RELIABILITY ANALYSIS
OF SEQUENTIAL CIRCUITS

BY

Pierre Des Marais

Submitted to the Department of Electrical Engineering in partial fulfilment of the requirements for the degree

of

Master of Applied Science

Department of Electrical Engineering

Faculty of Science and Engineering

University of Ottawa

Ottawa, Ontario.

June, 1972.

ABSTRACT

This thesis introduces a new approach to the computation of the probabilistic mapping matrix associated with a sequential circuit whose components are characterized with a known probability of malfunctioning. The concept of path sensitizing is used in obtaining a graphical representation for the propagation of errors within a circuit, and an efficient procedure is thereby developed for computing the matrix. The resulting error propagation method is found to overcome the computational drawbacks outlined in a critical survey of other methods. Furthermore, with the physical insight that the graphical representation provides, the method is easily extended to the detection of faults in combinational circuits, and procedures are developed for the generation of various test sets. Finally, a meaningful approach is suggested for further research with respect to the reliability analysis of very large circuits and the synthesis of more reliable circuits.

ACKNOWLEDGEMENTS

I am most grateful to my advisor, Professor M. Krieger, for his guidance, enthusiasm, and friendly encouragement throughout the course of the research.

I wish to acknowledge the financial assistance of the Government of Ontario (O.G.F. scholarship), the IBM Corp. (research grants RJ1 and RJ3), and the National Research Council of Canada (research grant A5157).

I also wish to extend very special thanks to Miss S. Danchinko for her patience in typing the dissertation.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

A probabilistic sequential machine has been generally considered as an abstract model in the fields of automata and of stochastic processes [1,2,3,4,5,6,15], where the probabilistic nature of its input-output response and state behavior has been usually defined with respect to a probabilistic matrix. However, over the past years a number of studies [7,8,9,10,11,12] have recognized the usefulness of the probabilistic sequential machine as a model for the reliability of a sequential (or combinational) circuit, whereby the probability of erroneous operation in the physical circuit could be estimated through the computation of the probabilistic matrix associated with the model. It is from this latter point of view that the reliability of sequential circuits shall be analysed in this study.

From a physical point of view, each component of a sequential circuit is characterized by a non-zero probability of malfunctioning. In order to determine the effect of these malfunction probabilities on the reliability of the overall circuit, the sequential circuit is modeled as a probabilistic sequential machine, and the corresponding probabilistic matrix is computed from a knowledge of the malfunction probabilities. This matrix, which shall be more specifically referred to as the probabilistic mapping matrix, can in turn be used to provide precise information about various aspects of the circuit's reliability.

Hence, the major task involved with the above method of analysis consists in computing the probabilistic mapping matrix. Several methods have been developed for that purpose [8,9,11], but the resulting procedures become

rapidly limited by excessively tedious computations as the circuit increases in size and complexity. The main purpose of this study is therefore to develop an efficient procedure for computing the probabilistic mapping matrix.

The material presented in this thesis can be divided into three major parts. The first part corresponds to Chapter II, which outlines the characteristics and uses of the probabilistic mapping matrix. The second part consists of Chapters III and IV, which describe various methods for computing the matrix. These methods are compared with respect to an example given in the Appendix. The final part corresponds to Chapter V, which extends the results of this research to the detection of faults in combinational circuits.

In Chapter II, the probabilistic mapping matrix is defined, its characteristics are outlined, and its computational uses are investigated with respect to obtaining specific information about the reliability of the corresponding circuit. Although the chapter is essentially a review of some well known results, its main purpose is to emphasize the physical interpretation of these results and to relate them more particularly to the problems that will be considered in the following chapters.

Chapter III presents a critical survey of two existing methods for computing the probabilistic mapping matrix, namely a matrix method [7] and an algebraic method [8]. In each case, a brief description of the method is given, and its advantages and disadvantages are discussed. The purpose of the chapter is to outline the major computational drawbacks of these methods, and thereby point out the need for a different approach to the problem.

In Chapter IV, a new approach to the problem of computing the probabilistic matrix is investigated. The concept of path sensitizing is used to obtain a graphical representation for the propagation of errors in a circuit. As with the other methods, these errors are assumed to be temporary, i.e. they

correspond to intermittent errors that can arise from loose wires, circuit noise, temporary overheating, critical timing, and unusual conditions on primary power input. The resulting error propagation method provides much physical insight into the problem, and an efficient procedure is thereby developed. Finally, the method is compared to those considered in Chapter III with respect to the example given in the Appendix.

In Chapter V, the results obtained with the method of analysis developed in the previous chapter are extended to the detection of faults in combinational circuits. Since fault detection is achieved with respect to permanent faults, as in the case of shorted or open transitors and diodes, the error propagation method is slightly modified, and a procedure is developed for detecting single faults in a circuit. The chapter also discusses the testing philosophy associated with fault detection, and it illustrates the flexibility of the error propagation method in generating limited test sets with respect to that testing philosophy.

Finally, Chapter VI concludes the thesis by discussing several topics of further research for which the error propagation method should provide a meaningful approach. The results developed in this study are thereby situated within the general topic of circuit reliability.

The major contribution of this thesis corresponds to the physical insight that it provides not only into the specific problem of computing the probabilistic mapping matrix, but also with respect to other related problems of circuit reliability. In allowing the present research to maintain a close relation to the physical aspects of the problems investigated, this insight provides both a significant computational simplification in computing the probabilistic mapping matrix and a unified approach to the problems of fault detection and reliability analysis. Furthermore, it suggests a meaningful approach to the problem of reliable circuit synthesis.

CHAPTER II

GENERAL BACKGROUND

This chapter introduces the probabilistic mapping matrix associated with a probabilistic sequential machine and examines its characteristics and computational uses. As these concepts are very basic, they will be reviewed informally. The chapter concludes with a short survey of applications of the probabilistic mapping matrix.

1. The Probabilistic Mapping Matrix

A sequential machine may be mathematically defined by the 6-tuple $< U, V, S, f, g, S_0 >$, where

$U = \{u_1, u_2, \ldots, u_\ell\}$ is an alphabet of input symbols,

$V = \{v_1, v_2, \ldots, v_m\}$ is an alphabet of output symbols,

$S = \{s_1, s_2, \ldots, s_n\}$ is the set of states of the machine,

f: $S \times U \rightarrow S$ is the next-state (or state transition) mapping,

g: $S \times U \rightarrow V$ is the output mapping, and

$S_0 \subseteq S$ is the set of initial states.

In the above definition, f is referred to as a mapping, since it associates with each state-input pair (s_i, u_p) a unique next-state s_j to which the machine will go when input u_p is applied to the machine in state s_i. Similarly, g is also a mapping, but its definition distinguishes between the Mealy and Moore models of sequential machines. In the case of the Mealy model, g is defined as above as it associates a unique output symbol v_r with each state-input pair (s_i, u_p). In the case of the Moore model, the output of the machine is independent of the input to the extent that it is associated with the state alone, i.e. g: $S \rightarrow V$.

By contrast with the above deterministic (completely specified) sequential machine, a probabilistic sequential machine (hereafter abbreviated as p.s.m.) is characterized by the probabilistic nature of mappings f and g. In

fact, f and g are no longer mappings, but relations such that for each pair (s_i, u_p) the machine may go to any state s_j with any output v_r with some probability $p(s_j, v_r/s_i, u_p)$.

We will therefore define a p.s.m. by the 5-tuple $\langle U, V, S, \Pi(\lambda), P \rangle$, where U, V, and S are defined as before,

$\Pi(\lambda) = \left[\pi_1(\lambda), \pi_2(\lambda), \ldots, \pi_n(\lambda)\right]$ is the initial state (probability distribution) vector, where $\pi_j(\lambda)$ is the probability that the machine is initially in state s_j,

and where $\sum_{j=1}^{n} \pi_j(\lambda) = 1$, and

$P = \left[p(s_j, v_r/s_i, u_p)\right]$, with $i, j = 1, 2, \ldots, n$;

$p = 1, 2, \ldots, \ell$;

$r = 1, 2, \ldots, m$;

is the $(n\ell \times nm)$ probabilistic mapping matrix, where each entry $p(s_j, v_r/s_i, u_p)$ represents the conditional probability that the machine will go to state s_j with output v_r, given that input u_p is applied to the machine in state s_i.

Matrix P is given in the following format:

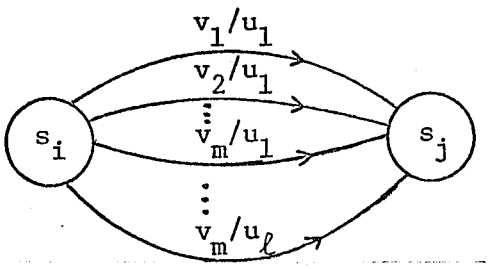
$$P = \begin{bmatrix} M_{11} & M_{12} & \cdots & M_{1n} \\ M_{21} & M_{22} & \cdots & M_{2n} \\ \vdots & \vdots & & \vdots \\ M_{n1} & M_{n2} & & M_{nn} \end{bmatrix}$$

where the $(\ell \times m)$ submatrix $M_{ij} = \left[m_{pr}\right]$,

where $m_{pr} = p(s_j, v_r/s_i, u_p)$, with $p = 1, 2, \ldots, \ell$;

$r = 1, 2, \ldots, m$.

The state diagram corresponding to M_{ij} is shown below:



where branch v_r/u_p is weighted with the probability $p(s_j, v_r/s_i, u_p)$

Thus, matrix P provides all the information necessary to describe the "inside the black box" operation of the p.s.m., since both relations f and g are defined by the probabilistic entries of P.

In reality, state relation f, output relation g, and initial state vector $\Pi(\lambda)$ need not be all probabilistic. Considering a p.s.m. as a sequential transducer which transforms input sequences $u_{p_1} u_{p_2} \ldots u_{p_k}$ into output sequences $v_{r_1} v_{r_2} \ldots v_{r_k}$ with a probability $p(v_{r_1} \ldots v_{r_k}/u_{p_1} \ldots u_{p_k})$, it is sufficient that only one of the above three be probabilistic to provide the sequential machine with a probabilistic input - output response.

2. Characteristics of Matrix P

Matrix P is characterized by the following basic properties of its entries:

(i) $0 \leq p(s_j, v_r/s_i, u_p) \leq 1$, for all i, j, p, r.

(ii) $\sum_{r=1}^{m} \sum_{j=1}^{n} p(s_j, v_r/s_i, u_p) = 1$, for all i, p; i.e. the sum of all entries of any row (i,p) of P is 1.

Note that a deterministic sequential machine is a special case of a p.s.m. for which the entries of matrix P are either 0 or 1, and for which $\Pi(\lambda)$ contains a single non-zero entry.

Thus, P is a stochastic matrix. Furthermore, within the context of the present study, the entries of P are assumed to be time-independent. Thus, a p.s.m. may be represented as a stationary Markov process. Such a representation is used by Glinski and Chung [11, 12] in estimating the reliability of a sequential circuit.

Furthermore, within the context of the present study, matrix P is defined with respect to the entire state-space (defined by the state variables) and the entire output-space (defined by the output variables) of the given sequential circuit. Moreover, by definition of its probabilistic entries, $p(s_j, v_r/s_i, u_p)$, matrix P corresponds to a single transition from any state s_i to any state s_j with any output symbol v_r, under the application of any input symbol u_p. In the case of a synchronous sequential circuit, each and every state of the state-space is considered as a stable state, and each time interval corresponds to a single state transition. Hence, the definition of matrix P corresponds exactly to the mode of operation of a synchronous circuit. However, in the case of an asynchronous sequential circuit, not all states are considered as stable states, as the "interval" of circuit operation, considered as a transition from one stable state to another stable state, generally involves one or more intermediate transitions to "transient" states. Thus, the corresponding probabilistic mapping matrix is a "reduced" matrix which is defined with respect to the subset of stable states. However, note that the results developed in the next section remain valid with respect to this reduced matrix.

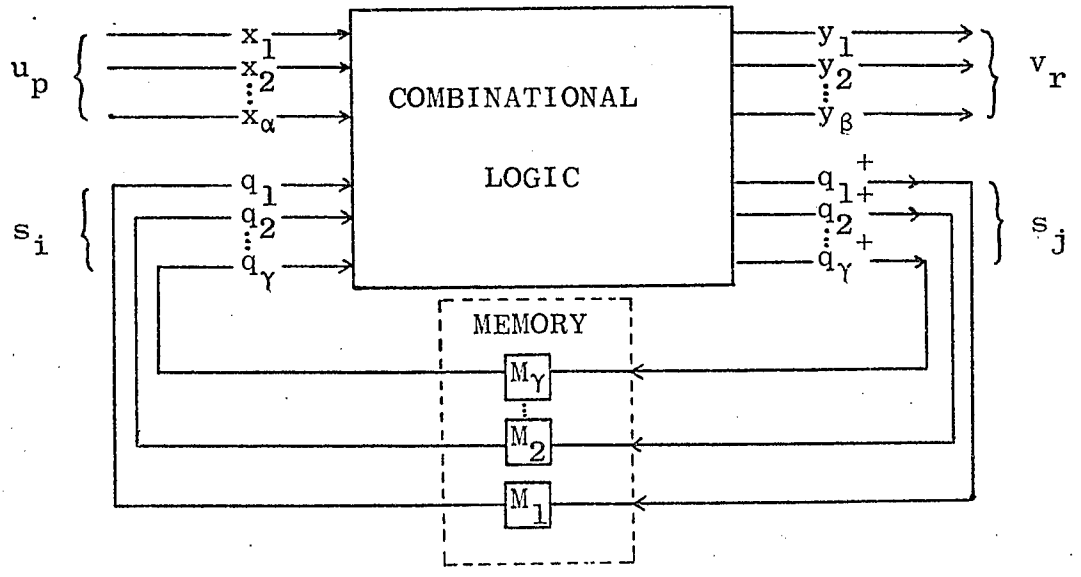


Fig. 2.1

Next, let us investigate additional properties of matrix P with respect to structual considerations of sequential machines. A sequential machine is generally given in the form shown in Fig. 2.1. As outlined in the previous section, there are two structural types of sequential machines: the Mealy model and the Moore model.

