in Hb and/or type B malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified IR(n) passes this test and proceeds to next test.

<u>Test 4</u> Set $C_1C_2C_3C_a=0111$ and apply $IR_tIR_r=11$. This will provide the tests JiKi=01 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, set $C_1C_2=10$, then observe the status of the data bus and the output pattern OD. If the observed output pattern $OD \neq 11..1$, and/or the status of the

data bus $Bi_1\ Bi_0 \neq 10$ for some i, then the presence of at least one stuck-at-fault in Ha and/or type C malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified IR(n) passes this test and the required test sequence for detecting the stuck-at-faults and malfunctions of the modified IR(n) is completed.

Once the modified IR(n) passes the above four tests, it is guaranteed to perform its normal function correctly when the control signals $C_1C_2C_3C_a$ is set to 0000 since its proper function is validated by the required test sequence.

4.3.4 Control Sequence Generator

The block diagram of the control sequence generator (CSG) is shown in Figure 4.12 which consists of a state generator and a combinational gating network. The state generator is actually a ring counter generating the desired machine states. For the proposed system, a five stage ring counter is required to generate the five machine states. The combinational network is used to gate the output of operation decoder and state output of the ring counter to form a set of control command signals to control the specified system operation.

A five stage ring counter is shown in Fig. 4.13(a). The design algorithm is then applied to convert it to a modified network as shown in Fig. 4.13(b). Assume all the state outputs ($Y = y_1 y_2 y_3 y_4 y_5$) are directly observable, then the proper

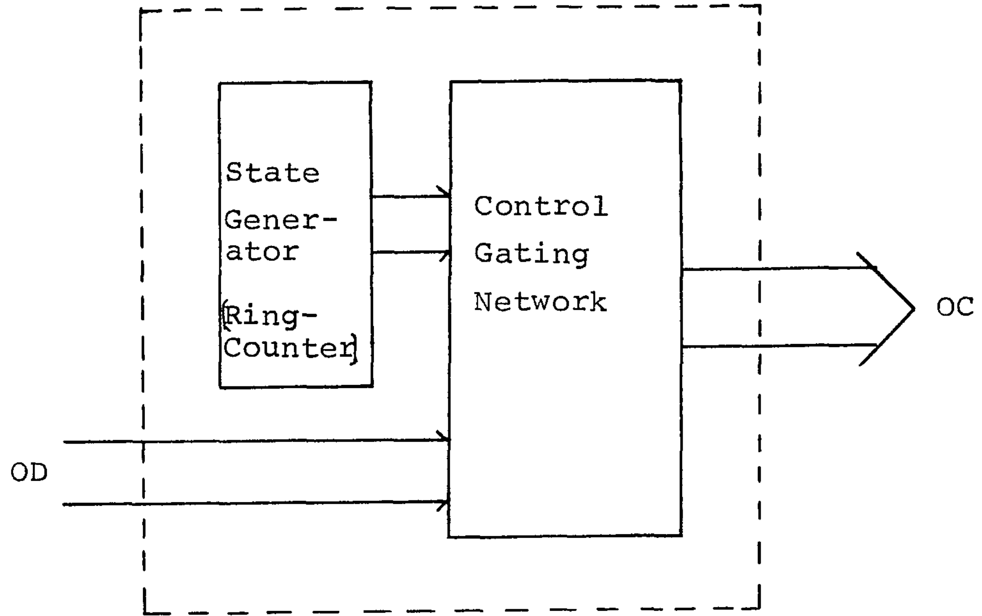
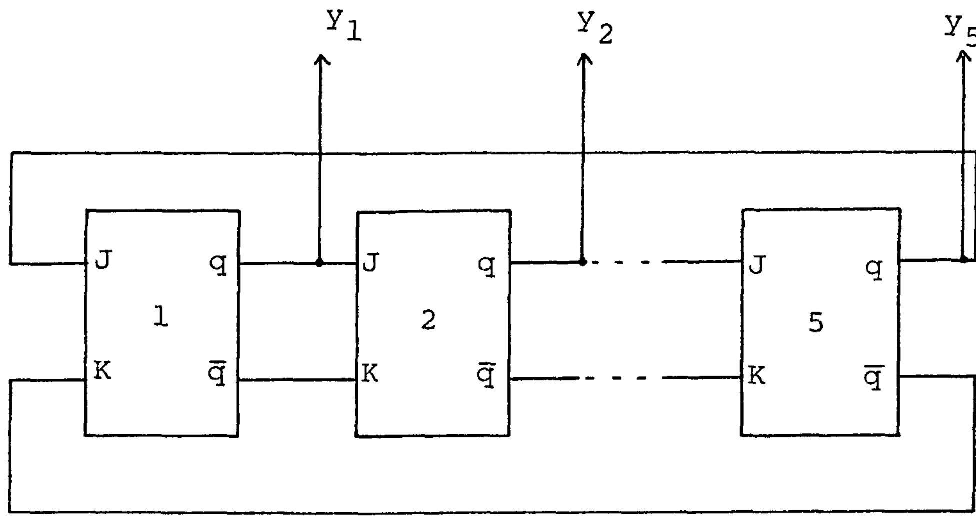
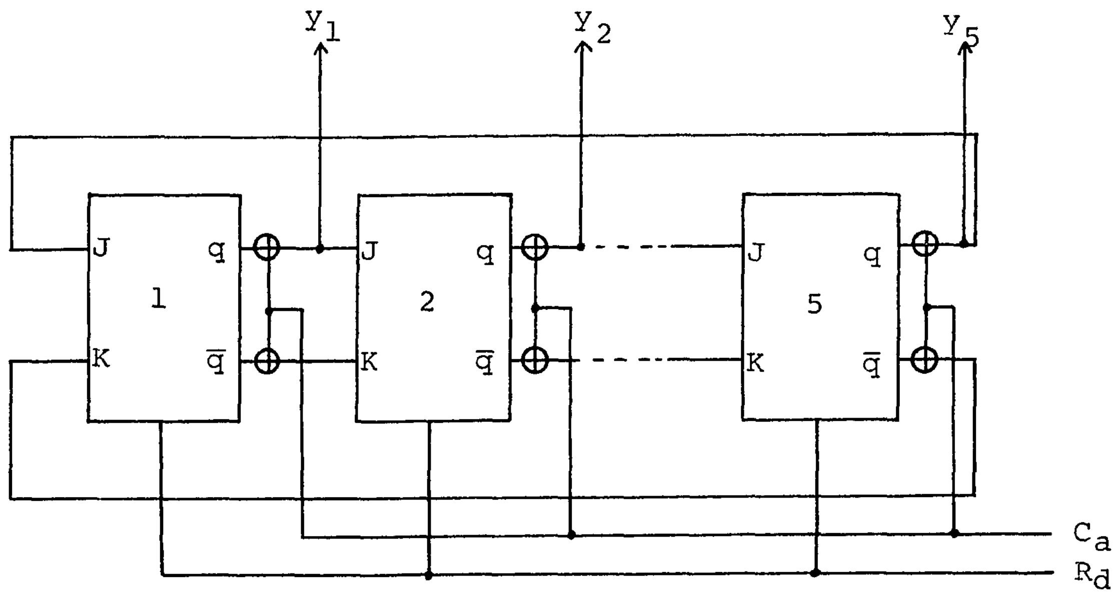


Figure 4.12
Block Diagram of Control Sequence Generator



(a)



(b)

Figure 4.13
A Five Stage Ring Counter and Its Modified Structure

operation of the modified counter can be verified by first applying the Rd=1 pulse to reset all the JKCMM to 0 state, then, performing the following four tests:

Test 1 Set Ca=0. It will provide the tests JiKi=01 for all $1 \leq i \leq 5$ simultaneously. After the system clock pulse is applied, the type A malfunction can be detected by observing the output pattern Y. If the observed output pattern Y $\neq$ 00000, then there exists the type A malfunction in the JKCMM which corresponds to the error output bit. Otherwise, the modified counter passes this test and forwards to next test.

Test 2 Set Ca=1. It will provide the tests JiKi=10 for all $1 \leq i \leq 5$ simultaneously. After the system clock pulse is applied, the type D malfunction can be detected by observing the output pattern Y. If the observed output pattern Y $\neq$ 00000, then there exists the type D malfunction in the JKCMM which corresponds to the error output bit. Otherwise, the modified counter passes this test and forwards to next test.

Test 3 Set Ca=0. It will provide the test JiKi=10 for all $1 \leq i \leq 5$ simultaneously. After the system clock pulse is applied, the type B malfunction can be detected by observing the output pattern Y. If the observed output pattern Y $\neq$ 11111, then there exists the type B malfunction in the JKCMM which corresponds to the error output bit. Otherwise, the modified counter passes this test and forwards to next test.

Test 4 Set Ca=1. It will provide the tests JiKi=01 for all $1 \leq i \leq 5$ simultaneously. After the system clock pulse is applied, the type C malfunction can be detected by observing the output pattern Y. If the observed output pattern Y $\neq$ 11111, then there exists the type C malfunction in the JKCMM which corresponds to the error output bit. Otherwise, the modified counter passes this test.

Once the modified counter passes the required four tests for detecting the malfunctions, it is ensured to perform its normal function properly when Ca is set to 0.

The combinational gating network of the CSG is mentioned in section 4.2.2. Its implementation is shown in Fig. 4.4 in which a set of AND gates is used to gate the output of the operation decoder and the state output of the ring counter. If we substitute each gate in the combinational network with proper PLM's according to the design algorithm and set the control signal of PLM1 as C_2. Then it is found that the input pattern OD and Y of the combinational gating network will provide the most sensitive input pattern for each gate in the modified gating network during each test mode of the fault detection on the modified IR(n) and the modified counter. Hence the modified gating network will sense the fault information carried on the OD and Y and propogate these fault information to the output (OC) of the modified gating network. Consequently, we conclude that the stuck-at-fault and malfunction of the modified IR(n) and the modified CSG can be

detected simultaneously provided that the status of the data bus and the output (OC) of the modified CSG are directly observale.