In both models, next-state s_j is defined as a function of the present-state s_i and of the input u_p applied, i.e. $s_j = f(s_i, u_p)$. Consequently, each next-state variable is defined as follows:

$q_\omega^+ = F_\omega(Q, X)$, $\omega = 1, 2, \ldots, \gamma$; where $Q = \{q_1, q_2, \ldots, q_\gamma\}$ is the set of state variables, $X = \{x_1, x_2, \ldots, x_\alpha\}$ is the set of input variables, and F_ω is a Boolean function of present-state and input variables.

In a p.s.m., the corresponding state transition probability, $p(s_j/s_i, u_p)$, is obtained from matrix P as follows:

$$p(s_j/s_i, u_p) = \sum_{r=1}^{m} p(s_j, v_r/s_i, u_p).$$

On the other hand, output v_r is defined differently in each model. For the Mealy model, the output is considered as a function of the present-state and of the input, i.e. $v_r = g^{ME}(s_i, u_p)$. Thus, each output variable is defined as follows:

$y_\delta = G_\delta^{ME}(Q,X)$, $\delta = 1, 2, \ldots, \beta$; where G_δ^{ME} is a Boolean function of present-state and input variables.

In a p.s.m., the corresponding output probability, $p(v_r/s_i, u_p)$, is obtained from matrix P as follows:

$$p(v_r/s_i, u_p) = \sum_{j=1}^{n} p(s_j, v_r/s_i, u_p).$$

For the Moore model, the output is said to be associated with the state alone. As defined originally by Moore [13], output v_r is a function of the present-state s_i, i.e. $v_r = g^{MO}(s_i)$, and each output variable $y_\delta = G_\delta^{MO}(Q)$, where G_δ^{MO} is a Boolean function of present-state variables.

With respect to sequential machine structure, this latter model is a special case of a Mealy model in which output variables are independent of input variables. Thus, in a p.s.m., the corresponding output probability, $p(v_r/s_i)$, is obtained from matrix P as in the case of the Mealy model, i.e. $p(v_r/s_i) = \sum_{j=1}^{n} p(s_j, v_r/s_i, u_p)$, but $p(v_r/s_i)$ is independent of the input u_p (p = 1, 2, ..., ℓ) chosen for the above summation.

However, if the Moore output is considered as being associated with the next-state of the machine, the combinational circuit of the machine can be considered in two different ways, as shown in Fig. 2.2.

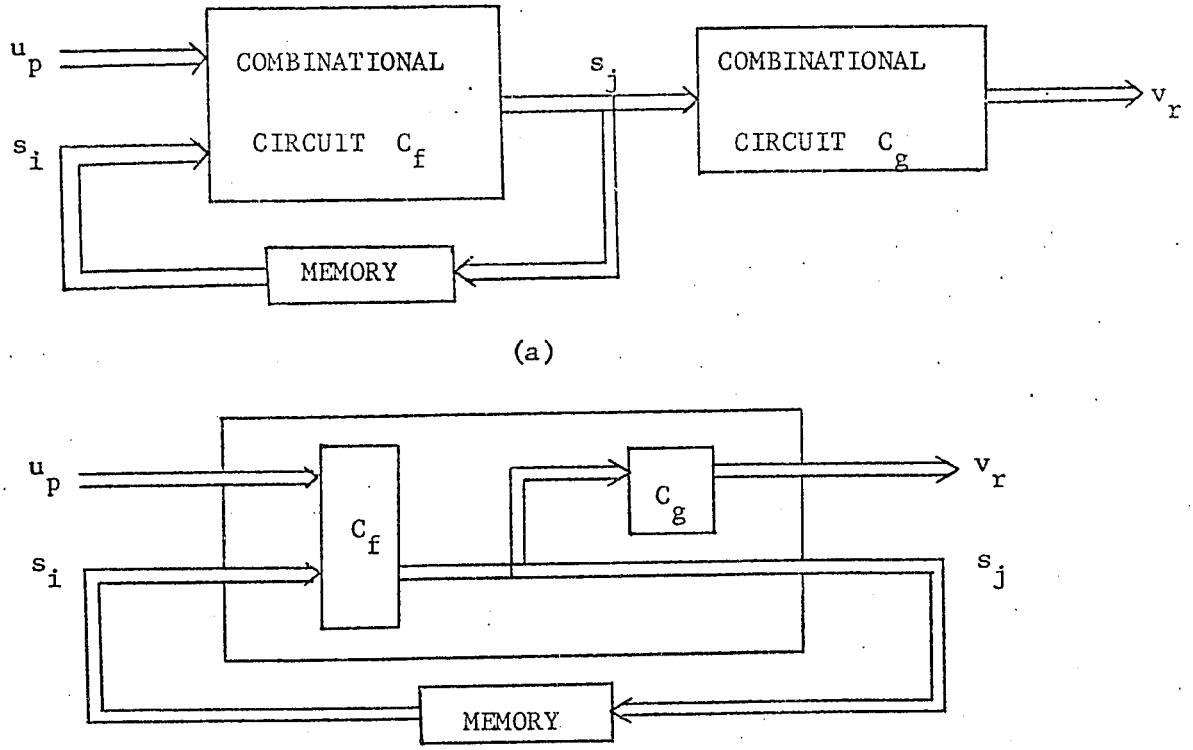


Fig. 2.2

From a structural point of view, the two configurations are identical, but with respect to reliability analysis (as developed in Chapter IV), they are analysed differently. In Fig. 2.2 (a), the combinational logic of the Moore model is considered as two separate circuits. Output v_r is a function of

next-state s_j, i.e. $v_r = g^{MO}(s_j)$, and each output variable $y_\delta = G_\delta^{MO}(Q^+)$, where $Q^+ = \{ q_1^+, q_2^+, \ldots, q_\gamma^+ \}$ and G_δ^{MO} is now a Boolean function of next-state variables. Hence, the corresponding output probability, $p(v_r/s_j)$, is characterized by the relation $p(s_j, v_r/s_i, u_p) = p(s_j/s_i, u_p)\, p(v_r/s_j)$, and it is therefore obtained from matrix P as follows:

$$p(v_r/s_j) = \frac{p(s_j, v_r/s_i, u_p)}{p(s_j/s_i, u_p)} = \frac{p(s_j, v_r/s_i, u_p)}{\sum_{r=1}^{m} p(s_j, v_r/s_i, u_p)}$$

Note that $p(v_r/s_j)$ is independent of the state-input pair (s_i, u_p) chosen for the above summation.

On the other hand, the combinational logic of the Moore model shown in Fig. 2.2 (b) is considered as a single circuit. Hence, within the context of the reliability analysis developed in Chapter IV, $p(s_j, v_r/s_i, u_p) \neq p(s_j/s_i, u_p)\, p(v_r/s_j)$ and the model is considered as a special case of a Mealy model, i.e. the output probability corresponds to $p(v_r/s_i, u_p) = \sum_{j=1}^{n} p(s_j, v_r/s_i, u_p)$.

The model associated with Fig. 2.2 (a) corresponds to the probabilistic Moore machine defined by Salomaa [5]. That model will be used as the representative Moore model in the next section, since the other interpretations are considered as special cases of the Mealy model.

To conclude this section, let us discuss the above results within the context of the present research. In this study, a p.s.m. is considered as a model of a (deterministic) sequential circuit whose components can malfunction, and the probabilistic entries of matrix P reflect the non-zero probability of malfunction in each circuit component. In Chapter IV, the reliability of a sequential circuit, as shown in Fig. 2.1 (Mealy model), is analysed by developing a procedure for computing the corresponding probabilistic mapping matrix (the results can be easily extended to the Moore model of Fig. 2.2 (a)).

A detailed physical interpretation of the above probabilities will then be given with respect to component malfunctions. However, at this point, let us consider how these probabilities are related.

As a mathematical model, a p.s.m. is often defined with respect to the probability $p(s_j, v_r/s_i, u_p)$ [1,5,15]. Booth [1] recognizes the fact that, in the general case, $p(s_j, v_r/s_i, u_p) \neq p(s_j/s_i, u_p)\, p(v_r/s_i, u_p)$. On the other hand, Salomaa [5] defines a Mealy - type p.s.m., characterized by the relation $p(s_j, v_r/s_i, u_p) = p(s_j/s_i, u_p)\, p(v_r/s_i, u_p)$, and a Moore - type p.s.m., characterized by the relation $p(s_j, v_r/s_i, u_p) = p(s_j/s_i, u_p)\, p(v_r/s_j)$ considered previously. From a physical (structural) point of view, relation $p(s_j, v_r/s_i, u_p) = p(s_j/s_i, u_p)\, p(v_r/s_i, u_p)$ is obtained only in the case where output functions G_δ ($\delta = 1, 2, \ldots, \beta$) are realized independently of next-state functions F_ω ($\omega = 1, 2, \ldots, \gamma$), and the corresponding circuits are analysed as two separate circuits. However, in general, the design equations corresponding to Boolean functions G_δ and F_ω will not be implemented independently; for economical reasons, some of the functions will share common components. Hence, it will be found that, in general, $p(s_j, v_r/s_i, u_p) \neq p(s_j/s_i, u_p)\, p(v_r/s_i, u_p)$. These points will be considered in greater detail in Chapter IV.

3. Computational Uses of Matrix P

In analysing the state transition behavior and output response of a p.s.m., matrix P does not lend itself directly to computations involving matrix operations. However, by simple manipulations on P, various matrices are obtained and used as follows.

(i) input-output matrices:

$$P\ (v_r/u_p) = \left[m_{ij}\right], \text{ where } m_{ij} = p(s_j, v_r/s_i, u_p),$$
$$\text{with } i, j = 1, 2, \ldots, n.$$

These (n x n) matrices describe the probabilistic input-output relationship of the machine, and their entries are obtained directly from P. Clearly, there are ℓm such matrices for ℓ-input and m-output alphabets.

These input-output matrices can be used directly for computing the probability of obtaining a specific output sequence $v_{r_1}\ v_{r_2} \ldots v_{r_k}$, given that an input sequence $u_{p_1}\ u_{p_2} \ldots u_{p_k}$ is applied to the p.s.m. with an initial-state vector $\Pi(\lambda)$:

$$p(v_{r_1} \ldots v_{r_k}/u_{p_1} \ldots u_{p_k}) = \Pi(\lambda)\ P\ (v_{r_1}/u_{p_1}) \ldots P(v_{r_k}/u_{p_k}) \begin{bmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} (n \times 1)$$

(ii) transition matrices:

$$F\ (u_p) = \left[f_{ij}\right], \text{ where } i, j = 1, 2, \ldots n;$$
$$\text{and where } f_{ij} = p\ (s_j/s_i, u_p) = \sum_{r=1}^{m} p\ (s_j, v_r/s_i, u_p).$$

These (n x n) matrices describe the probabilistic state transition behavior of the machine (without consideration of the output), and their entries are obtained by adding appropriate entries of matrix P, as shown above. Clearly, there is a transition matrix for each symbol of the input alphabet, and each transition matrix is stochastic, i.e. $\sum_{j=1}^{n} f_{ij} = 1$.

Transition matrices are used for computing the final state probabilities after some input sequence $u_{p_1}\ u_{p_2}\ \ldots\ u_{p_k}$ is applied to the p.s.m. with initial-state vector $\Pi(\lambda)$:

$\Pi(u_{p_1}\ u_{p_2}\ \ldots\ u_{p_k}) = \Pi(\lambda)\ F\ (u_{p_1})\ F\ (u_{p_2})\ \ldots\ F\ (u_{p_k})$,

where $\Pi\ (u_{p_1}\ u_{p_2}\ \ldots\ u_{p_k})$ is an n-dimensional row vector whose j^{th} entry, $\pi_j(u_{p_1}\ u_{p_2}\ \ldots\ u_{p_k})$, represents the probability that the machine is in state s_j after the input sequence is applied.

Note that $\sum_{j=1}^{n} \pi_j(u_{p_1}\ u_{p_2}\ \ldots\ u_{p_k}) = 1$

(iii) output matrices:

(a) Mealy output

$G(u_p) = \left[g_{ir}\right]$, where $i = 1, 2, \ldots, n$; $r = 1, 2, \ldots, m$;

and where $g_{ir} = p(v_r/s_i, u_p) = \sum_{j=1}^{n} p(s_j, v_r/s_i, u_p)$

These (n x m) matrices describe the output behavior of the machine, and their entries are obtained by adding entries of matrix P, as shown above. Since a Mealy output depends both on the state and on the input, there is one output matrix for each symbol of the input alphabet. Also, each output matrix is stochastic, i.e. $\sum_{r=1}^{m} g_{ir} = 1$.

Output matrices are useful for computing the final output probabilities after some input sequence $u_{p_1}\ u_{p_2}\ \ldots\ u_{p_k}$ is applied to the p.s.m. with initial-state vector $\Pi(\lambda)$:

$\left[p(v_1),\ p(v_2),\ \ldots,\ p(v_m)\right] = \Pi(\lambda)\ F\ (u_{p_1})\ F\ (u_{p_2})\ \ldots\ F\ (u_{p_{k-1}})\ G\ (u_{p_k})$,

where $\left[p(v_1),\ p(v_2),\ \ldots,\ p(v_m)\right]$ is an m-dimensional vector whose r^{th} entry, $p(v_r)$, represents the probability that the machine will have final output v_r, and $\sum_{r=1}^{m} p(v_r) = 1$,

and where $\Pi(\lambda)\ F\ (u_{p_1})\ \ldots\ F\ (u_{p_{k-1}})$ gives the state probabilities after application of $u_{p_1}\ u_{p_2}\ \ldots\ u_{p_{k-1}}$. The final multiplication by matrix $G\ (u_{p_k})$ yields the final output probabilities (m-dimensional row vector) after the last input, u_{p_k}, is applied.

Remarks: (1) if one is interested only in the probability of obtaining a particular symbol v_r as a final output, then the final matrix multiplication will involve only the column vector corresponding to the r^{th} column of matrix $G(u_{p_k})$, instead of the entire matrix $G(u_{p_k})$.