4.3.5 Memory Block

The structure of a k words of n-bit memory unit M(k,n) is shown in Figure 4.14(a). It consists of a block of memory elements, a memory address decoder and an output gating network. The block of memory elements contains k memory registers, each consists of n JK flip-flops. The memory address decoder is used to decode the contents of memory address register. The M(k,n) is a synchronous sequential network. Assume that the data input pattern D ($D=d_1\bar{d}_1 \ldots d_n\bar{d}_n$) and the address input pattern A($A=a_1,a_2 \ldots a_m$) and the control command Mw (for writing the data into the memory register) are directly accessible, and the memory output pattern Q ($Q= Q_1 Q_2 \ldots Qn$) is directly observable. Then the systematic design algorithm can be applied to design a real time detectable M(k,n). The resulting modified M(k,n) is shown in Figure 4.14(b) in which the generated test input pattern $D= C_1C_2 \ldots C_1C_2$, $A=C_2C_2 \ldots C_2$, and $Mw=C_2$ the desired output pattern $Q=C_1 C_1 \ldots C_1$, and the proper control signals are indicated. After the Rd=1 pulse is applied to reset all the JK flip-flops to 0 state, the stuck-at-faults and malfunctions of the modified M(k,n) can be detected by the test sequence described below:

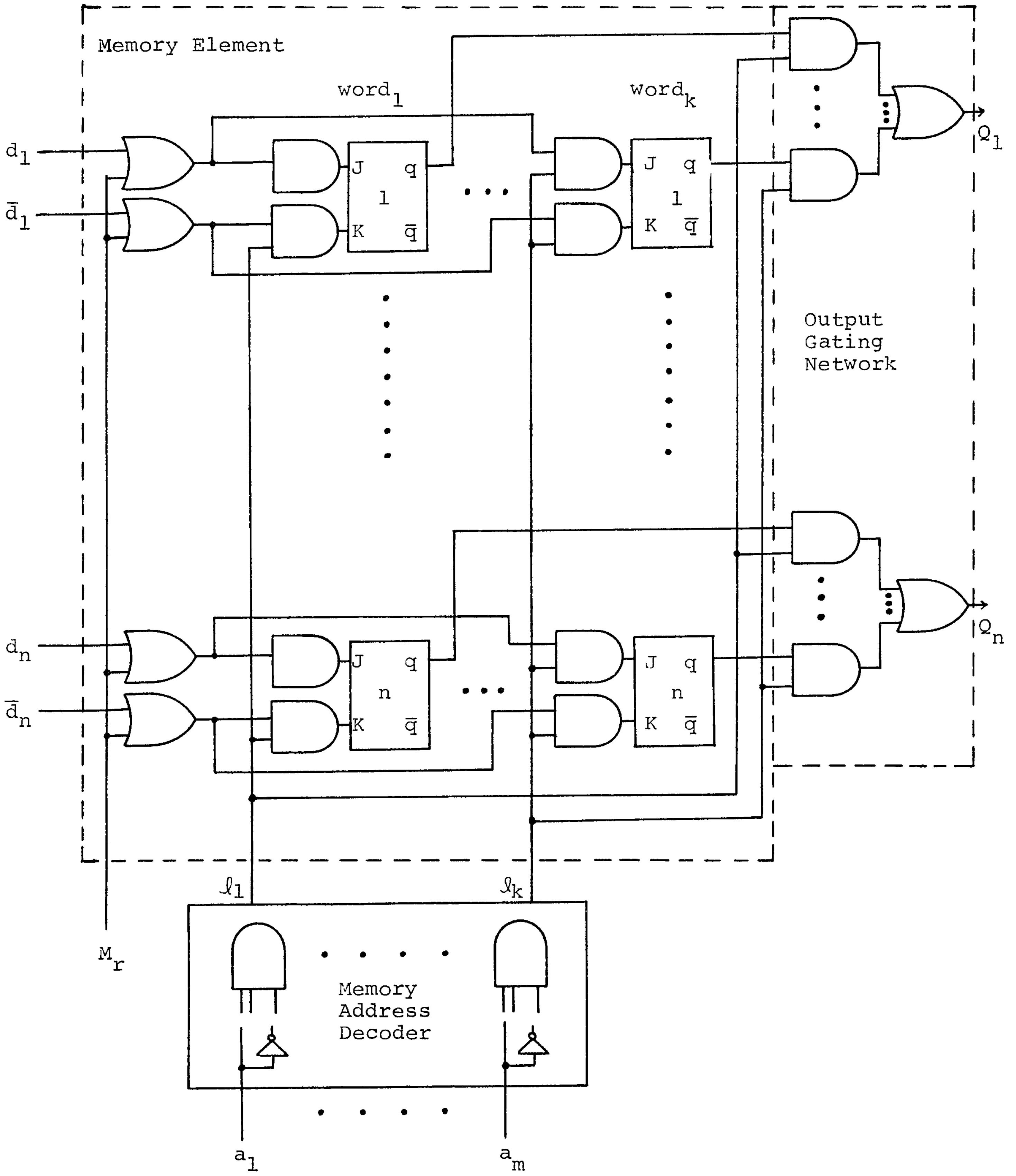


Figure 4.14 (a)

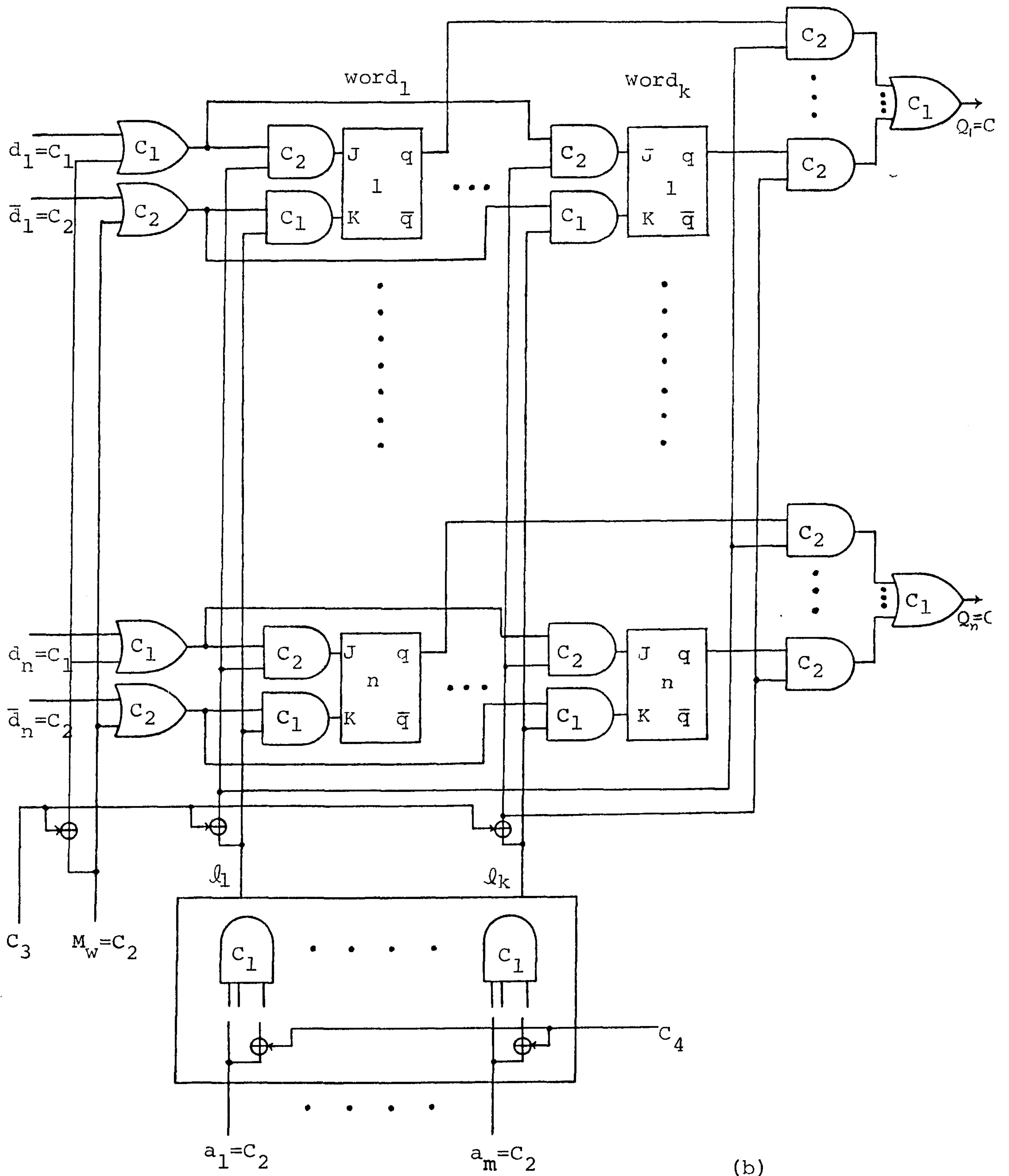


Figure 4.14

A Structure of k Words of n-Bit Memory Unit and Its Modified Version

Test 1 Set $C_1C_2C_3C_4$=0110 and apply the test input pattern D=01...01, A=11...1, and Mw=1. This will provide the tests JK=01 for all the JK flip-flop (JKFFs) and the most sensitive input pattern (MSIP) for each gate in this test mode simultaneously. After the system clock pulse is applied, the stuck-at-faults in Ha and type A malfunctions in JKFFs of the modified M(k,n) can be detected by observing the output pattern Q. If the observed pattern Q $\neq$00...0, then there exists at least one stuck-at-fault in Ha and/or type A malfunction in JKFF along the signal path which corresponds to the error output bit(s). Otherwise, the modified M(k,n) passes this test and proceeds to next test.

Test 2 Set $C_1C_2C_3C_4$=1010 and apply the test input pattern D=10...10,A=00...0 and Mw= 0. This will provide the tests JK=10 for all JKFFs and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, the stuck-at-faults in Hb and type D malfunctions in JKFFs of the modified M(k,n) can be detected by observing the output pattern Q. If the observed pattern Q $\neq$11...1, then there exists at least one stuck-at-fault in Hb and/or type D malfunction in JKFF along the signal path which corresponds to the error output bit(s). Otherwise, the modified M(k,n) passes this test and proceeds to next test.

Test 3 Set $C_1C_2C_3C_4$=1010 and apply the test input pattern D=10...10, A=00...0 and Mw=0. This will provide the tests JK=10

for all JKFFs and the MSIP for each gate in the test mode simultaneously. After the system clock pulse is applied, the stuck-at-fault in Hb and type B malfunction in JKFFs of the modified M(k,n) can be detected by observing the output pattern Q. If the observed pattern Q $\neq$11...1, then there exists at least one stuck-at-fault Hb and/or type B malfunction in JKFF along the signal path which corresponds to the error output bit(s). Otherwise, the modified M(k,n) passes this test and proceeds to next test.