(2) one may compute the output probabilities after application of any input symbol u_{p_w} of the sequence $u_{p_1}\ u_{p_2}\ \ldots\ u_{p_k}$ by post-multiplying the product $\Pi(\lambda)\ F\ (u_{p_1})\ F\ (u_{p_2})\ \ldots\ F\ (u_{p_{w-1}})$ by matrix $G\ (u_{p_w})$.

(b) Moore output

In this case, the output is independent of the input, since it is associated with the state alone. Therefore, by contrast with the Mealy case, there is a single (n x m) output matrix

$G = \left[g_{jr}\right]$, where $j = 1, 2, \ldots, n$; $r = 1, 2, \ldots, m$;

and where $g_{jr} = p(v_r/s_j) = \dfrac{p(s_j, v_r/s_i, u_p)}{p(s_j/s_i, u_p)}$

$$= \frac{p(s_j, v_r/s_i, u_p)}{\sum_{r=1}^{m} p(s_j, v_r/s_i, u_p)}$$

for any state-input pair (s_i, u_p).

Again, note that matrix G is stochastic, i.e. $\sum_{r=1}^{m} g_{jr} = 1$.

As in the Mealy case, output matrix G is used for computing final output probabilities after some input sequence $u_{p_1}\ u_{p_2}\ \ldots\ u_{p_k}$ is applied to the p.s.m. with initial-state vector $\Pi(\lambda)$:

$$\left[p(v_1),\ p(v_2),\ \ldots,\ p(v_m)\right] = \Pi(\lambda)\ F\ (u_{p_1})\ F(u_{p_2})\ \ldots\ F\ (u_{p_k})\ G,$$

where $\sum_{r=1}^{m} p(v_r) = 1$.

Note that, in this case, matrix G post-multiplies the product $\Pi(\lambda)\ F\ (u_{p_1})\ \ldots\ F\ (u_{p_k})$ which includes the transition matrix corresponding to

the final input u_{p_k}. This difference arises from the fact that the probabilistic Moore output is associated with the state to which the machine goes after the input is applied.

Remarks similar to those made in the Mealy case also apply in this case.

4. Applications Involving Matrix P

To conclude this chapter, applications of the probabilistic mapping matrix (and of its associated matrices) in various related fields are outlined very briefly.

One of the main topics to require the use of the probabilistic mapping matrix is the reliability of sequential circuits. From that practical point of view, the p.s.m. is considered as a model of a deterministic sequential circuit which exhibits erroneous behavior because of random malfunctioning of its components. In that case, the probabilistic mapping matrix may be used directly for estimating the reliability of the circuit. In [7], W. Chung applies $\mathcal{Z}$ - transform and signal flow graph techniques to the matrix in order to compute the mean-time-to-first-error of the circuit.

In the field of automata, state transition matrices are used extensively in determining the set of inputs (language) accepted by an automaton. In [3], Rabin defines a probabilistic automaton with respect to a cut-point λ, $0 \leq \lambda \leq 1$, whereby an input sequence is said to be accepted (recognized) if the probability that the automaton is in any one of a set of designated final states exceeds the value λ. Probabilistic automata and languages are studied in detail by Salomaa [5].

Transition matrices are also important to the problems of equivalence and stability of p.s.m.'s. The "stability" problem consists in characterizing the perturbations in the transition matrices which do not change the behavior of the machine. These problems were introduced by Rabin [3].

Finally, the limiting state behavior of a p.s.m. (or probabilistic automaton) can be characterized from a knowledge of the eigenvalues of the transition matrix. A short classification of state behavior is given by Warfield [2]. From the reliability point of view, J.T. Tou [6] describes a procedure whereby the eigenvalues of the deterministic transition matrix of an automaton and the eigenvalues of the corresponding probabilistic transition matrix are computed, and the differences between powers of corresponding eigenvalues are used to estimate the reliability of the automaton.

Thus, it appears that the information provided by the probabilistic mapping matrix is essential not only to the reliability analysis of sequential circuits, but also to many other related topics.

CHAPTER III

EXISTING METHODS FOR COMPUTING THE PROBABILISTIC MAPPING MATRIX

The purpose of this chapter is to present a critical survey of methods which have been developed for obtaining the probabilistic mapping matrix of a sequential circuit. Two representative methods are considered: a matrix method and an algebraic method. Both methods are described to some detail in order to indicate clearly the drawbacks which create a need for a different approach to the problem, but, in each case, the description is presented informally.

The following assumptions are made for both methods:

(i) malfunction probabilities are time-independent and are known for every component of the given circuit.

(ii) all malfunctions are considered transient and statistically independent.

(iii) second and higher order products of malfunction probabilities are neglected.

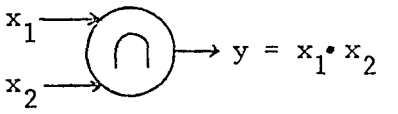
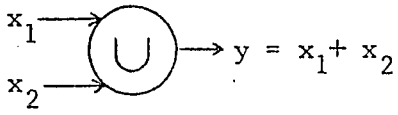
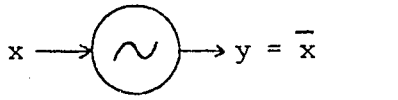
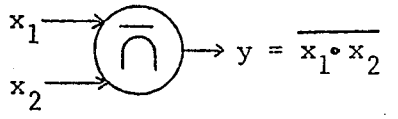
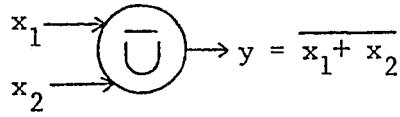
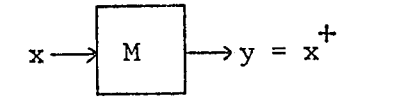
1. Matrix Method

The basic concepts of this method were introduced by Udagawa and Inagaki [9], and by Levin [10]. The method described in the following was presented by Glinski and Chung [11].

With this matrix method, the probability of failure in a component is represented in a matrix format. Then the procedure consists essentially in combining these matrices according to a pertinent partitioning of the circuit, using appropriately defined matrix operations.

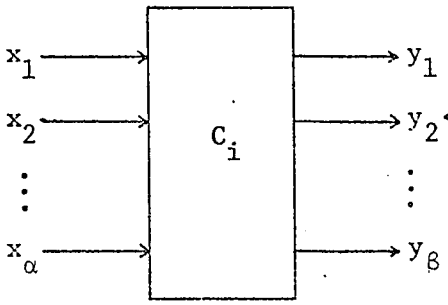
Error matrices corresponding to various types of components are given in Table 3.1. Note that, for each of the components considered in Table 3.1, $a_0 = a_1 = a_2 = a_3$ if the probability of failure is independent of input.

TABLE 3.1

COMPONENT	ERROR MATRIX
AND 	$P_{\wedge} =$ y: 0, 1 x_1x_2: 0 0, 0 1, 1 0, 1 1 $\begin{bmatrix} 1-a_0 & a_0 \\ 1-a_1 & a_1 \\ 1-a_2 & a_2 \\ a_3 & 1-a_3 \end{bmatrix}$
OR 	$P_{\vee} =$ y: 0, 1 x_1x_2: 0 0, 0 1, 1 0, 1 1 $\begin{bmatrix} 1-a_0 & a_0 \\ a_1 & 1-a_1 \\ a_2 & 1-a_2 \\ a_3 & 1-a_3 \end{bmatrix}$
NOT 	$P_{\sim} =$ y: 0, 1 x: 0, 1 $\begin{bmatrix} a_0 & 1-a_0 \\ 1-a_1 & a_1 \end{bmatrix}$
NAND 	$P_{\bar{\wedge}} =$ y: 0, 1 x_1x_2: 0 0, 0 1, 1 0, 1 1 $\begin{bmatrix} a_0 & 1-a_0 \\ a_1 & 1-a_1 \\ a_2 & 1-a_2 \\ 1-a_3 & a_3 \end{bmatrix}$
NOR 	$P_{\bar{\vee}} =$ y: 0, 1 x_1x_2: 0 0, 0 1, 1 0, 1 1 $\begin{bmatrix} a_0 & 1-a_0 \\ 1-a_1 & a_1 \\ 1-a_2 & a_2 \\ 1-a_3 & a_3 \end{bmatrix}$
MEMORY 	$P_M =$ y: 0, 1 x: 0, 1 $\begin{bmatrix} 1-a_0 & a_0 \\ a_1 & 1-a_1 \end{bmatrix}$

Combinations of Circuits:

In order to extend this matrix formulation to combinational circuits, several matrix operations must be defined, corresponding to the various ways in which circuits can be combined. With each combinational circuit C_i is associated a matrix P_i as follows:



$$\Longleftrightarrow P_i = \left[p_{jk}\right]_{(2^\alpha \times 2^\beta)}, \text{ with } j = 0,1,\ldots,2^\alpha-1; \quad k = 0,1,\ldots,2^\beta-1;$$

where p_{jk} is the probability that the k^{th} output β - tuple is obtained when the j^{th} input α - tuple is applied to the circuit.

The components considered in Table 3.1 are elementary combinational circuits.

The three basic ways in which two circuits, C_1 and C_2, can be combined, and the corresponding operations defined on their matrices, P_1 and P_2, are given in Table 3.2. In addition to these three combinations there is a fourth combination which consists of mixed "parallel" and "side-by-side" combinations. A mixed combination of two circuits is shown below:

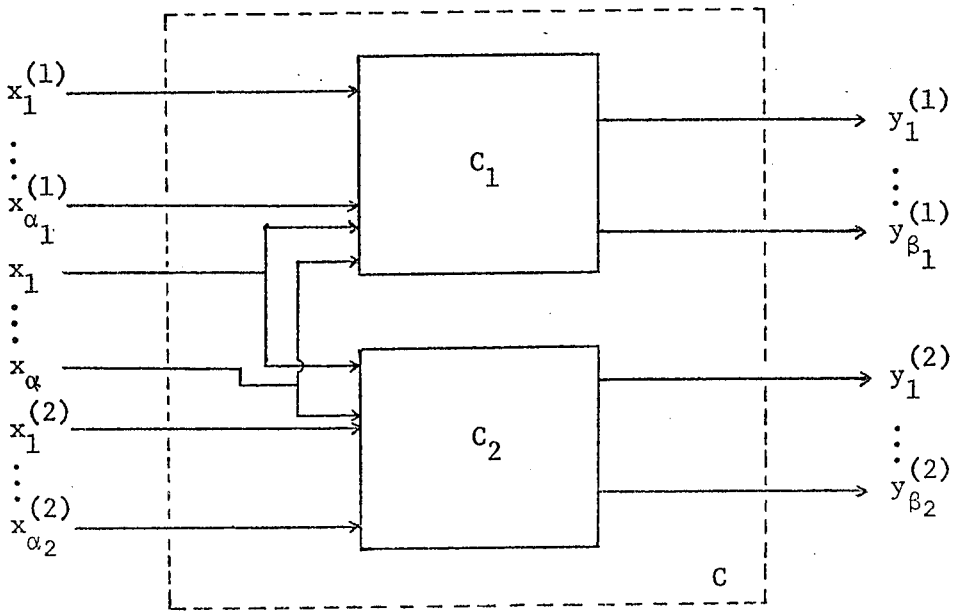
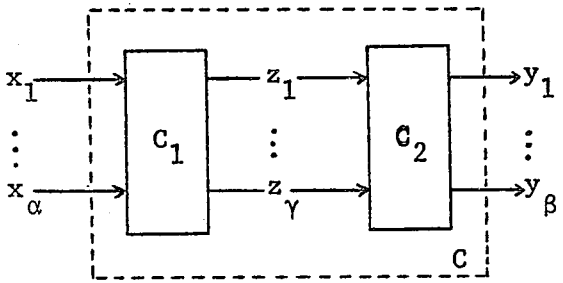
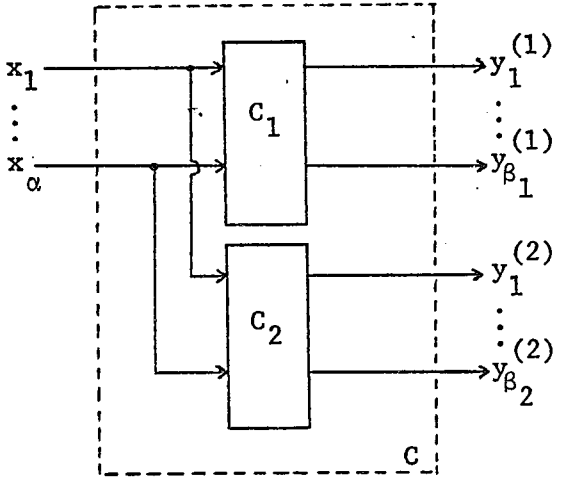
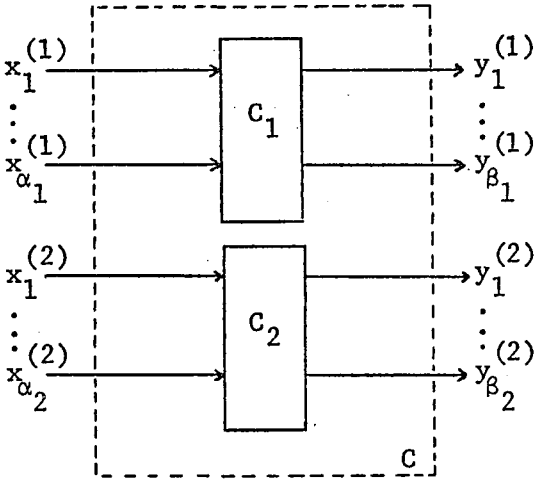


TABLE 3.2

SERIES COMBINATION	PARALLEL COMBINATION	SIDE BY SIDE COMBINATION
		
$\left[P\right]_{2^{\alpha}x\ 2^{\beta}} = \left[P_1\right]_{2^{\alpha}x\ 2^{\gamma}} \circ \left[P_2\right]_{2^{\gamma}x\ 2^{\beta}}$	$\left[P\right]_{2^{\alpha}x2^{\beta}1+\beta 2} = \left[P_1\right]_{2^{\alpha}x2^{\beta}1} * \left[P_2\right]_{2^{\alpha}x2^{\beta}2}$	$\left[P\right]_{2^{\alpha}1+\alpha 2x2^{\beta}1+\beta 2} = \left[P_1\right]_{2^{\alpha}1x2^{\beta}1} \textcircled{x} \left[P_2\right]_{2^{\alpha}2x2^{\beta}2}$
where $\circ$ denotes conventional matrix multiplication	where * is defined by $P_{i,jk} = p^{(1)}_{i,j}\ p^{(2)}_{i,k}$ with $i = 0,1, \ldots ,2^{\alpha}- 1;$ $j = 0,1, \ldots ,2^{\beta}1- 1;$ $k = 0,1, \ldots ,2^{\beta}2- 1.$	where $\textcircled{x}$ denotes the Kronecker product

In order to obtain the P - matrix of the overall circuit C, the "mixed" combination is transformed into a parallel combination. Thus, matrices P_1 and P_2 must be enlarged such that each matrix will be defined over all input variables:

let $X = (x_1^{(1)}, x_2^{(1)}, \ldots, x_{\alpha_1}^{(1)}, x_1, x_2, \ldots, x_\alpha, x_1^{(2)}, x_2^{(2)} \ldots, x_{\alpha_2}^{(2)})$ be the input $(\alpha_1 + \alpha + \alpha_2)$ - tuple which includes all input variables to C,

then $P_1^E = \left[p(y_1^{(1)}\ y_2^{(1)} \ldots\ y_{\beta_1}^{(1)}/X)\right]_{(2^{\alpha_1} + \alpha + \alpha_2 \times 2^{\beta_1})}$, where

$$p(y_1^{(1)} \ldots y_{\beta_1}^{(1)}/X) = p(y_1^{(1)} \ldots y_{\beta_1}^{(1)}/x_1^{(1)} \ldots x_{\alpha_1}^{(1)}\ x_1 \ldots x_\alpha).$$

Thus, in enlarging matrix P_1, each row of P_1 is duplicated $2^{\alpha 2}$ times, since the entries of P_1 are independent of the α_2 input variables which are applied to C_2 alone.

Enlarged matrix P_2^E is obtained in a similar way.

Thus, circuits C_1 and C_2, corresponding to enlarged matrices P_1^E and P_2^E respectively, are in parallel combination, and $P = P_1^E * P_2^E$.

Note: for simplicity the above combinations were restricted to two circuits, but the results can be easily extended to N circuits.

The Procedures:

The probabilistic mapping matrix corresponding to a given sequential circuit can now be computed by partitioning the circuit into blocks, and by using the matrix operations defined above to combine these blocks. There are two procedures for achieving that computation: the series-partitioning procedure and the parallel-partitioning procedure.

With the series - partitioning procedure, the circuit is divided into w blocks in series, where w is the number of components in the longest path from the state-input terminals to the next-state-output terminals. Thus, each block contains at most one component in each path. Then, matrix P_i of the i^{th}

block is computed, for i = 1, 2, ..., w, and the P-matrix of the sequential circuit is given by $P = P_1 \circ P_2 \circ \ldots \circ P_w$.

With the parallel-partitioning procedure, the circuit is now divided into w blocks in parallel, where w is the total number of output and next-state variables. The i^{th} block contains all the components between the present-state and input variables and the i^{th} next-state or output variable. Again, matrix P_i is computed, for i = 1, 2, ..., w, and the P-matrix of the circuit is given by $P = P_1 * P_2 * \ldots * P_w$. However, the resulting P-matrix requires an additional modification step to correct errors that occur whenever the output from one component is used as an input to more than one component. In that case, the errors arise from the fact that the component appears in more than one block, and its effect on the failure probabilities of the overall circuit is erroneously duplicated through the parallel combination of the blocks.

Remark: in the procedure described above, each memory element of the sequential circuit can be considered simply as another component.

Advantages and Drawbacks of the Matrix Method:

The main advantage of the matrix method lies in the simplicity of the procedures. Together with the fact that matrix operations are relatively easy to program, this makes the method fairly easy to computerize.

However, as the circuit to be analysed reaches a non-trivial size, the method becomes impractical as matrices become too large, requiring very lengthy operations. Since the procedures do not allow for any simplifications, the computation of the P-matrix becomes extremely tedious.

In the case of the series-partitioning method, the P-matrix is obtained directly by conventional matrix multiplication of the block matrices P_i. However, each block C_i will generally have a large number of input and output lines (some of which will not go through any component) such that the corresponding P_i matrices will reach relatively large dimensions.

By comparison with the "series" method, the "parallel" method yields smaller P_i matrices, but the computation of each P_i matrix becomes more involved, since the blocks usually contain a larger number of components. Furthermore, since some of the components will generally appear in more than one block, there will be some redundancy in the required computations, and the procedure will require an additional corrective step.

2. Algebraic Method:

The algebraic method described in this section was developed by I-Ngo Chen [8].

With the algebraic method, the probability of malfunction in a component is represented with a Boolean "error" function. The method consists essentially in applying Boolean algebra to combine these error functions in order to express the next-state variables and output variables of the sequential circuit as Boolean functions of the input, present-state, and also "component" variables. The probabilistic mapping matrix can then be computed by evaluating these functions for all possible combinations of input and present-state variables.

In order to evaluate the probability of error associated with a Boolean function, a transformation must be defined from the Boolean algebra of binary variables and functions to the set of probabilities of having these functions assume logical values ZERO or ONE. Such a transformation is based on the following properties:

let x_i, $x_j \in X$, a set of n mutually independent random binary variables, and

let $p(x_i) = \text{prob}\ (x_i = 1)$ and $p(\bar{x}_i) = \text{prob}\ (x_i = 0)$,

where $0 \leq p(x_i) \leq 1$ and $p(x_i) + p(\bar{x}_i) = 1$;

then $p(x_i \cdot x_j) = p(x_i)\ p(x_j)$

$$p(x_i + x_j) = p(x_i) + p(\bar{x}_i)\ p(x_j)$$

$$= p(x_j) + p(\bar{x}_j)\ p(x_i)$$

and $p(x_i \cdot \bar{x}_i) = 0$

$p(x_i + x_i) = 1$

$p(x_i \cdot x_i) = p(x_i + x_i) = p(x_i)$

These results can be extended to Boolean functions by considering every Boolean function as an event generated from set X. The following properties are obtained:

let f_i and f_j be any two Boolean functions,

let $p(f_i)$ be the probability the f_i occurs (i.e. that $f_i = 1$),

let $p(\bar{f}_i)$ be the probability that f_i does not occur (i.e. that $f_i = 0$),

then, (i) $p(f_i \cdot f_j)$ = joint probability that both f_i and f_j occur

$= 0$, if $f_i = \bar{f}_j$

$= p(f_i)$, if $f_i = f_j$

$= p(f_i)\, p(f_j)$, if f_i and f_j are independent

(ii) $p(f_i + f_j)$ = probability that at least one of f_i and f_j occurs

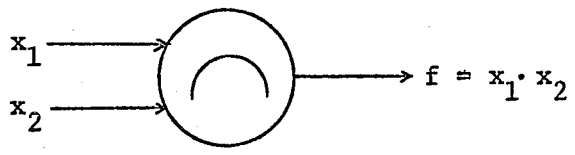
$= 1$, if $f_i = \bar{f}_j$

$= p(f_i)$, if $f_i = f_j$

$= p(f_i) + p(f_j)$, if f_i and f_j are mutually exclusive.

Let us now consider an example to show how the unreliable operation of a component can be described with a Boolean function.

EXAMPLE: consider the two-input AND gate shown below:



A binary variable, g, is associated with the AND gate, such that

$p(g)$ = probability that gate g functions properly,

$p(\bar{g})$ = probability that gate g malfunctions.

Clearly, output f can assume the logical value ONE in two (mutually exclusive) ways:

(i) $x_1 \cdot x_2 = 1$ and the gate functions properly ($g = 1$),

(ii) $x_1 \cdot x_2 = 0$ and the gate malfunctions ($g = 0$).

Thus, $p(f) = p(x_1 \cdot x_2 \cdot g) + p(\overline{x_1 \cdot x_2} \cdot \bar{g}) = p(x_1 \cdot x_2 \oplus \bar{g})$,

where $\oplus$ denotes the EXCLUSIVE-OR operation.

Similarly, $p(\bar{f}) = p(x_1 \cdot x_2 \oplus g)$.

The probability of output for the components considered in Table 3.1 are given in Table 3.3.

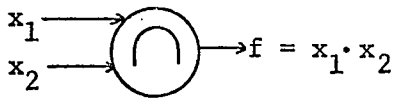
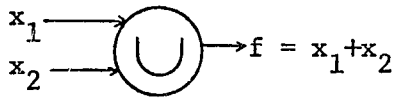
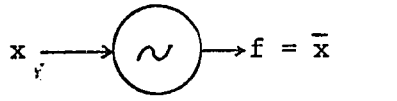
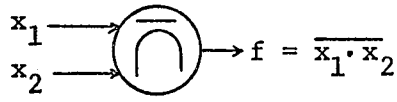
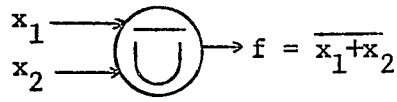
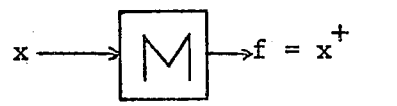
COMPONENT	OUTPUT PROBABILITIES
AND 	$p(f) = p((x_1 \cdot x_2) \oplus \bar{g}_n)$ $p(\bar{f}) = p((x_1 \cdot x_2) \oplus g_n)$
OR 	$p(f) = p((x_1 + x_2) \oplus \bar{g}_u)$ $p(\bar{f}) = p((x_1 + x_2) \oplus g_u)$
NOT 	$p(f) = p(x \oplus g_w)$ $p(\bar{f}) = p(x \oplus \bar{g}_w)$
NAND 	$p(f) = p(\overline{x_1 \cdot x_2} \oplus \bar{g}_{\bar{n}})$ $p(\bar{f}) = p(\overline{x_1 \cdot x_2} \oplus g_{\bar{n}})$
NOR 	$p(f) = p(\overline{x_1 + x_2} \oplus \bar{g}_{\bar{u}})$ $p(\bar{f}) = p(\overline{x_1 + x_2} \oplus g_{\bar{u}})$
MEMORY 	$p(f) = p(x \oplus \bar{g}_M)$ $p(\bar{f}) = p(x \oplus g_M)$

TABLE 3.3

Thus, given a sequential circuit having the general form shown in Fig. 3.1, and given the malfunction probabilities of its components, $p(\bar{g}_1)$, $p(\bar{g}_2)$, ..., $p(\bar{g}_\theta)$, the probability of output from each component can be expressed with a Boolean function, as given in Table 3.3.

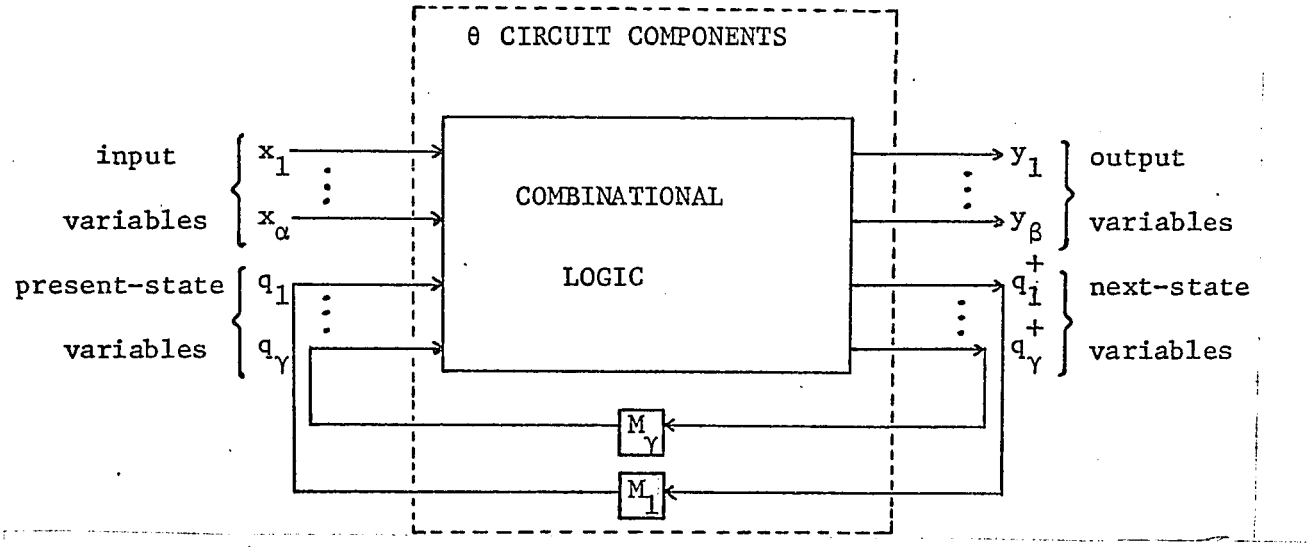


Fig. 3.1

Applying Boolean algebra to combine these functions according to the configuration of the circuit, each next-state variable, q_j^+, and each output variable, y_k, can be expressed as a Boolean function of the input, present-state, and component variables,

i.e. $q_j^+ = F_j(X, Q, G)$, with $j = 1, 2, \ldots, \gamma$,

$y_k = Z_k(X, Q, G)$, with $k = 1, 2, \ldots, \beta$,

where $X = \{x_1, x_2, \ldots, x_\alpha\}$, $Q = \{q_1, q_2, \ldots, q_\gamma\}$, $G = \{g_1, g_2, \ldots, g_\theta\}$.

In order to obtain the entries of the probabilistic mapping matrix, the joint probabilities of all next-state variables and all output variables must be considered. The joint probabilities correspond to the $2^{\beta+\gamma}$ "minterms" of all next-state and output variables, where each minterm is expressed as a Boolean function, using functions F_j ($j = 1, 2, \ldots, \gamma$) and Z_k ($k = 1, 2, \ldots, \beta$). Each of these "minterm" functions must then be evaluated for all possible $2^{\alpha+\gamma}$ combinations of input and present-state variables. With respect to the

probabilistic mapping matrix, the number of "minterm" functions corresponds to the number of columns in the matrix, and the number of times that each function must be evaluated corresponds to the number of rows of the matrix. Each function evaluation yields a sum of products of component variables, and the corresponding matrix entry is computed, using the given malfunction probabilities.