Test 4 Set $C_1C_2C_3C_4$=0110 and apply the test input pattern D=01...01, A=11...1 and Mw=1. This will provide the tests JK=01 for all JKFFs and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, the stuck-at-faults in Ha and type C malfunctions in JKFFs of the modified M(k,n) can be detected by observing the output pattern Q. If the observed pattern Q$\neq$00...0, then there exists at least one stuck-at-fault in Ha and/or type C malfunction in JKFF along the signal path which corresponds to the error output bit(s). Otherwise, the modified M(k,n) passes this test.

If the modified M(k,n) passes the required four tests, it is guaranteed to perform its normal function properly when

the control siganls $C_1C_2C_3C_4$ is set to 0001.

4.3.6 Memory Address Register

Figure 4.15(a) shows a typical configuration of a memory address register MAR(m). Its input is connected to the address portion of the data bus (the least significant m bits). Its output is connected to the input of memory address decoder in M(k,n). The MAR(m) is used to hold the address of a memory register whose contents is to be fetched. The structure of MAR(m) indicates that the test input pattern for detecting the MAR(m) has to be provided externally. This problem can be avoided by connecting the output of MAR(m) to the data bus through tristate gates. Assume that the status of the data bus is directly observable and the control command MAR_t and MAR_r are directly accessible. Then the MAR(m) can be converted to a real time fault detectable network by employing the proposed design techniques. The resulting modified MAR(m) is shown in Figure 4.15(b) in which the proper control signals and the generated test input $MAR_r=C_2$ are indicated. The stuck-

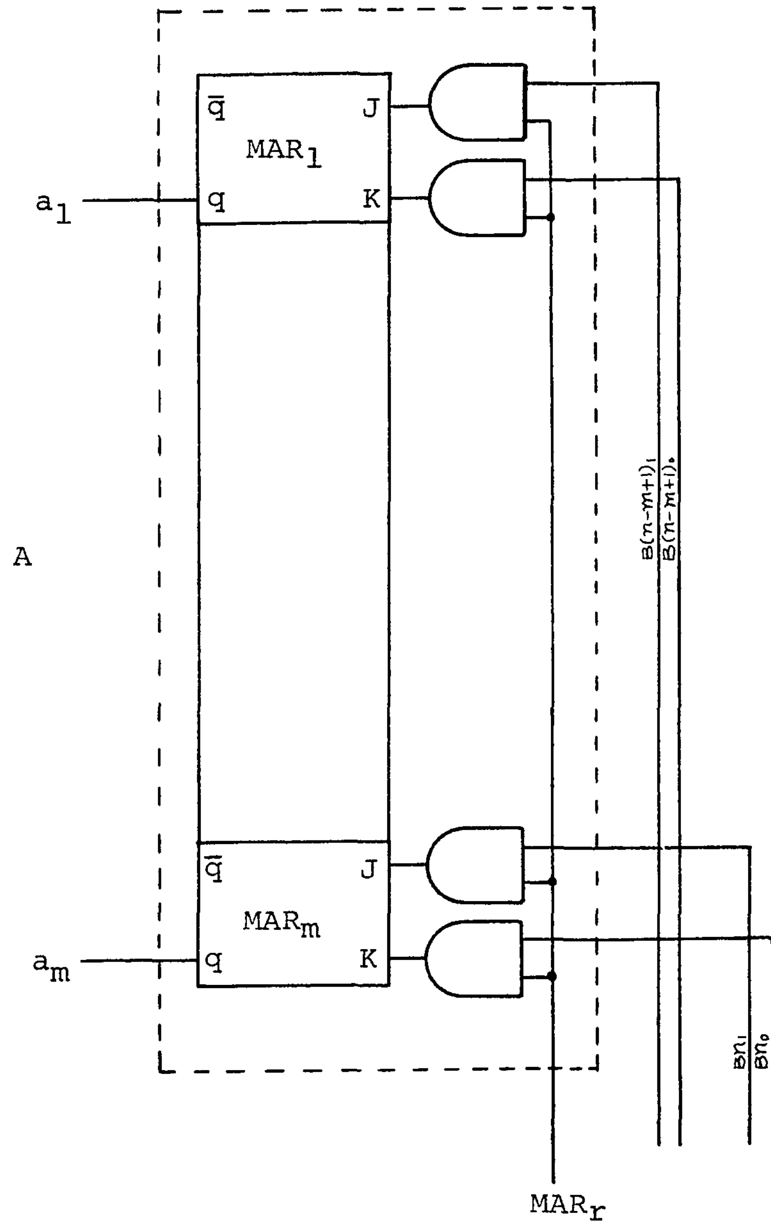


Figure 4.15 (a)

(b)

Figure 4.15
A Typical Configuration of Memory Address Register and Its Modified Version

at-faults and malfunctions of the modified MAR(m) can be detected by first, applying the Rd=1 pulse to reset each JKCMM to 0 state, then, carrying out the following test sequence:

<u>Test 1</u> Set $C_1C_2C_3C_a$= 0110 and apply MAR_t MAR_r =11. This will provide the tests JiKi=01 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, observing the status of the data bus. If the observed status Bi_1 $Bi_0 \neq 01$ for some i, then the presence of at least one stuck-at-fault in Ha and/or type A malfunction in JKCMM along the signal path which corresponds to the error ouput bit(s) is indicated. Otherwise, the modified MAR(m) passes this test and goes to next test.

<u>Test 2</u> Set $C_1C_2C_3C_a$=1011 and apply MAR_t MAR_r=10. This will provide the tests JiKi=10 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, observing the status of the data bus. If the observed status Bi_1 $Bi_0 \neq$ 01 for some i, then the presence of at least one stuck-at-fault in Hb and/or type D malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified MAR(m) passes this test and goes to next test.

<u>Test 3</u> Set $C_1C_2C_3C_a$=1010 and apply MAR_t MAR_r =10. This will provide the tests JiKi=10 for all i and the MSIP for each

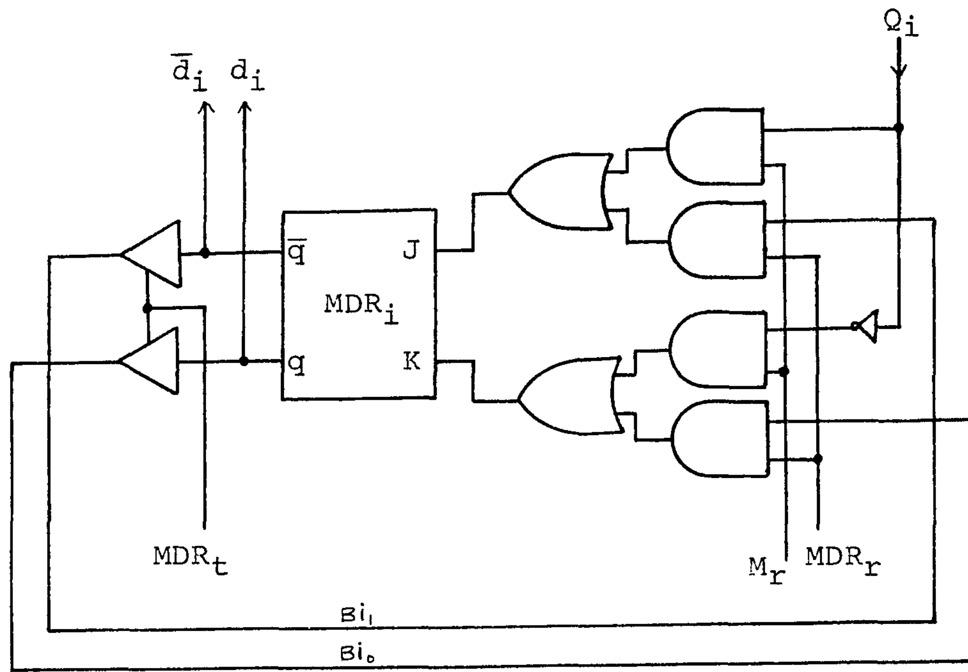
gate in this test mode simultaneously. After the system clock pulse is applied, observing the status of the data bus. If the observed status $Bi_1\ Bi_0 \neq 10$ for some i, then the presence of at least one stuck-at-fault in Hb and/or type B malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified MAR(m) passes this test and goes to the next test.

Test 4 Set $C_1C_2C_3C_a$=0111 and apply $MAR_t\ MAR_r$=11. This will provide the tests JiKi=10 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, observing the status of the data bus. If the observed status $Bi_1\ Bi_0 \neq 10$ for some i, then the presence of at least one stuck-at-fault in Ha and/or type C malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified MAR(m) passes this last test and complete the test sequence.

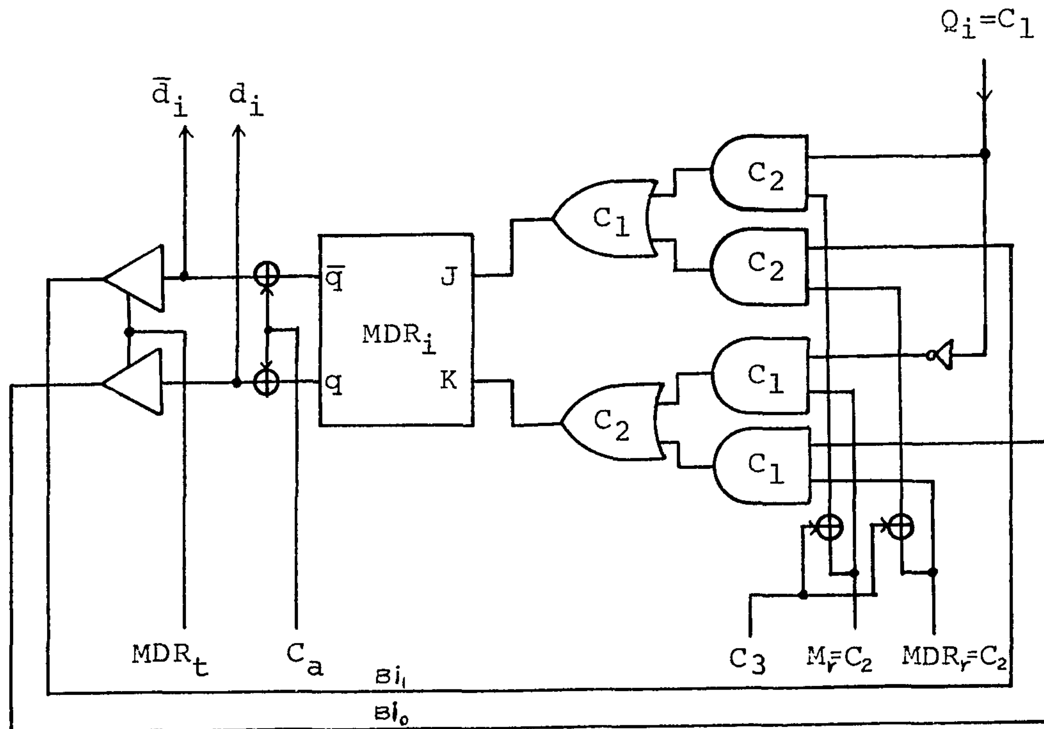
Once the modified MAR(m) passes the required four tests, it is assured to operate its normal function correctly if the control signals $C_1C_2C_3C_a$ is set to 0000.