Note: In the case of input-dependent malfunction probabilities, an additional step is required in computing a probabilistic entry from the sum of products of component variables, since $p(\bar{g})$ will vary for different values of the inputs to component g. The reader is referred to Chen's paper [8] for details on this additional step.

Advantages and Drawbacks of the Algebraic Method

By comparison with the matrix method, the algebraic method reduces to some extent the amount of work required in obtaining the probabilistic mapping matrix, because it allows for simplifications through the use of Boolean algebra. However, the procedure cannot be computerized as easily as in the case of the matrix method, and it requires an additional step in dealing with input-dependent malfunction probabilities.

Furthermore, as the sequential circuit increases in size (and in the number of input, output, and state variables), the algebraic method requires the computation of many Boolean functions, where each function must be evaluated for a large number of input-state values. These Boolean functions can be simplified algebraically, but this simplification becomes fairly involved as the function becomes more complex. Thus, the need to compute and evaluate a large number of "minterm" functions becomes a serious drawback of the algebraic method.

3. Summary

In conclusion to the above survey, it is clear that existing methods for computing the probabilistic mapping matrix are relatively inefficient. On the one

hand, the matrix method offers simple but brute-force procedures, and on the other hand, the algebraic method requires the computation and evaluation of a large number of Boolean functions.

In [8] and in [11], the authors illustrate the use of the above methods with respect to very simple circuits. However, as the number of lines going in and out of the circuit increases, the matrix method involves the manipulation of very large matrices, and as the circuit increases in complexity and in the number of components, the algebraic method requires the computation of very complex functions. The inadequacy of these methods in dealing with larger circuits becomes evident with the example given in the Appendix.

Thus, there is a definite need for a different approach to the problem. For a more efficient method, the procedures should remain simple, but the amount of computations required should increase in a reasonable proportion with the size and complexity of the sequential circuit.

CHAPTER IV

THE ERROR PROPAGATION METHOD

1. Introduction

The survey presented in Chapter III clearly indicated the shortcoming of existing methods for computing the probabilistic mapping matrix as the given sequential circuit increases in size and complexity. In this chapter, a new approach to the problem is investigated, and a more efficient procedure is thereby developed.

The "error propagation" method described in this chapter is based on the concept of path sensitizing, introduced by Eldred [14] for the detection of faults in combinational circuits. The method consists essentially in representing the given sequential circuit with an "error propagation graph", thereby establishing the conditions for the propagation of component malfunctions to the output and next-state variables of the circuit. Using Boolean algebra, the "error" sets of malfunctions which propagate to these variables are determined for all values of input and present-state variables. Finally, the entries of the probabilistic mapping matrix are computed, using set theory to obtain the joint probabilities associated with the "error" sets.

The method of analysis developed in this chapter applies to asynchronous as well as to synchronous sequential circuits. In the case of a synchronous circuit, the final probabilistic mapping matrix is computed with the procedure. In the case of an asynchronous circuit, the results obtained with the error propagation procedure constitute a first step towards obtaining the reduced probabilistic matrix discussed in Chapter II.

2. Basic Assumptions and Definitions:

In order to define the problem more clearly, the following assumptions are made:

4.2.1 - each component of the sequential circuit has a non-zero probability of malfunctioning, and each malfunction probability is known. Note that memory elements are also considered as components.

4.2.2 - component malfunction probabilities are time independent.

4.2.3 - component malfunctions are statistically independent.

4.2.4 - errors occurring in the circuit are transient (temporary).

4.2.5 - second and higher order products of malfunction probabilities are neglected. This is justified by the relatively small magnitude of the given malfunction probabilities.

Note that assumption 4.2.5 is equivalent to assuming that at most one component may malfunction during any single (state transition) cycle of circuit operation.

Definition of "malfunction" variables:

In addition to the input, output, and state variables associated with a given sequential circuit, "malfunction" variables are introduced in order to represent the probability of malfunction in the circuit.

With present-day technology, sequential circuits, and computer circuits in particular, are binary in operation, i.e. the input, output, and state variables describing the circuits are all binary. Hence, the occurrence of an error at the output of any component of a circuit is defined as follows: either the component outputs a (binary) ONE when it should logically output a (binary) ZERO, or vice versa. Thus, the occurrence of an error at the output of a component (labelled g) is represented with an "error" variable, e_g, defined as:

Definition 4.1: $e_g = \begin{cases} 1, & \text{if the output of component g is in error} \\ 0, & \text{if the output of component g is correct.} \end{cases}$

Furthermore, note that an error e_g, as defined above, is due either to a malfunction of component g or to some other component malfunction which has propagated to the output of component g. Therefore, the occurrence of a malfunction in a component g is represented with a "component" variable g, defined as

Definition 4.2: $g = \begin{cases} 1, & \text{if component g functions properly} \\ 0, & \text{if component g malfunctions.} \end{cases}$

Note: whether g should be considered as a component label or as a component variable will always be clear from context.

Remark 4.2.1: variables e_g and g can be considered as events whereby $p(e_g)$ refers to the probability that an error occurs at the output of component g, $p(\bar{e}_g)$ refers to the probability that no error occurs at the output of component g, $p(g)$ refers to the probability that component g functions properly, and $p(\bar{g})$ refers to the probability that component g malfunctions,

where $p(e_g) + p(\bar{e}_g) = 1$ and $p(g) + p(\bar{g}) = 1$.

Note that event e_g is an immediate consequence of event $\bar{g}$.

Remark 4.2.2: by assumption 4.2.3, $P(\bar{g}_i\bar{g}_j) = p(\bar{g}_i)\ p(\bar{g}_j)$; therefore, by assumption 4.2.5, the probability associated with event $\bar{g}_i\bar{g}_j$ (i.e. the occurrence of simultaneous malfunctions in any two components g_i and g_j) is set to zero. However, since error e_g may propagate to other component outputs, two or more error variables may be simultaneously ONE.

3. The Concept of Error Propagation:

The error progagation method consists basically in establishing the conditions for the propagation of component malfunctions to the output and next-state variables of the given circuit. In this section, the concept of error propagation is introduced at the component level, by way of an example.

Consider the two-input AND gate, shown in Fig. 4.1, as being part of a given sequential circuit. The two inputs are represented as Boolean functions,

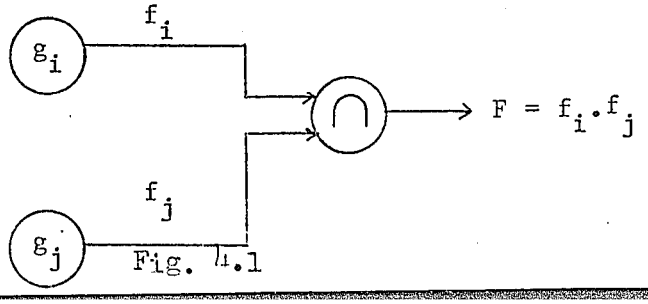


Fig. 4.1

f_i and f_j, corresponding to the outputs from some components g_i and g_j, respectively.

An error in output F, denoted by e_o , can occur in anyone of the following three ways:

(1) the AND gate malfunctions: $e_o^{(1)} = \bar{g}_o$.

(2) an error in one (and only one) of the inputs, f_i and f_j, propagates to output F:

in this case, it is assumed that the AND gate functions properly (re assertion 4.1) and that $e_{g_i}\, e_{g_j} = 0$, i.e. at most one of the inputs may be in error.

Assume that f_j is in error, i.e. $e_{g_j} = 1$:

by assumption $e_{g_i}\, e_{g_j} = 0$, $e_{g_j} = 1 \Rightarrow e_{g_i} = 0$, and f_i corresponds to the (Boolean) function defined by the error-free logic of the circuit at the output of component g_i. The following results are thereby obtained:

$f_i = 0 \Rightarrow F = 0$, regardless of the value of f_j, but

$f_i = 1 \Rightarrow F = f_j$.

Therefore, an error in f_j, e_{g_j}, is propagated to F iff $f_i = 1$.

Similarly, an error in f_i, e_{g_i}, is propagated to F iff $f_j = 1$.

Thus, $e_o^{(2)} = f_j\, e_{g_i}\, \bar{e}_{g_j} + f_i\, \bar{e}_{g_i}\, e_{g_j}$.

Remark 4.3.1: $f_j\, e_{g_i}\, \bar{e}_{g_j}$ and $f_i\, \bar{e}_{g_i}\, e_{g_j}$ represent mutually exclusive events.

(3) errors in both inputs, f_i and f_j, propagate to output F:

in this case, it is assumed that the AND gate functions properly (re assertion 4.1) and that $e_{g_i}\, e_{g_j} = 1$. The following results are obtained:

if $f_i \oplus f_j = 1$, then $e_{g_i}\, e_{g_j} = 1 \Rightarrow$ no error in F,

if $f_i \oplus f_j = 0$, then $e_{g_i}\, e_{g_j} = 1 \Rightarrow$ F is in error,

where $\oplus$ denotes exclusive - OR.

Thus, $e_o^{(3)} = \overline{(f_i \oplus f_j)}\, e_{g_i}\, e_{g_j}$.

Remark 4.3.2: since, in the above analysis, f_i and f_j correspond to error-free logic functions, then $p(f_i) = \begin{cases} 1, & \text{if } f_i = 1 \\ 0, & \text{if } f_i = 0 \end{cases}$ and $p(f_j) = \begin{cases} 1, & \text{if } f_j = 1 \\ 0, & \text{if } f_j = 0 \end{cases}$

Similarly, $p(\overline{f_i \oplus f_j}) = \begin{cases} 1, & \text{if } \overline{f_i \oplus f_j} = 1 \\ 0, & \text{if } \overline{f_i \oplus f_j} = 0 \end{cases}$

Remark 4.3.3: the above analysis also holds in cases where f_i and/or f_j corespond to input or present-state variables of the sequential circuit. These cases are considered in more detail in Remarks 4.4.2 and 4.4.3.

Assertion 4.1: $e_o^{(1)}$, $e_o^{(2)}$, $e_o^{(3)}$ represent mutually exclusive events. Hence,

$$p(e_o) = p(\bar{g}_o) + f_i\, p(\bar{e}_{g_i}\, e_{g_j}) + f_j\, p(e_{g_i}\, \bar{e}_{g_j}) + \overline{f_i \oplus f_j}\, p(e_{g_i}\, e_{g_j}) \quad \ldots\ldots\ldots (4.1)$$

proof: consider $e_o^{(1)}$, $e_o^{(2)}$, $e_o^{(3)}$ as occurrences of an error at the output of the AND gate (as outlined in Remark 4.2.1), corresponding to the above cases (1), (2), and (3), respectively.

Then, $e_o^{(1)} = 1 \Rightarrow \bar{g}_o = 1 \Rightarrow e_{g_i} = 0$ and $e_{g_j} = 0$ (by assumption 4.2.5)

$$\Rightarrow e_o^{(2)} = e_o^{(3)} = 0.$$

Similarly, $e_o^{(2)} = 1 \Rightarrow e_{g_i} \oplus e_{g_j} = 1 \Rightarrow e_{g_i}\, e_{g_j} = 0$ and $\bar{g}_o = 0$ (by assumption 4.2.5)

$$\Rightarrow e_o^{(1)} = e_o^{(3)} = 0.$$

Finally, $e_o^{(3)} = 1 \Rightarrow e_{g_i}\, e_{g_j} = 1$

$$\Rightarrow e_{g_i}\, \bar{e}_{g_j} = \bar{e}_{g_i}\, e_{g_j} = 0 \text{ and } \bar{g}_o = 0 \text{ (by assumption 4.2.5)}$$

$$\Rightarrow e_o^{(1)} = e_o^{(2)} = 0.$$

Thus, $e_o^{(1)}$, $e_o^{(2)}$, $e_o^{(3)}$ represent mutually exclusive events. Therefore,

$$\begin{aligned} p(e_o) &= p(e_o^{(1)}) + p(e_o^{(2)}) + p(e_o^{(3)}) \\ &= p(\bar{g}_o) + p(f_i\, \bar{e}_{g_i}\, e_{g_j} + f_j\, e_{g_i}\, \bar{e}_{g_j}) + p(\overline{f_i \oplus f_j}\, e_{g_i}\, e_{g_j}) \\ &= p(\bar{g}_o) + p(f_i\, \bar{e}_{g_i}\, e_{g_j}) + p(f_j\, e_{g_i}\, \bar{e}_{g_j}) + p(\overline{f_i \oplus f_j}\, \underline{e_{g_i}\, e_{g_j}}) \text{ (by Remark 4.3.1)} \\ &= p(\bar{g}_o) + p(f_i)\, p(\bar{e}_{g_i}\, e_{g_j}) + p(f_j)\, p(e_{g_i}\, \bar{e}_{g_j}) + p(\overline{f_i \oplus f_j})\, p(e_{g_i}\, e_{g_j}) \\ &= p(\bar{g}_o) + f_i\, p(\bar{e}_{g_i}\, e_{g_j}) + f_j\, p(e_{g_i}\, \bar{e}_{g_j}) + \overline{f_i \oplus f_j}\, p(e_{g_i}\, e_{g_j}) \text{ (by Remark 4.3.2)} \end{aligned}$$

Q.E.D.

Assertion 4.2: If at most one input to the AND gate can be in error at any time instant (i.e. $e_{g_i}\, e_{g_j} = 0$), then $p(e_o) = P(\bar{g}_o) + f_i\, p(e_{g_i}) + f_j\, p(e_{g_i}) \ldots (4.2)$

proof: by the assumption $e_{g_i}\, e_{g_j} = 0$, the event corresponding to $e_{g_i}\, e_{g_j} = 1$ cannot occur, i.e. $p(e_{g_i}\, e_{g_j}) = 0$. Furthermore, if $e_{g_i}\, e_{g_j} = 0$, then

$e_{g_i}\, \bar{e}_{g_j} = 1 \Leftrightarrow e_{g_i} = 1$ and $\bar{e}_{g_i}\, e_{g_j} = 1 \Leftrightarrow e_{g_j} = 1$.