4.3.7 Memory Data Register

A typcial stage of a memory data register MDR(n) is shown in Fig. 4.16(a). The input of MDR(n) comes from two sources: (1) the data bus, (2) the output Q of memory block. The output of MDR(n) goes into two areas: (1) the data bus,



(a)



(b)

Figure 4.16

A Typical Stage of Memory Data Register
and Its Modified Structure

(2) the data input D of the memory block. The control command M_r is used to strobe the data from memory register in the memory block to the MDR(n), and MDR_r is used to strobe the data from the data bus to the MDR(n). Assume that the status of the data bus is directly observable and the input pattern Q and control commands MDR_t MDR_r M_r are directly accessible. There the MDR(n) can be converted to a real time detectable network by applying the design algorithm. The resulting modified MDR(n) is shown in Fig. 4.16(b) in which the proper control siganls and the generated test input pattern $Q=C_1C_1 \ldots C_1$ and MDR_r M_r = $C_2C_2$are presented. The stuck-at-faults and malfunctions of the modified MDR(n) can be detected by first, applying the Rd=1 pulse to reset all the JKCMM to 0 state, then performing the four tests described below:

Test 1 Let $C_1C_2C_3C_a$=0110 and apply MDR_t MDR_r M_r =111 and test input pattern Q=00...0. This will provide the tests JiKi=01 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, observing the status of the data bus. If the observed status Bi_1 $Bi_0 \neq 01$ for some i, then the presence of at least one stuck-at-fault in Ha and/or type A malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified MDR(n) passes this test and proceeds to next test.

Test 2 Set $C_1C_2C_3C_a$=1011 and apply MDR_t MDR_r M_r=100 and test input pattern Q=11...1. This will provide the tests JiKi=10 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, observing the status of the data bus. If the observed status Bi_1 Bi_0≠01 for some i, then the presence of at least one stuck-at-fault in Hb and/or type D malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified MDR(n) passes this test and proceeds to next test.

Test 3 Set $C_1C_2C_3C_a$=1010 and apply MDR_t MDR_r M_r=100 and test input pattern Q=11...1. This will provide the tests JiKi=10 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, observing the status of the data bus. If the observed status Bi_1 Bi_0≠10 for some i, then the presence of at least one stuck-at-fault in Hb and/or type B malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified MDR(n) passes this test and proceeds to next test.

Test 4 Set $C_1C_2C_3C_a$=0111 and apply MDR_t MDR_r M_r=111 and test input pattern Q=00...0. This will provide the tests JiKi=01 for all i and the MSIP for each gate in this test mode simultaneously. After the system clock pulse is applied, observing the status of the data bus. If the

observed status $Bi_1\ Bi_0 \neq 10$ for some i, then the presence of at least one stuck-at-fault in Ha and/or type C malfunction in JKCMM along the signal path which corresponds to the error output bit(s) is indicated. Otherwise, the modified MDR(n) passes this test.

Once the modified MDR(n) passes the required four tests. The proper function of the network is verified. Therefore, it is guaranteed to operate its normal function properly when $C_1C_2C_3C_a=0000$.

From the design of modified M(k,n) and modified MDR(n), it is noted that the generated desired output pattern Q of the modified M(k,n) equals to the generated test input pattern Q of the modified MDR(n), and the generated desired output pattern of the modified MDR(n) equals to the generated test input pattern D of the modified M(k,n). Therefore, we can combine these two functional blocks and perform the required test sequence simultaneously. The resulting network configuration is shown in Fig. 4.17. Assume that the status of the data bus is directly observable and the input pattern A and control commands MDR_t, MDR_r, Mw, Mr are directly accessible. Then the new network can be detected by the same test sequence by the systematic fault detection procuedure described in section 4.3.5 and 4.3.7 and the additional test which is identical to test 1 for detecting the stuck-at-fault in Ha and the type C malfunctions in JKCMM of the modified memory block.

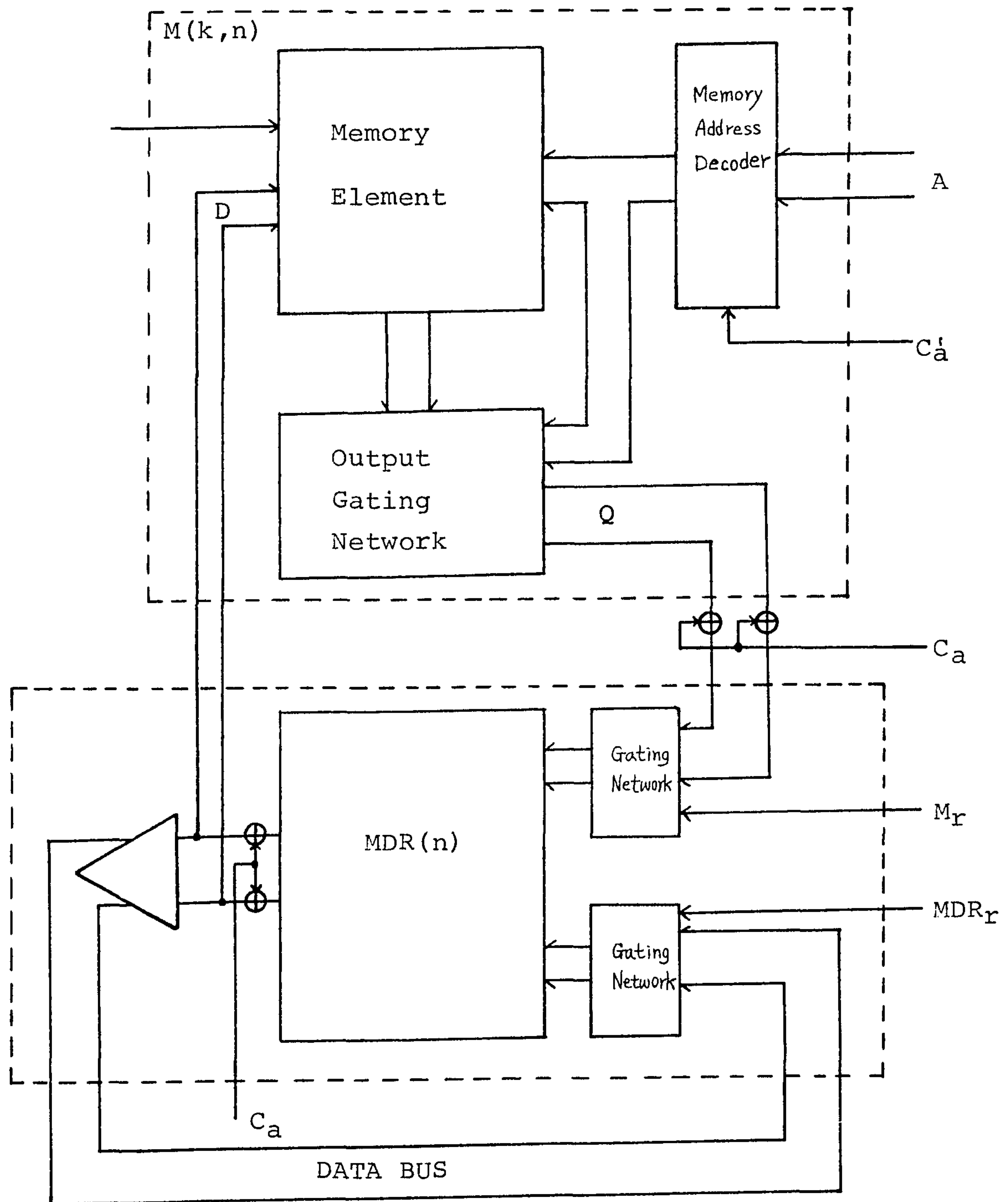


Figure 4.17
A Combined Structure of Modified Memory Unit and Modified Memory Data Register

Notice that the test input pattern A can be provided by the modified MAR(m) if all its JKCMM is reset to 0 state simultaneously with the new network and its control signals C_a' is set as C_a'=1,0,0 and 1 during test 1, 2, 3 and 4 respectively.

In summary, we note that each functional block can be implemented by the design algorithms developed earlier which, in turn, will enable the block to perform real time fault detection according to the systematic fault detection procedure. Table 4-2 tabulates the resulting test input pattern, the control signal and desired output patterns for each test of the block.

4.4 System Fault Detection

Based on the fault detection tests derived in the preceding section for each functional block in the system, one can utilize a hard core ROM (read only memory) to store the control signal pattern, test input pattern and the desired output pattern for each test presented in Table 4.2 in sequential order, and organize a system as shown in Fig. 4.18. In this system, the outputs of the control sequence generator are disconnected from the control bus by the control signal Cp=1 during the system testing. On the other hand, the outputs of the ROM are connected to the control bus by the control Cp=1 during the testing of the system and disconnected from the control bus by the control signal Cp=0 during the normal operation of the system. All the test input pattern and

Functional Block	Test No.	Control Signal Pattern	Test Input Pattern	Desired Output Pattern
AR(n)	1	$C_1C_2C_3C_4Ca$=01100	$z_1c_1, p_1p_2p_3p_4, p_5p_6p_7p_8E_1$=10, 0000, 00000 ARtARr =11	Bi_1Bi_0 =01 ∀ i $z_{n+1}c_{n+1}$ =10
	2	10101	01, 1111, 11111 10	01 ∀ i 10 (when C_1C_2=01)
	3	10100	01, 1111, 11111 10	10 ∀ i 01
	4	01101	10, 0000, 00000 11	10 ∀ i 01 (when C_1C_2=10)
PC(m)	1	$C_1C_2C_3Ca$=0110	PCtPCrPCinc=110	Bi_1Bi_0 =01 ∀ i
	2	1011	101	01 ∀ i
	3	1010	101	10 ∀ i
	4	0111	110	10 ∀ i
IR(n) and CSG	1	$C_1C_2C_3Ca$=0110	IRtIRr=11	Bi_1Bi_0 =01 ∀ i OC =00...0
	2	1011	10	01 ∀ i 11...1 (when C_1C_2=01)
	3	1010	10	10 ∀ i 11...1
	4	0111	11	10 ∀ i 00...1 (when C_1C_2=10)
MAR(m)	1	$C_1C_2C_3Ca$=0110	MARtMARr=11	Bi_1Bi_0 =01 ∀ i
	2	1011	10	01 ∀ i
	3	1010	10	10 ∀ i
	4	0111	11	10 ∀ i
M(k, n) and MDR(n)	1	$C_1C_2C_3C_4CaCa'$=011001	MDRtMDRrMwMr=1111	Bi_1Bi_0=01 ∀ i
	2	101010	1000	01 ∀ i
	3	101000	1000	10 ∀ i
	4	011011	1111	10 ∀ i
	5	01100-	1111	01 ∀ i

Table 4.2

The Test Patterns of Each Functional Block

control pattern to each functional block will be supplied by the test stored in the ROM. The tested output pattern is then compared with the desired output pattern for the corresponding test stored in the ROM through a hard core comparator as shown in the Figure. The fault information will be indicated by the outputs of the comparators.

The operation of the system can be designed such that the initiation of a new task is to activate a control command to set the hard core Cp flip-flop to enable the output of ROM to the control bus and the comparator. Therefore, each test stored in the ROM is performed under the control of system clock. If there is any fault indication from the comparator for the corresponding test, then this fault indication will be used to call the repair service or to activate a switching network to switch a good spare block to replace the corresponding faulty module. Otherwise, the testing process continues until all the tests stored in the ROM are exercised. At the completion of the last test, a control command from ROM can reset the Cp flip-flop to disconnect the output of ROM and to connect the output of control sequence generator to the control bus, to resume the normal operation of the system.

Once the system passes all the tests stored in the ROM, then the proper function of each block is validated as described in the section 4.3. The system is guaranteed to operate its normal operation correctly when the control

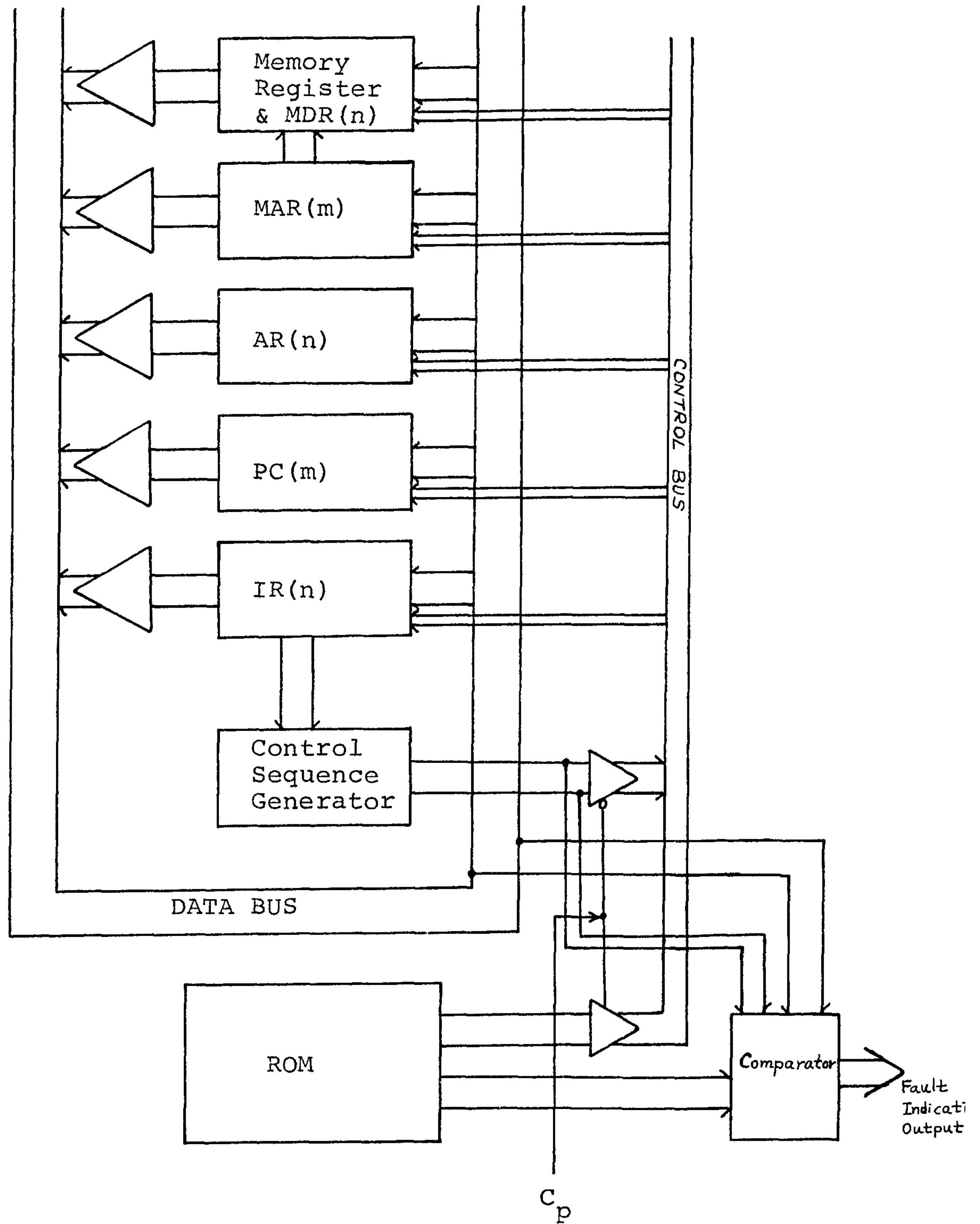


Figure 4.18

An Architectual Configuration of a Real Time Self-Testing Digital System

signal $C_1C_2C_3C_4C_a$ is set to 00010.

The system implemented in such a manner can be detected within 21 clock periods which is less than five machine cycles (each machine cycle equals to five clock periods as mentioned in section 4.2) since we combine the testing of the instruction register and the control sequence generator, and that of the memory register and the memory data register. This proposed system provide the software engineer the feature to check the correct operation of the system at the desired frequency for a specified real time application. It should be mentioned that all the data in each register will be altered after the testing of the system. Therefore the software engineer is responsible for storing all the necessary datas before the activation of the test operation.

The upperbound of fault detection tests of a system will be, in general, 4M tests from the observation of the results derived in section 4.3, where M is the number of functional blocks contained in a system. 4M tests will require no more than M machine cycles if the machine cycle of a system equals to 4 or more clock periods. Therefore, we conclude that the fault detection process of a system implemented according to the described manner requires, in general, at most 4M clock periods which is usually no more than M machine cycles of the system, and the testing is self-contained.

4.5 Summary

It is demonstrated that the proposed systematic design algorithms enable us to design a real time fault detectable digital system. The test is generated automatically at the completion of the design. Hence, there is no need to generate the test sequence by a complex algorithm requiring a large amount of computer time and memory as developed by the conventional techniques. The test sets derived can be stored in a hard core ROM. The fault detection capability of the proposed system is self-contained by the utilization of a hard core ROM and a hard core comparator.

By comparison, the conventional techniques require a hard core computer to perform the generated test sequence and the testing is not self-contained. The proposed techniques, on the other hand, requires only a set of hard core control signals, a hard core ROM to store the generated tests and a hard core comparator to compare the tested output and the desired output to indicate the fault information.

The major difference between the conventional techniques and the proposed technique is the length of the test sequence for detecting the fault of a system. The former requires an extremely long test sequence and the length of the test sequence is proportional to the complexity of the system, which makes it impossible to use in the real time application environments. The proposed technique requires,

at most, 4M tests. The upperbound is proportional to the number of blocks (M) contained in the system, but is independent of the complexity of each individual block. The test sequence will require, in general, no more than M cycles of a system, which makes the technique applicable in real time.

It should be emphasized that the advantages are obtained through the systematic use of redundnat hardware in the logic modules and the controlled signal invertors. The cost of the real time fault detectable system will be higher than the ordinary system. In applications demanding the assurance of the system operation in real time, however, the proposed technique provides a systematic method to design a system which fulfills this requirement at the expense of redundant hardware.

CHAPTER 5

ANALYSIS OF SYSTEM PERFORMANCE

In earlier chapters, it was shown that a real time fault detectable digital system can be designed by the proposed algorithm through the use of redundant hardware embedded in the logic modules. The utilization of redundant hardware will decrease the reliability of a system. In this chapter, the reliability of the ordinary system and real time fault detectable system are compared.

The <u>failure rate</u> of a device d, λd, is defined as the probability of failure of device d within T uint of time under certain environmental conditions. The λd can be determined by testing N identical devices for T units of time. If Nf out of N have failed during this time period, then $\lambda d=Nf/T$ and the unit of λd is failures per unit time.

The <u>reliability</u> of a device d, Rd, is defined as the probability that the device will perform normally for a specified time under certain environmental conditions. The Rd can be determined by testing N identical devices for certain amount of time. If Ns out of N still survive, then $Rd=Ns/N$ at that time.

In reality, the failure rate and reliability of a device are functions of time, therefore we denote them as $\lambda d(t)$

and Rd(t) respectively. The relationship between $\lambda d(t)$ and Rd(t) can be derived as follows:

Assume that we put N same device on test under certain environmental conditions, starting at time t=0. If at time t only Ns out of N devices still survive, then Nf=N-Ns must be failed. Hence the failure rate of each of the Nf failed device is dNf/dt. Consequently, each of the remaining Ns devices will fail at the rate $\lambda d(t)$, where

$$\lambda d(t) = \frac{1}{Ns} \frac{dNf}{dt} \qquad (1)$$

By definition, the reliability of the device at time t, Rd(t), equals Ns/N. Therefore,

$$Rd(t) = 1 - \frac{Nf}{N} \qquad (2)$$

Taking the derivative on Rd(t):

$$\frac{dRd(t)}{dt} = -\frac{1}{N} \frac{dNf}{dt} \qquad (3)$$

From equation (1) and (3), we conclude,

$$\lambda d(t) = -\frac{1}{Rd(t)} \frac{dRd(t)}{dt} \qquad (4)$$

or,

$$Rd(t) = \exp(-\int_0^t \lambda d(t)\,dt) \qquad (5)$$

Equations (4) and (5) define the relationship between $\lambda d(t)$ and Rd(t). Physically, $\lambda d(t)$ is a positive number at any time. Thus equation (5) implies that the value of Rd(t) is monotonically decreasing with time.

Let Fd(t) be the failure distribution of device d.