Therefore, $p(e_{g_i} \bar{e}_{g_j}) = p(e_{g_i})$ and $p(\bar{e}_{g_i} e_{g_j}) = p(e_{g_j})$.
Hence, equation (4.1) is reduced to equation (4.2) Q.E.D.

By comparison with equation (4.1), equation (4.2) provides a simpler expression for the probability of error at the output of an AND gate. As will be shown in the next section, equation (4.2) leads to a useful graphical representation of error propagation (for an AND gate). Similar results can be obtained for other types of components, as shown in Table 4.1, but these representations also fail to account for occurrences of errors of the type $e_n^{(3)}$. On the other hand, by assumption 4.2.5, only a single component may malfunction at any time instant, and only for circuits with reconvergent fanout can a single component malfunction (at the fanout point) propagate to both inputs of another component (at the reconvergent point only). Hence, errors of the type $e_n^{(3)}$ will occur for a relatively small number of components.

Thus, the procedure for obtaining the probabilistic mapping matrix will be developed with respect to the basic results given in Table 4.1, and, as will be shown in a later section, occurrence of errors of the type $e_n^{(3)}$ will be easily detected and corrected through a simple modification.

4. The Error Propagation Graph:

As outlined in the previous section, the concept of error propagation can be represented with a simple "error propagation" graph, which will be used, in the next section, as the basis of the procedure for computing the probabilistic mapping matrix.

Equation (4.2) obtained for the AND gate considered in the previous section (Fig. 4.1) can be represented as follows:

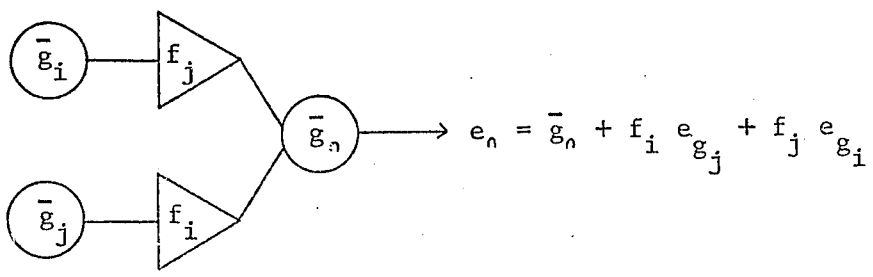
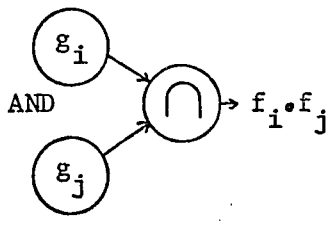
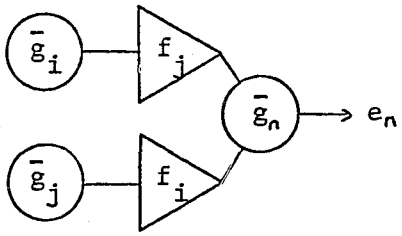
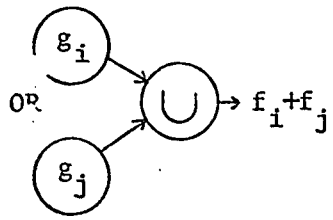
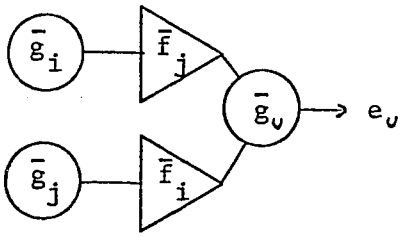
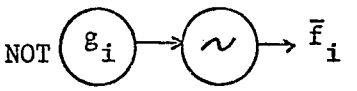
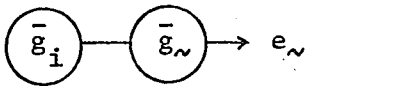
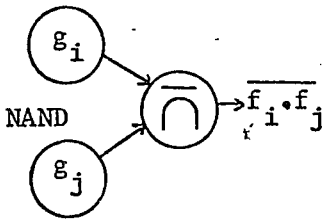
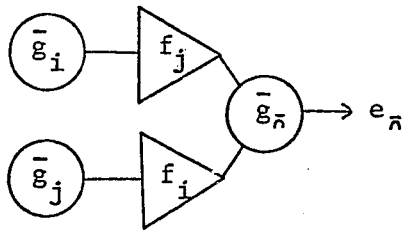
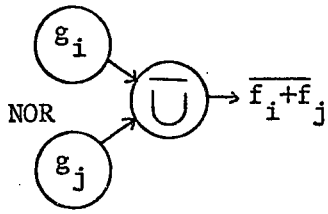
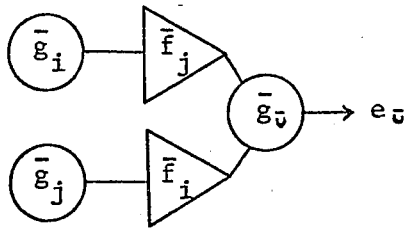
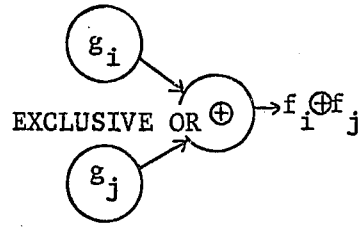
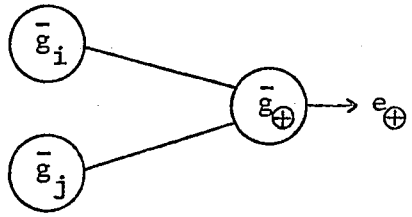
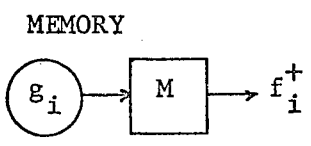
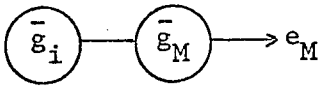
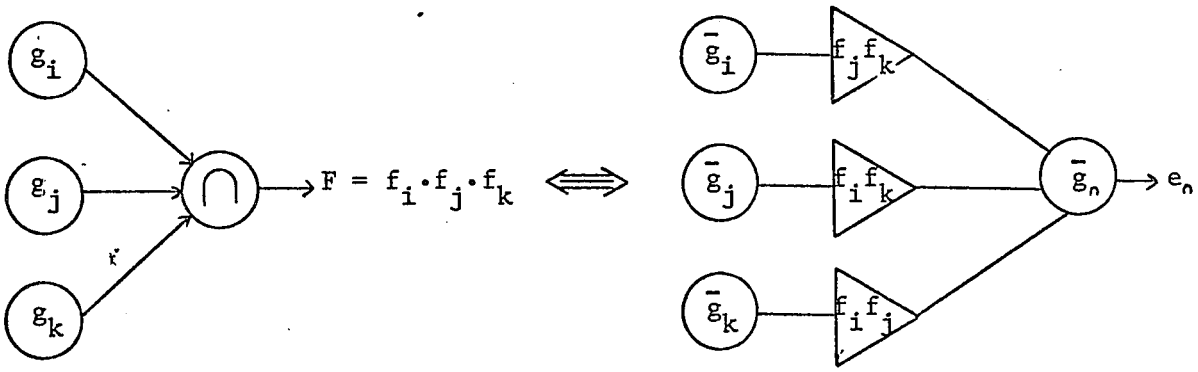


Table 4.1

COMPONENT	OUTPUT ERROR PROBABILITY	ERROR PROPAGATION GRAPH
	$p(e_{\cap}) = p(\bar{g}_{\cap}) + f_i p(e_{g_j}) + f_j p(e_{g_i})$	
	$p(e_{\cup}) = p(\bar{g}_{\cup}) + \bar{f}_i p(e_{g_j}) + \bar{f}_j p(e_{g_i})$	
	$p(e_{\sim}) = p(\bar{g}_{\sim}) + p(e_{g_i})$	
	$p(e_{\bar{\cap}}) = p(\bar{g}_{\bar{\cap}}) + f_i p(e_{g_j}) + f_j p(e_{g_i})$	
	$p(e_{\bar{\cup}}) = p(\bar{g}_{\bar{\cup}}) + \bar{f}_i p(e_{g_j}) + \bar{f}_j p(e_{g_i})$	
	$p(e_{\oplus}) = p(\bar{g}_{\oplus}) + p(e_{g_i}) + p(e_{g_j})$	
	$p(e_M) = p(\bar{g}_M) + p(e_{g_i})$	

The circular nodes of the graph correspond to component malfunctions, and the edges represent paths along which errors can be propagated. The triangular nodes (f_i and f_j) are referred to as "propagation functions", as they correspond to error-free logic functions whose value must be ONE in order for errors to propagate along the paths which they "govern".

Similarly, output error probabilities and error propagation graphs are obtained for other types of components, as shown in Table 4.1. For simplicity, only one and 2-input components are considered, but error propagation graphs can also be obtained for multiple-input components, as illustrated with the 3-input AND gate shown below:



The concept of error propagation can now be easily extended to sequential circuits, using the graphs obtained at the component level. Given a sequential circuit of the form shown in Fig. 4.2, an error propagation graph

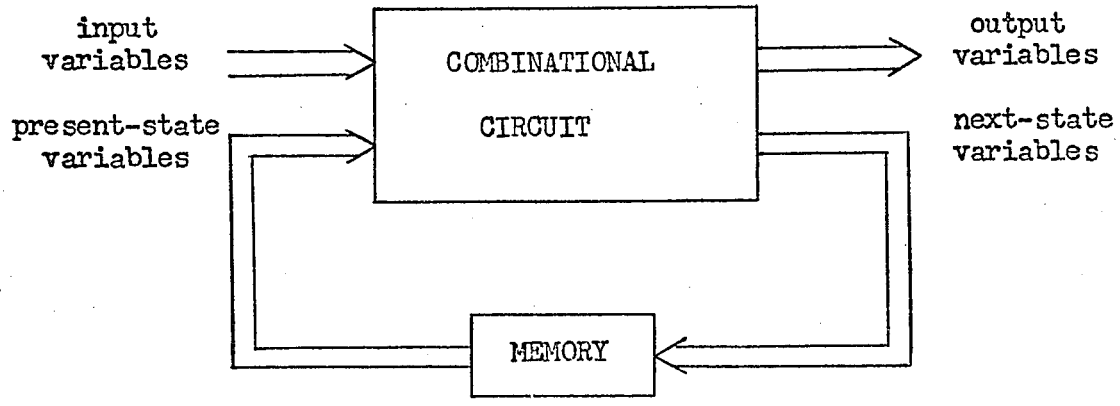


Fig. 4.2

is drawn for each output variable and each next-state variable by simple inspection of the circuit configuration. In each case, the graph is obtained by tracing the variable back to the input and present-state variables, replacing each component encountered with its corresponding error propagation graph. This process is illustrated with the following example.

Example: Consider the binary counter shown in Fig. 4.3.

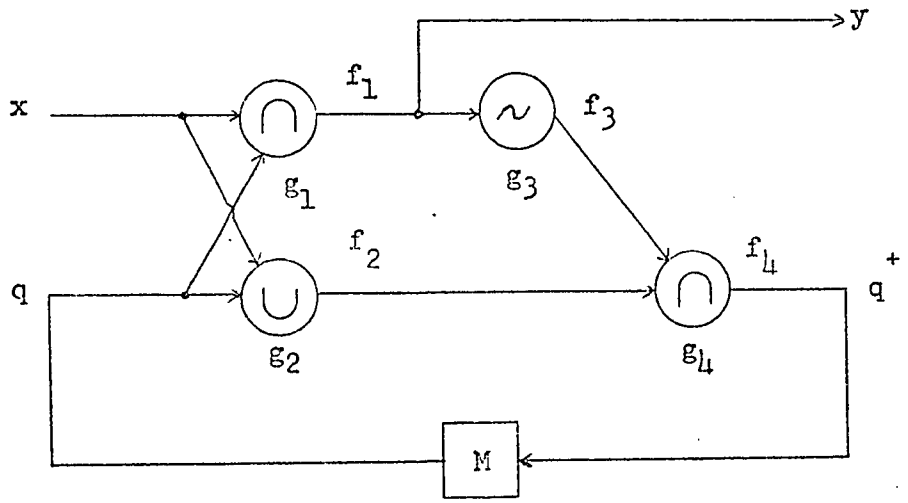
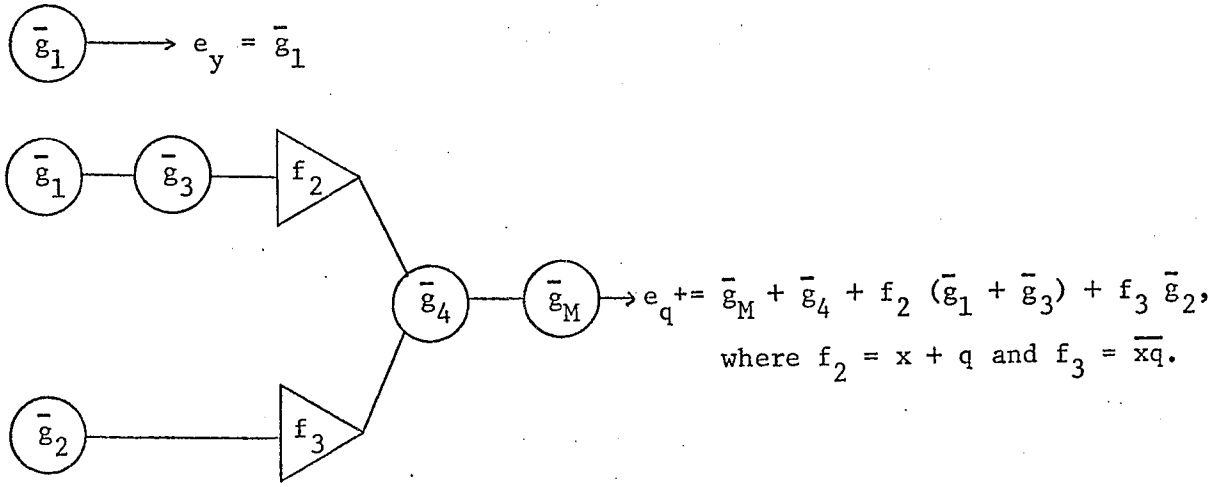


Fig. 4.3

Each component is labelled as shown above. The following error propagation graphs are obtained for output variable y and next-state variable q^+:



Remark 4.4.1: In the case of a next-state variable, the corresponding memory element is always included in the graph (usually as the right-most node), since it also has a non-zero probability of malfunctioning thereby contributing to the probability that the next-state may be in error. Note that the memory element can be any device capable of storing information (delay line, flip-flop, etc.)

Remark 4.4.2: Within the present context, input and present-state variables are assumed errorless (as in the above example), since the probabilistic mapping matrix is defined in terms of the probability $p(s_j, v_r/s_i, u_p)$, whereby it is assumed that the sequential circuit is in a given state s_i, and that input u_p is applied. However, one must consider the possibility that the input variables and/or present-state variables may, in fact, be probabilistic.

In the case of present-state variables, probabilistic initial values are expressed with a stochastic initial-state vector $\Pi(\lambda)$, as defined in Chapter II (Section 1). For the subsequent operation of the circuit, occurrences of errors in the present-state variables can result only from the feedback of errors in the next-state variables, which is accounted for by matrix multiplication, as discussed in Chapter II (Section 3).

In the case of input variables, the problem is considered under two different aspects. First, if the input is probabilistic in itself, then each input variable is assumed errorless in obtaining the probabilistic mapping matrix, and the probabilistic nature of the input is taken into account in the computations required for estimating the reliability of the system [12]. On the other hand, if the probability that an input variable x is in error arises from the fact that x corresponds to an output variable from some

other circuit, then the probability of error in x, $p(e_x)$, is expressed as an additional node, e_x, in the error propagation graphs of the circuit. In that case, propagation of error e_x to the output and next-state variables of the circuit is governed by the propagation functions of the corresponding graphs.

Remark 4.4.3: Any disturbance in a connection from a primary input variable or state variable to a component of a given sequential circuit is considered as a malfunction of that component.

Remark 4.4.4: Every error propagation graph displays a tree structure.

As seen from the above example, the error propagation graph provides much insight into the reliability problem by displaying how component malfunctions can propagate as errors in the output and next-state variables of the circuit. For simple circuits such as the one considered above, error probabilities can be computed by inspection of the graphs and evaluation of the propagation functions. However, as the circuit increases in size and complexity, graphical inspection alone becomes difficult, and a more formal procedure is therefore needed.

5. The Error Propagation Procedure:

In order to illustrate each step more clearly, the procedure is developed with respect to an example. For the purpose of comparison with other methods in the concluding section of this chapter, the example corresponds to the one given in the Appendix, where it is used to illustrate the matrix and algebraic methods in computing the probabilistic mapping matrix.

5.1. Obtaining the Error Propagation Graphs:

As pointed out in the previous sections, the basic information required to compute the probabilistic mapping matrix is obtained with the error propagation graph. Hence, the first step of the procedure is to obtain a graph

for each output variable and each next-state variable of the given circuit. This is easily done by inspection, as described in the preceding section. For our example, the error propagation graphs are shown in Fig. 4.4.

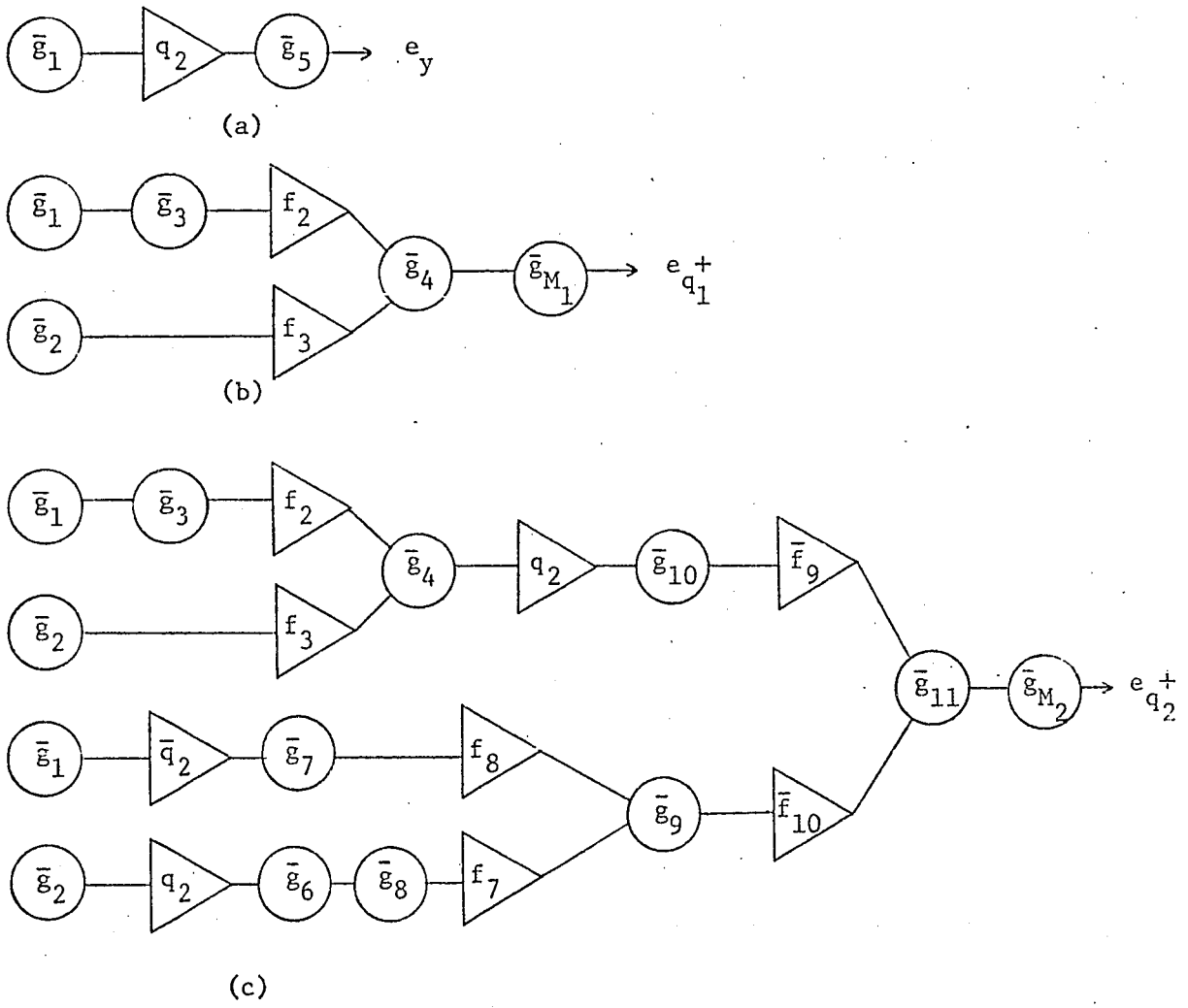


Fig. 4.4

At this point, the error propagation graphs display the component malfunctions that can propagate as errors in each output or next-state variable, and they specify the conditions under which propagation can occur. Furthermore, cases of reconvergent fanout are readily detected by inspection of the graphs;

for example, from Fig. 4.4 (c) it is clear that each one of component malfunctions $\bar{g}_1, \bar{g}_2$ can propagate to q_2^+ through two different paths.

5.2. Formulation:

The next step is to obtain a convenient formulation of the information provided by the error propagation graphs. By considering the tree structure of the graphs, two general matrix formats are suggested, and their advantages and disadvantages are pointed out.

F - formulation: In achieving this formulation, it is useful to consider each graph as a tree in which the error variable (associated with the corresponding output or next-state variable) is the root, the component malfunctions are the nodes, the propagation paths are the branches, and the propagation functions (associated with triangular nodes) correspond to cut-sets.

In going from the root to the leaves of the tree, the first node encountered is always directly connected to the root. Physically, this means that the corresponding component malfunction propagates directly to the output or next-state variable, independently of any propagation function, since the variable corresponds to the output of that component. However, note that, for some graphs, more than one malfunction may propagate directly to the root of the tree, as in the case of Fig. 4.4 (b) and Fig. 4.4 (c).

In going deeper into the tree structure, branches connecting some of the nodes to the root will each contain a single propagation function. Physically, the propagation of the corresponding component malfunctions is governed by each of these functions alone. For example, from Fig. 4.4, functions q_2 (Fig. 4.4 (a)), f_2 and f_3 (Fig. 4.4 (b)), and $\bar{f}_9$ and $\bar{f}_{10}$ (Fig. 4.4 (c)) must be considered.

In going still further along the branches of the tree, propagation of the remaining component malfunctions to the root will depend on two or more propagation functions. Hence, the corresponding (Boolean) products of these functions must be considered. For example, from Fig. 4.4 (c), functions

$q_2\bar{f}_9$, $f_8\bar{f}_{10}$, $f_7\bar{f}_{10}$, $f_2q_2\bar{f}_9$, $\bar{q}_2f_8\bar{f}_{10}$, and $q_2f_7\bar{f}_{10}$ must be computed.

Proceeding as described above, one can obtain all propagation functions and products of propagation functions needed to determine the component malfunctions that propagate to the output and next-state variables. These results can be formulated with the following matrix equation:

$$E = G_F P_F + G_D \tag{4.3}$$

where $E = \left[e_1\ e_2\ \cdots\ e_\gamma\ e_{\gamma+1}\ \cdots\ e_{\gamma+\beta}\right]^T \equiv \left[e_{q_1}^+\ \cdots\ e_{q_\gamma}^+\ e_{y_1}\ \cdots\ e_{y_\beta}\right]^T$ is a $(\gamma+\beta)$ - dimensional column vector of error variables corresponding to the output and next-state variables of the circuit (re Fig. 3.1),

P_F is a column vector of propagation functions and products of propagation functions, as discussed above,

G_F is a matrix whose $(i, j)^{th}$ entry corresponds to the (sum of) component malfunctions which can propagate as an error e_i, depending on the j^{th} element of vector P_F,

and G_D is a $(\gamma+\beta)$ - dimensional column vector whose i^{th} entry corresponds to the (sum of) component malfunctions which can propagate directly as an error e_i.

Note that if the j^{th} entry of vector P_F does not govern the propagation of any component malfunction to the i^{th} output or next-state variable, then the $(i, j)^{th}$ entry of matrix G_F is zero. Also note that every entry of vector G_D is non-zero. Furthermore, a case of reconvergent fanout originating at the output of a component g_i will appear as the occurrence of malfunction $\bar{g}_i$ in more than one entry of a same row of matrix G_F.

The resulting matrix equation for our example is shown in Fig. 4.5.

The F - formulation will be advantageous with sequential circuits that have a small number of component levels. Otherwise, the corresponding error propagation graphs will display long propagation paths involving many

$$
\begin{bmatrix} e_{q_1}^{+} \\ e_{q_2}^{+} \\ e_y \end{bmatrix} = \begin{bmatrix} 0 & \bar{g}_1+\bar{g}_3 & \bar{g}_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \bar{g}_{10} & \bar{g}_9 & \bar{g}_4 & \bar{g}_7 & \bar{g}_6+\bar{g}_8 & \bar{g}_1+\bar{g}_3 & \bar{g}_2 & \bar{g}_1 & \bar{g}_2 \\ \bar{g}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} q_2 \\ f_2 \\ f_3 \\ \bar{f}_9 \\ \bar{f}_{10} \\ \bar{f}_9 q_2 \\ \bar{f}_{10} f_8 \\ \bar{f}_{10} f_7 \\ \bar{f}_9 q_2 f_2 \\ \bar{f}_9 q_2 f_3 \\ \bar{f}_{10} f_8 \bar{q}_2 \\ \bar{f}_{10} f_7 q_2 \end{bmatrix} + \begin{bmatrix} \bar{g}_4+\bar{g}_{M_1} \\ \bar{g}_{11}+\bar{g}_{M_2} \\ \bar{g}_5 \end{bmatrix}
$$

Fig. 4.5

propagation functions, in which case a large number of function products will need to be considered. In this latter case, the next formulation will prove more appropriate.

M - formulation: In this case, the information is formulated with respect to each component malfunction of the circuit. By inspection of each error propagation graph, the path through which a component malfunction can propagate (to the corresponding output or next-state variable) is expressed as a product of the propagation functions encountered along the path; the resulting function is referred to as a path-function. In the case where a component malfunction can propagate to an output or next-state variable through two or more paths (i.e. reconvergent fanout case), the corresponding path-function involves a sum of products of propagation functions. Hence, the M - formulation is given by the following matrix equation:

$$E = F_M G_M \tag{4.4}$$

where E is a $(\gamma+\beta)$ - dimensional column vector, as defined with the F - formulation,

G_M is a θ - dimensional column vector whose j^{th} entry corresponds to the j^{th} component malfunction, where θ is the number of components in the circuit,

and F_M is a $((\gamma+\beta) \times \theta)$ matrix whose $(i, j)^{th}$ element, F_{ij}, is the path-function which governs the propagation of the j^{th} component malfunction to the i^{th} output or next-state variable. Note that $F_{ij} = 0$ if the j^{th} malfunction cannot propagate to the i^{th} variable, and that $F_{ij} = 1$ if the j^{th} malfunction propagates directly to the i^{th} variable.

For our example, the resulting matrix equation is shown in Fig. 4.6.