Then,

$$Fd(t) + Rd(t) = 1$$

due to the fact that the success and failure of a device are mutually exclusive. Therefore the failure density of device d, fd(t), is

$$fd(t) = \frac{dFd(t)}{dt} = 1 - \frac{dRd(t)}{dt} \qquad (6)$$

Then the mean time to failure of devide d, MTTFd, can be determined mathematically by finding the mean of fd(t), i.e.,

$$MTTFd = \int_0^\infty t \cdot fd(t)dt \qquad (7)$$

From equation (6) and (7), we find

$$MTTFd = \int_0^\infty Rd(t)\ dt \qquad (8)$$

In the following discussion, the equation (5) and (8) are applied to compare the reliability and mean time to failure between the ordinary gates and the proposed modules. Assume that the failure rate of each gate, λg, in the proposed PLM's is the same. In the worst case, each gate in the module will fail independently. Then, the failure rate of each PLM, λm, will be four times of λg (i.e., $\lambda m=4\lambda g$). We also let the MTTFg and MTTFm denote the mean time to failure of a gate, and that of a module respectively. At the current state of semiconductor technology, the failure rate of integrated circuit (IC) is greatly reduced to the range of $\lambda=10^{-7} \sim 10^{-8}$ failures/hour [4].

Three models of failure rate are considered to compare the reliability and mean time to failure of a gate and a module as follows:

(1) Constant Model In this model, the failure rate

λg is independent of time and equals to a constant value, i.e., $\lambda g=\lambda$. Therefore, from equation (5) and (8), we obtain

$$Rg_1(t) = \exp(-\lambda t), \quad MTTFg_1 = 1/\lambda$$

$$Rm_1(t) = \exp(-4\lambda t), \quad MTTFm_1=1/4\lambda$$

(2) Linearly Increasing Model In this model, the failure rate λg is linearly increasing with time, i.e. $\lambda g=\lambda t$. Substitute this into equations (5) and (8), we obtain

$$Rg_2(t)=\exp(-\lambda t^2/2), \quad MTTFg_2=\sqrt{\pi/(2\lambda)}$$

$$Rm_2(t)=\exp(-2\lambda t^2), \quad MTTFm_2=\frac{1}{2}\sqrt{\pi/(2\lambda)}$$

(3) Nonlinearly Increasing Model In this model, the failure rate λg is nonlinearly increasing with time. There are various models in this class [45]. We choose the Weibull Modle for m=2, i.e., $\lambda_g=\lambda t^2$ as a sample example of this class. Then, from equation (5) and (8) we derive

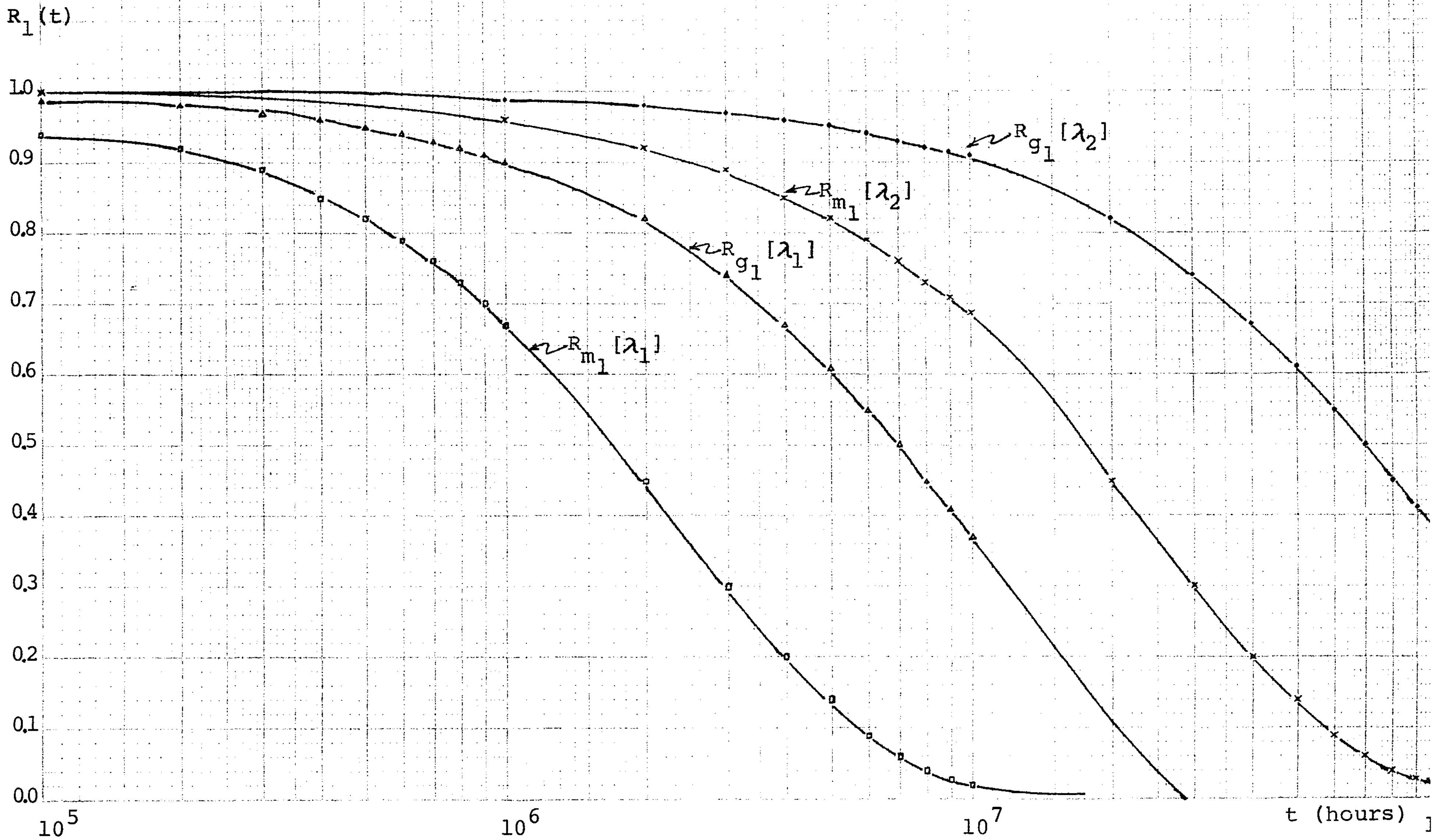
$$Rg_3(t)=\exp(-\lambda t^{3}/3), \quad MTFFg_3= \Gamma(4/3)/[\lambda/3]^{1/3}$$

$$Rm_3(t)=\exp(-4\lambda t^3/_3), \quad MTFFm_3= \Gamma(4/3)/[4\lambda/_3]^{1}/_3$$

The values of $Rg_i(t)$ and $Rm_i(t)$ for i=1,2,3 are calculated (based on $\lambda_1=10^{-7}$ and $\lambda_2=10^{-8}$ failures/hour) and shown in Figure 5.1, 5.2 and 5.3 respectively. The MTTFgi and MTTFmi for i=1,2,3 are also computed as shown in Table 5.1.

Figure 5.1 indicates that the $Rm_1(t)$ is degraded by less than 10% for 2.10^5 and 2.10^6 hours (respectively for λ_1 and λ_2) of usage, but is rapidly degraded thereafter. Figure 5.2 shows that the reliability of module in model two $Rm_2(t)$ is

Figure 5.1
Reliability Comparison between Gate and Module
with Constant Failure Rate Model



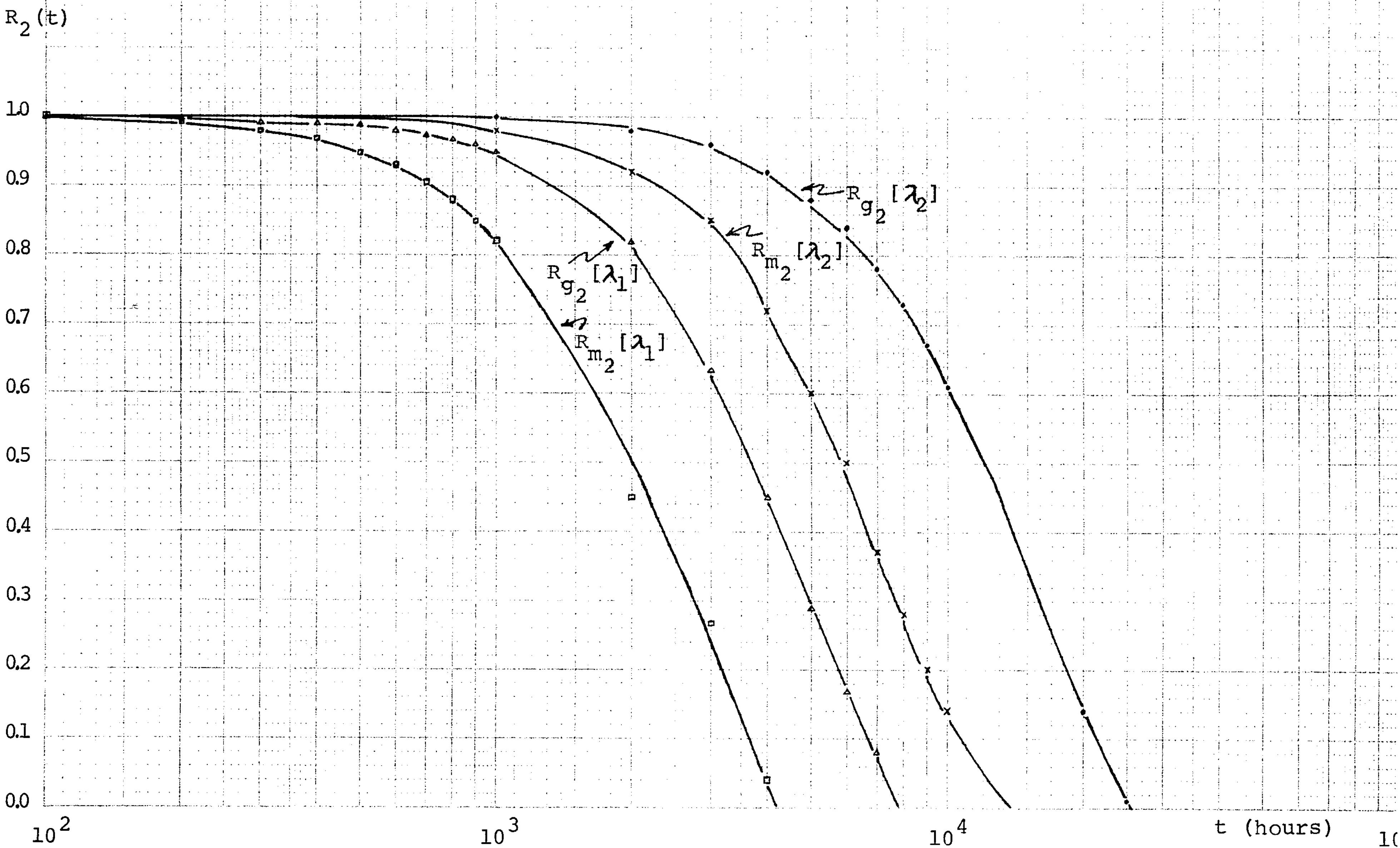
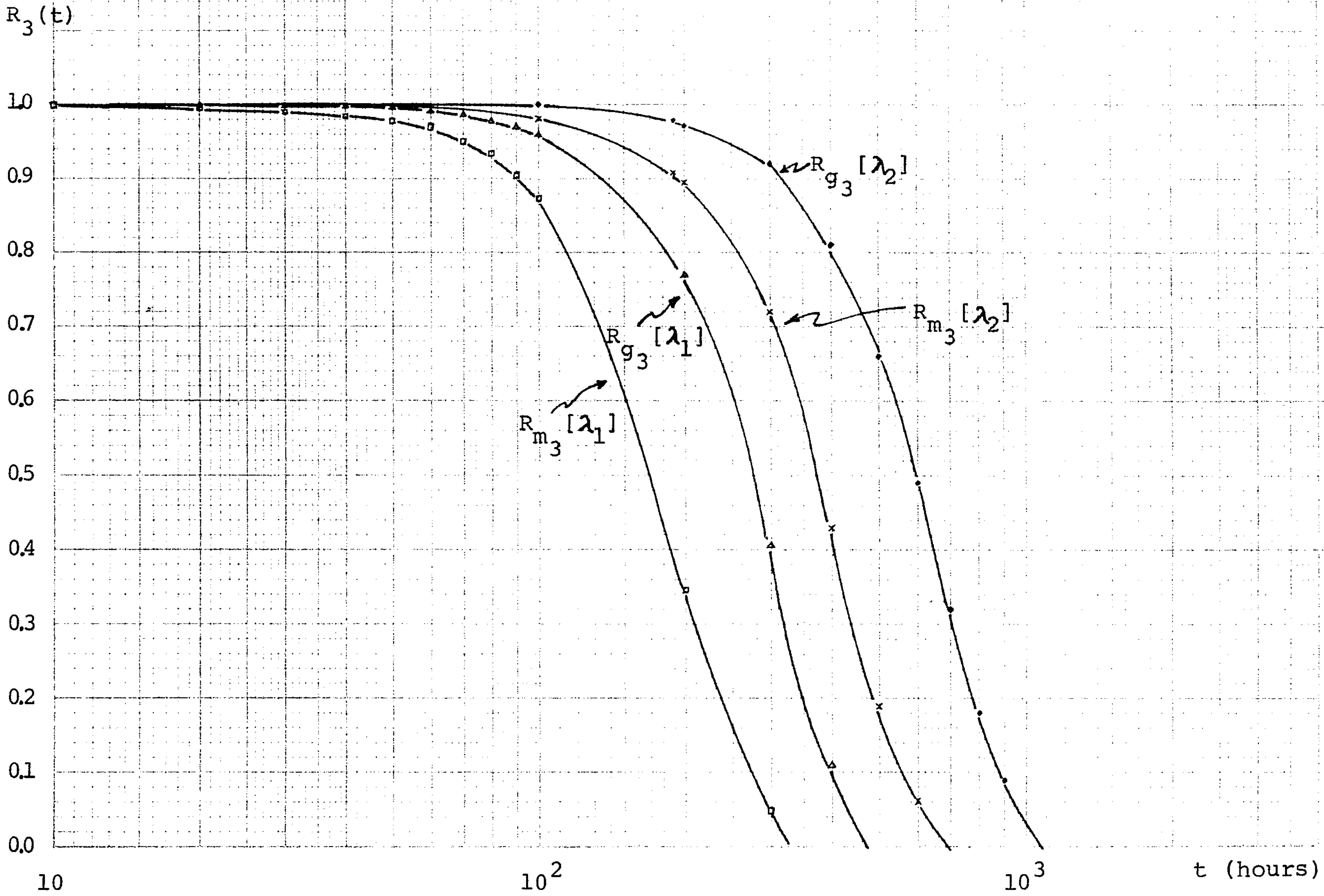


Figure 5.2

Reliability Comparison between Gate and Module with Linearly Increasing Failure Rate Model

Figure 5.3

Reliability Comparison between Gate and Module with Non-Linearly Increasing Failure Rate Model



degraded by less than 10% for 7.10^2 and 2.10^3 hours (respectively for λ_1 and λ_2) of usage but is greatly degraded thereafter. Figure 5.3 shows that the reliability of module in model three $Rm_3(t)$ is only degraded by less than 10% for 90 and 190 hours (respectively for λ_1 and λ_2) of usage, but is greatly degraded thereafter. Table 5.1 shows that the MTTF of the module is reduced to 25%, 50% and 63% for the failure rate model 1, 2 and 3 respectively.

In combinational network design, the modified dual network (MDN) uses PLM's instead of ordinary gate. Therefore, the reliability of the modified network will be degraded accordingly. If a MDN contains CSI's, there will be additonal decrease in the network reliability. But, in general, the number of CSI used in the MDN is far less than that of PLM's used, hence the effect introduced by the CSI can be neglected.

In the controllable memory module (CMM), the redundant components are the two additional exclusive-or gates. Comparing with the master-slave flip-flop, the redundancy introduced by the CMM is increased only by two-fold since the exclusive-or gate can be implemented by no more than four gates and the master-slave flip-flop can be implemented by eight gates. Consequently, we note that the reliability of the modified synchronous sequential network (MSSN) will be degraded by the amount no more than that described above, since the failure rate of PLM's is increased by four times

	Model 1		Model 2		Model 3	
	$\lambda_1=10^{-7}$	$\lambda_2=10^{-8}$	λ_1	λ_2	λ_1	λ_2
$MTTF_g$	10^7 hr	10^8 hr	3963 hr	12533 hr	278 hr	598 hr
$MTTF_m$	$2.5\ 10^6$ hr	$2.5\ 10^7$ hr	1982 hr	6267 hr	175 hr	377 hr
$\frac{MTTF_g}{MTTF_m}$	25%	25%	50%	50%	63%	63%

Table 5.1
Mean Time to Failure of Gate and Module in Three Models

than that of the ordinary gates and the failure rate of CMM's is increased by two times than that of the ordinary flip-flops, therefore the failure rate of the MSSN will be increased by no more than four times of that of the ordinary system.

A system with high reliability implies that it will have high probability of good performance under the specified operating ambiance. In a complex system, however, component failures are impossible to avoid, and the presence of a failure will sometimes cause serious effects on the overall system performance regardless of the system reliability. In some applications, therefore, it is required that a system must be frequently tested for its normal function in order to insure that no fault is present in the system. This performance criteria will be referred to as the accuracy.

The accuracy of a system is defined as the amount of time required to verify the functional correctness of a system and to indicate the fault information if one or more faults is present in the system. The shorter the amount of time needed, the higher the accuracy of a system.

The conventional techniques generate the test sequence for a system from its original design. Thus the reliability will not be degraded. But the length of the generated test sequence is, in general, too long to test the normal operation of the system frequently. Therefore, the

accuracy of the system is low.

On the other hand, the design algorithm, developed in Chapters 2 and 3 allow us to design a modified system such that the fault in a modified system containing M functional blocks can be detected by only 4M tests through the utilization of the logic moduels. The 4M tests will usually require no more than M machine cycle times as demonstrated in Chapter 4. Therefore, it is feasible to verify the proper operation of the modified system with adequate frequency to achieve high accuracy for the system. The reliability of the modified system is degraded as discussed earlier. This tradeoff can be solved by using lower failure rate component as illustrated below.

Consider a system implemented with ordinary components whose failure rate equals to λ_1 (or $\lambda_1 t$, $\lambda_1 t^2$) where $\lambda_1 = 10^{-7}$ failures/hour and the same system implemented with the logic modules whose failure rate equals to $4\lambda_2$ (or $4\lambda_2 t$, $4\lambda_2 t^2$) where $\lambda_2 = 10^{-8}$ failures/hour. Then Figures 5.1, 5.2 and 5.3 show that the reliability of the later system is higher than that of the former system. This demonstrates that the reliability of a system can be enhanced through the use of components with improved failure rate, namely, from $\lambda_1 = 10^{-7}$ to $\lambda_2 = 10^{-8}$. However, a highly accurate system can only be achieved through real time testing.

As a summary, a highly reliable and highly accurate system can be designed by the systematic design algorithms

developed in this study and the use of lower failure rate components. The cost of this system will be higher. However, in application environments requiring high accuracy in system performance as a major concern, the developed techniques provide a systematic approach to the design satisfying this criteria.

CHAPTER 6

CONCLUSIONS

Recent advances in intergrated circuit (IC) technology development have resulted in an increasing emphasis on fault detection, since the detection of functional failure is sufficient to identify the IC package for replacement. In some real time applications, the presence of a fault has to be detected promptly in order to insure the recovery of a system before the normal operations are affected. Therefore, a highly accurate, real time fault detectable system is required for this purpose.

The design algorithms developed in this dessertation provide a designer with systematic means for designing:

(1) A combinational network whose stuck-at-faults can be detected by two tests and the test input pattern is generated by the design algorithm developed in Chapter 2.

(2) A synchronous sequential network whose stuck-at-faults and malfunctions can be detected by four tests and the test pattern is generated by the design algorithm developed in Chapter 3.

(3) A real time, on-line, self-testing digital system whose stuck-at-faults and malfunctions can be detected by 4M tests (M is the number of functional

blocks contained in the system) by extending the developed design algorithms and with the utilization of a hard core read only memory and a hard core comparator.

The established systematic fault detection procedures for combinational network and synchronous sequential network demonstrate that the test procedure is easier to apply compared with other techniques [20], [49].

It is concluded that a highly accurate and reliable, real time fault detectable digital system can be implemented by the developed design technique and through the use of lower failure rate components.

The required redundant hardware in the proposed modules will increase the component cost, and the reliability of the module compared with functionally equivalent single gate is degraded. However, it was shown in Chapter 5 that a highly reliable module can be fabricated with lower failure rate components. The fabrication of lower failure rate components will, on the other hand, increase the cost. Consequently, an analysis of cost-benefit trade-off for the feature must be made for the overall system in specific application environments. The trend of fabrication technology allows us to expect that the component failure rate will continue to improve, which, in turn, will make the design algorithms developed in this dissertation more attractive. At present, the

design techniques are most suitable for applications which require high accuracy as the major design criterion rather than the cost.

It should be mentioned that the problem on the fault detection of an asynchronous sequential network is not treated in this dissertation. Recently, Chuang [11] proposed to design an asynchronous sequential network by utilizing the self-synchronization principle, in which the state variables are realized by flip-flops triggered by an internally generated clock signal. The clock is generated only when an internal state change is required. An extension of this concept and application of developed design techniques in this dissertation to a real time fault detectable asynchronous sequential network remain as a major problem of practical importance.

REFERENCES

1. Armstrong, D. B., "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets." IEEE Trans. on Electronic Computers, vol. EC-15, pp. 66-73, Feburary 1966.

2. Bearnson, L. W. and Carrol, C. C., "On the Design of Minimum Test Length Fault Tests for Combinational Circuits," IEEE Trans. on Computers, vol. C-20, pp. 1353-1356, November 1971.

3. Betancourt, R., "Derivation of Minimum Test Sets for Unate Logical Circuits," IEEE Trans. on Computers, vol. C-20, pp. 1264-1269, November 1971.

4. Borgerson, B. R. and Freitas, R. F., "An Analysis of PRIME Using a New Reliability Model," The Fourth Annual International Symposiun on Fault-Tolerant Computing, Section 2, pp. 26-31, 1974.

5. Bosson, D. C. and Hong, S. J., "Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks," IEEE Trans. on Computers, vol. C-20, pp. 1252-1257, November 1971.

6. Brule et.al., "Diagnosis of Equipment Failure," IRE Trans. on Reliability and Control, vol. RQC-9, pp. 23-34, 1960.

7. Chang, H. Y., Manning, E. G. and Metze, G., "Fault Diagnosis of Digital Systems," Wiley-Interscience, New York, 1970.

8. Chang, H. Y., "An Algorithm for Selecting an Optimum Set of Diagnostic Tests," IEEE Trans. on Electronic Computers, vol. EC-14, pp. 706-711, October 1965.

9. Chuang, C. S. and Oh, S. J., "Design of Easily Detectable Combinational and Synchronous Sequential Circuits," Eighth Asilomar Conference on Circuits, Systems and Computers, Conference Record pp. 68-72, December 1974.

10. Chuang, C. S. and Oh, S. J., "Testability Enhancement in Digital System Design," IEEE Intercon75, Record Paper No. 11-3, pp. 1-7, April 1975.

11. Chuang, H. Y. H., "Fail-Safe Asynchronous Machines with Multiple Input Changes," International Symposium on Fault-Tolerant Computing, pp. 124-129, 1975.

12. Das, P. and Farmer, D. E., "Fault Detection Experiments for Parallel Decomposable Sequential Machines," IEEE Trans. on Computers, vol. C-24, pp. 1104-1108, November 1975.

13. Dias, F. J. O., "Fault Masking in Combinational Logic Circuits," IEEE Trans. on Computers, vol. C-24, pp. 476-482, May 1975.

14. Friedman, A. D. and Menon, P. R., "Fault Detection in Digital Circuits," Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

15. Fujiwara, H., Nagao, Y., Sasao, T. and Kinoshita, K., "Easily Testable Sequential Machines with Extra Inputs," IEEE Trans. on Computers, vol. C-24, pp. 821-826, August 1975.

16. Galey, J. M., Norby, R. E. and Roth, J. P., "Techniques for Diagnosis for Switching Circuit Failures," Proc. 2nd Annual Symp. on Switching Theory and Logic Design, pp. 152-160, October 1961.

17. Gault, J. W., Robinson, J.P. and Reddy, S. M., "Multiple Fault Detection in Combinational Networks," IEEE Trans. on Computers, vol. C-21, pp. 31-36, January 1972.

18. Gonenc, G., "A Method for the Design of Fault Detection Experiments," IEEE Trans. on Computers, vol. C-19, pp. 551-558, June 1970.

19. Hays, J. P., "On Realizations of Boolean Functions Requiring a Minimal or Near-Minimal Number of Tests." IEEE Trans. on Computers, vol, C-20, pp. 1506-1513, December 1971.

20. Hays, J. P., "On Modifying Logic Networks to Improve Their Diagnosability," IEEE Trans. on Computers, vol. C-23, pp. 56-62, January 1974.

21. Hennie, F. C., "Fault Detecting Experiments for Sequential Circuits," Proc. of the 5th Annual Switching Theory and Logic Design Symposium, S-164, pp. 95-110, November 1964.

22. Hsieh, E. P., "Checking Experiments for Sequential Machines," IEEE Trans. on Computers, vol. C-20, pp. 1152-1166, October 1971.

23. Kautz, W. H., "Fault Testing and Diagnosis in Combinational Digital Circuits." IEEE Trans. on Computers, vol. C-17, pp. 352-366, April 1968.

24. Kime, C. R., "An Organization for Checking Experiments on Sequential Circuits," IEEE Trans. on Electronic Computers, vol. EC-15, pp. 113-115, 1966.

25. Kohavi, Z. and Lavallee, P., "Design of Sequential Machines with Fault Detection Capability," IEEE Trans. on Electronic Computers, vol. EC-16, pp. 473-484, August 1967.

26. Kohavi, I. and Kohavi, Z., "Variable Length Distinguishing Sequences and Their Application to the Design of Fault Detection Experiments," IEEE Trans. on Computers, vol. C-17, pp. 792-795, 1968.

27. Ku, C. T. and Masson, G. M., "The Boolean Difference and Multiple Fault Analysis," IEEE Trans. on Computers, vol. C-24, pp. 62-71, January 1975.

28. Manning, E., "On Computer Self-Diagnosis : Part I - Experimental Study of a Processor, Part II - Generalizations and Design Principles," IEEE Trans. on Electronic Computers, vol. EC-15, pp. 873-890, December 1966.

29. Mayeda, W. and Ramamothy, C. V., "Distinguishability Criteria in Oriented Graphs and Their Application to Computer Diagnosis : Part I," IEEE Trans. on Circuit Theory, vol. CT-16, pp. 448-454, 1969.

30. McCluskey, E. J. and Clegg, F. W., "Fault Equivalence in Combinational Logic Networks," IEEE Trans. on Computers, vol. C-20, pp. 1286-1293, November 1971.

31. Moore, E. F., "Gedanken-Experiments on Sequential Machines in Automata Studies," Princeton University Press, Princeton, N.J., pp. 129-153, 1956.

32. Poage, J. F. and McCluskey, E. J., "Derivation of Optimum Test Sequences for Sequential Machines," Proc. of the 5th Annual Symp. Switching Theory and Logic Design, pp. 121-123, November 1964.

33. Powell, T. J., " A Procedure for Selecting Diagnostic Tests," IEEE Trans. on Computers, vol. C-18, pp.168-175, Feburary 1969.

34. Ramamothy, C. V., "A Structure Theory of Machine Diagnosis," Proc. AFIPS, SJCC, pp. 743-756, 1967.

35. Reddy, S. M., "Complete Test Sets for Logic Functions," IEEE Trans. on Computers, vol. C-22, pp. 1016-1020, November 1973.

36. Roth, J. P., "Diagnosis of Automata Failures : A Caluculus and a Method," IBM J. Res. Develop, vol. 10, pp. 278-294, July 1966.

37. Roth, J. P. Bouricius, W. G. and Schneider, P. R., "Programmed Algorithms to Compute Tests to Detect and Distinguish between Failures in Logic Circuits," IEEE Trans. on Electronic Computers, vol. EC-16, pp. 567-580, October 1967.

38. Roy, B. K., "Diagnosis and Fault Equivalence in Combinational Circuits," IEEE Trans. on Computers, vol. C-23, pp. 955-963, September 1974.

39. Russel, J. D. and Kime, C. R., "System Fault Diagnosis : Closure and Diagnosability with Repair," IEEE Trans. on Computers, vol. C-24, pp. 1078-1088, November 1975.

40. Russel, J. D. and Kime, C. R., "System Fault Diagnosis : Masking, Exposure and Diagnosability without Repair," IEEE Trans. on Computers, vol. C-24, pp. 1155-1160, December 1975.

41. Sakauchi, M. and Inose, H., "Synthesis of Fault Diagnosable Logic Circuits by Function Conversion Method," Trans. C of J IEC, vol. 56-D, No. 1, pp. 47-54, January 1973.

42. Sellers, F. F., Hsiao, M. Y and Bearson, L. W., "Analyzing Errors with the Boolean Difference," IEEE Trans. on Computers, vol. C-17, pp. 676-683, July 1968.

43. Seshu, S. and Freeman, D. N., "The Diagnosis of Asynchronous Sequential Switching System," IRE Trans. on Electronic Computers, vol. EC-11, pp. 459-465, 1962.

44. Seshu, S., "On an Improved Diagnosis Program," IEEE Trans. on Electronic Computers, vol. EC-14, pp. 76-79, 1965.

45. Shooman, M. L., "Probabilistic Reliability : an Engineering Approach," McGraw-Hill Book Co., New York, 1968.

46. Smith, G. R. and Yau, S. S., "A Programmed Fault Detection Algorithm for Combinational Switching Networks," Proc. Nat. Electron. Conf., vol. 25, pp. 668-673, December 1969.

47. Su, S. Y. H. and Cho, Y. C., "A New Approach to Fault Location of Combinational Circuits," IEEE Trans. on Computers, vol C-21, pp. 21-30, January 1972.

48. To, K., "Fault Folding for Irredundant and Redundant Combinational Circuits," IEEE Trans. on Computers, vol. C-23, pp. 955-963, September 1973.

49. Williams, M. J. Y. and Angell, J. B., "Enhancing Testability of Large Scale Integrated Circuits via Test Points and Additional Logics," IEEE Trans. on Computers, vol. C-22, pp. 46-60, January 1973.

50. Yau, S. S. and Tang, Y. S., "An Efficient Algorithm for Generating Complete Test Sets for Combinational Logic Circuits," IEEE Trans. on Computers, vol. C-20, pp. 1245-1251, November 1971.

AUTOBIOGRAPHY

Chin S. Chuang was born in Taipei, Taiwan, on May 30, 1948. He received the B.S. degree in electrophysics from National Chiao Tung University, Hsichu, Taiwan, in 1970, and the M.S. degree in electrical engineering from Polytechnique Institute of Brooklyn, Brooklyn, New York, in 1972, and the Ph.D. degree in electrical engineering from the City University of New York, New York, in 1976.

His current research interests are in fault-tolerant computing and microprocessor applications.

Dr. Chuang is a member of the Institute of Electrical and Electronics Engineers.