Remark 4.5.1: in dealing with cases of reconvergent fanout, the corresponding sum-of-product expressions obtained with the M - formulation immediately indicate the propagation functions that must be ONE in order that more than one input to the component at the point of reconvergence be in error.

$$
\begin{bmatrix} e_{q_1}^{+} \\ e_{q_2}^{+} \\ e_y \end{bmatrix} = \begin{bmatrix} f_2 & f_3 & f_2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ f_2 q_2 \bar{f}_9 + \bar{q}_2 f_8 \bar{f}_{10} & f_3 q_2 \bar{f}_9 + q_2 f_7 \bar{f}_{10} & f_2 q_2 \bar{f}_9 & q_2 \bar{f}_9 & 0 & f_7 \bar{f}_{10} & f_8 \bar{f}_{10} & f_7 \bar{f}_{10} & \bar{f}_{10} & \bar{f}_9 & 1 & 0 & 1 \\ q_2 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{g}_1 \\ \bar{g}_2 \\ \bar{g}_3 \\ \bar{g}_4 \\ \bar{g}_5 \\ \bar{g}_6 \\ \bar{g}_7 \\ \bar{g}_8 \\ \bar{g}_9 \\ \bar{g}_{10} \\ \bar{g}_{11} \\ \bar{g}_{M_1} \\ \bar{g}_{M_2} \end{bmatrix}
$$

Fig. 4.6

By comparison with the F - formulation, the M - formulation will generally prove more convenient as the number of component levels in the given sequential circuit increases. Furthermore, as outlined in Remark 4.5.1, the M - formulation is also more convenient in dealing with cases of reconvergent fanout. Hence, the remainder of this section will be developed with respect to the M - formulation.

5.3. Modification in the case of reconvergent fanout:

As outlined in Section 3 of this chapter, the error propagation graph is developed under the assumption that at most one input to each component may be in error at any time instant. However, this assumption is not valid in the case of reconvergent fanout, and a modification must be made with respect to path-functions associated with component malfunctions that can propagate to the fanout point.

Let us consider the general case of reconvergent fanout where the point of reconvergence corresponds to a two-input component, as shown in Fig. 4.7. Fanout occurs at the output of component g_f, and error e_{g_f} can be propagated as

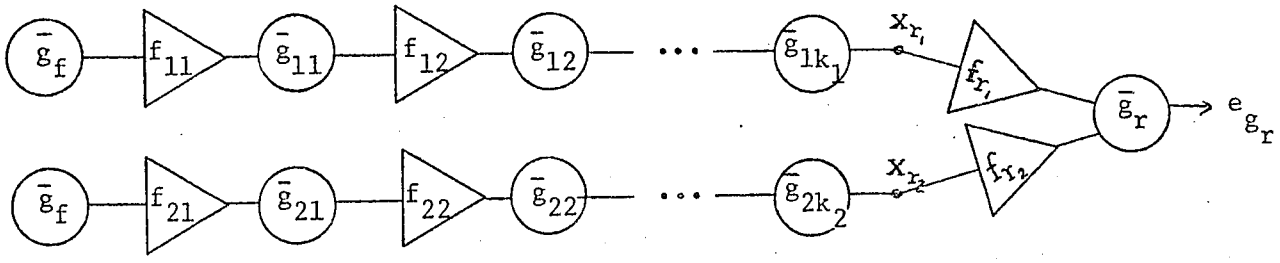


Fig. 4.7

an error e_{g_r} (at the point of reconvergence) through two paths, p_1 and p_2. Propagation function f_{ij} ($j = 1, 2, \ldots, k_i$) governs the propagation of an error from the input to the output of component g_{ij}, encountered on path

p_i (i = 1, 2). Similarly, functions f_{r_1} and f_{r_2} govern the propagation of an error at inputs x_{r_1} and x_{r_2}, respectively to the output of component g_r, but only under the assumption that at most one input may be in error.

A necessary condition for both inputs, x_{r_1} and x_{r_2}, to be in error is that the Boolean product $F_{p_1} F_{p_2} = 1$, where $F_{p_1} = \prod_{j_1=1}^{k_1} f_{1j_1}$ and $F_{p_2} = \prod_{j_2=1}^{k_2} f_{2j_2}$. However, the propagation of an error in both x_{r_1} and x_{r_2} to the output of component g_r depends on the values of x_{r_1} and x_{r_2} and on the type of component corresponding to g_r.

In the case where component g_r is of the type AND, OR, NAND, or NOR, an error in both inputs to g_r is propagated to the output of g_r (i.e. e_{g_r} =1) if and only if $x_{r_1} \oplus x_{r_2} = 0$. Hence, the path-function associated with malfunction $\bar{g}_f$ (or any other component malfunction which can propagate to the fanout point) will contain F_{rf} as a factor, where

$$F_{rf} = F_{p_1} f_{r_1} \oplus F_{p_2} f_{r_2} \oplus F_{p_1} F_{p_2},$$

where $\begin{cases} f_{r_1} = x_{r_2} \text{ and } f_{r_2} = x_{r_1} \text{ if } g_r \text{ is of the type AND or NAND,} \\ f_{r_1} = \bar{x}_{r_2} \text{ and } f_{r_2} = \bar{x}_{r_1} \text{ if } g_r \text{ is of the type OR or NOR.} \end{cases}$

On the other hand, if g_r is of the type EXCLUSIVE-OR or EQUIVALENCE, then $e_{g_r} = 0$ whenever both inputs to g_r are in error. Hence, expression F_{rf} is given as $F_{rf} = F_{p_1} \oplus F_{p_2}$.

By a similar analysis, modified path-functions can be obtained in cases of reconvergent fanout involving more than two paths. For example, in the case of a fanout which reconverges through three paths at the input of a 3-input AND gate, as shown schematically in Fig. 4.8, the corresponding

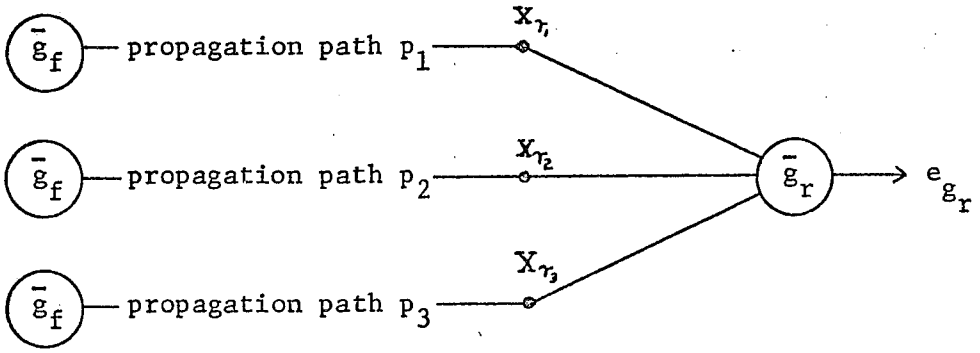


Fig. 4.8

expression F_{rf} is given as

$$F_{rf} = F_{p_1} x_{r_2} x_{r_3} \oplus F_{p_2} x_{r_1} x_{r_3} \oplus F_{p_3} x_{r_1} x_{r_2} \oplus F_{p_1} F_{p_2} x_{r_3} \oplus F_{p_1} F_{p_3} x_{r_2} \oplus F_{p_2} F_{p_3} x_{r_1} \oplus F_{p_1} F_{p_2} F_{p_3}$$

Since these cases very rarely occur, they will not be analysed to any detail.

For our example, reconvergent fanout is detected at the output of components g_1 and g_2. With respect to the M-formulation, the path-function corresponding to malfunction $\bar{g}_1$ becomes $F_{31} = f_2 q_2 \bar{f}_9 \oplus \bar{q}_2 f_8 \bar{f}_{10} \oplus f_2 q_2 \bar{q}_2 f_8$, and that corresponding to $\bar{g}_2$ becomes $F_{32} = f_3 q_2 \bar{f}_9 \oplus q_2 f_7 \bar{f}_{10} \oplus f_3 q_2 f_7$. Note that in the case of F_{31}, no modification is needed, because $f_2 q_2 \bar{q}_2 f_8 = 0$, i.e. the two inputs to component g_{11} will never be simultaneously in error.

5.4. Determining the propagated malfunctions:

In order to determine the component malfunctions that can propagate to each output variable and each next-state variable for every state-input pair, the path-functions obtained with the M-formulation (subsections 5.2 and 5.3) must be computed for all $(\gamma+\alpha)$ - tuples of present-state and input variables (re Fig. 3.1).

In formatting these results, it becomes more convenient to express separately each variable e_i ($i = 1, 2, \ldots, \gamma+\beta$) of vector E:

$$e_i = F_{Mi} G_{Mi} \tag{4.5}$$

where $F_{Mi} = \left[F_{i_1} \; F_{i_2} \; \ldots \; F_{i_{\theta_i}} \right]$ is a θ_i - dimensional row vector ($\theta_i \leq \theta$) which includes the θ_i non-zero entries (path-functions) of the i^{th} row of matrix F_M,

and G_{Mi} is a θ_i - dimensional column vector which includes the θ_i component malfunctions that can propagate to the output or next-state variable corresponding to e_i.

Next, using Boolean algebra, each path-function F_{i_k} ($k = 1,2,\ldots,\theta_i$) is expressed in terms of the input and present-state variables. In order to

obtain a more condensed formulation, all path-functions that are found to be equal are represented by a single path-function, i.e.

$$e_i = F_i G_i \tag{4.6}$$

where $F_i = \left[F_1^i \; F_2^i \; \ldots \; F_{\eta_i}^i\right]$ is a η_i - dimensional row vector ($\eta_i \leq \theta_i$) which includes the η_i different path-functions associated with variable e_i,

and G_i is a η_i - dimensional column vector whose k^{th} entry ($k = 1,2,\ldots, \eta_i$) corresponds to the sum of all component malfunctions whose propagation is governed by path-function F_k^i.

For our example, the latter formulation corresponds to the following equations:

$$e_1 = e_{q_1}+ = \left[F_1^1 \; F_2^1 \; F_3^1\right] \begin{bmatrix} \bar{g}_1 + \bar{g}_3 \\ \bar{g}_2 \\ \bar{g}_4 + \bar{g}_{M_1} \end{bmatrix}, \text{ where } F_1^1 = f_2 = x + q_1,$$
$$f_2^1 = f_3 = \overline{xq}_1,$$
$$f_3^1 = 1.$$

$$e_2 = e_{q_2}+ = \left[F_1^2 \; F_2^2 \; F_3^2 \; F_4^2 \; F_5^2 \; F_6^2 \; F_7^2 \; F_8^2\right] \begin{bmatrix} \bar{g}_1 \\ \bar{g}_2 \\ \bar{g}_3+\bar{g}_4 \\ \bar{g}_6+\bar{g}_8 \\ \bar{g}_7 \\ \bar{g}_{10} \\ \bar{g}_9 \\ \bar{g}_{11}+\bar{g}_{M_2} \end{bmatrix}$$

where $F_1^2 = f_2 q_2 \bar{f}_9 + \bar{q}_2 f_8 \bar{f}_{10} = x + q_1 + \bar{q}_2$

$F_2^2 = f_3 q_2 \bar{f}_9 \oplus q_2 f_7 \bar{f}_{10} \oplus f_3 q_2 f_7 = x q_1 q_2$

$F_3^2 = f_2 q_2 \bar{f}_9 = q_2 \bar{f}_9 = q_2(x + q_1)$

$F_4^2 = f_7 \bar{f}_{10} = xq_1 + \overline{\bar{x}\bar{q}}_1 q_2$

$$F_5^2 = f_8\bar{f}_{10} = \bar{q}_2 + \bar{x}\bar{q}_1$$

$$F_6^2 = \bar{f}_9 = q_1q_2 + \bar{x}\bar{q}_2 + x\bar{q}_1$$

$$F_7^2 = \bar{f}_{10} = \bar{q}_2 + (x \oplus \bar{q}_1)$$

$$F_8^2 = 1$$

and $e_3 = e_y = \begin{bmatrix} F_1^3 & F_2^3 \end{bmatrix} \begin{bmatrix} \bar{g}_1 \\ \bar{g}_5 \end{bmatrix}$, where $F_1^3 = q_2$

$F_2^3 = 1.$

Finally, each path-function F_k^i ($k = 1, 2, \ldots, \eta_i; i = 1, 2, \ldots, (\gamma+\beta)$) must be evaluated for every $(\gamma+\alpha)$ - tuple of present-state and input variables. This evaluation can be conveniently formatted in matrix form, which leads to the final formulation

$$E_{e_i} = F_{e_i} G_{e_i}, \quad i = 1, 2, \ldots, (\gamma+\beta), \tag{4.7}$$

where F_{e_i} is a $(2^{(\gamma+\beta)} \times \eta_i)$ matrix whose k^{th} column ($k = 1, 2, \ldots, \eta_i$) corresponds to a "truth-table" representation of path-function F_k^i,

$G_{e_i} = G_i$, as defined above,

hence E_{e_i} is a $2^{(\gamma+\alpha)}$ - dimensional column vector whose j^{th} entry corresponds to the sum of component malfunctions that can propagate to the i^{th} (output or next-state) variable when the j^{th} state-input $(\gamma+\alpha)$ - tuple is applied to the circuit.

Note that the final formulation corresponding to equation (4.7) can be obtained directly from the error propagation graphs upon expressing the path-functions (obtained in subsections 5.2 and 5.3) in terms of the input and state variables of the circuit. Formulations (4.5) and (4.6) were introduced for the purpose of explanation.

For our example, the following matrix equations are obtained:

$$E_{e_1} = E_{q_1+} =$$

q_1q_2x	F_1^1	F_2^1	F_3^1	
0 0 0	0	1	1	
0 0 1	1	1	1	
0 1 0	0	1	1	$\bar{g}_1+\bar{g}_3$
0 1 1	1	1	1	$\bar{g}_2$
1 0 0	1	1	1	$\bar{g}_4+\bar{g}_{M_1}$
1 0 1	1	0	1	
1 1 0	1	1	1	
1 1 1	1	0	1	

(4.8)

$$E_{e_2} = E_{q_2+} =$$

	F_1^2	F_2^2	F_3^2	F_4^2	F_5^2	F_6^2	F_7^2	F_8^2	
0 0 0	1	0	0	0	1	1	1	1	$\bar{g}_1$
0 0 1	1	0	0	0	1	1	1	1	$\bar{g}_2$
0 1 0	0	0	0	1	1	0	1	1	$\bar{g}_3+\bar{g}_4$
0 1 1	1	0	1	0	0	1	0	1	$\bar{g}_6+\bar{g}_8$
1 0 0	1	0	0	0	1	1	1	1	$\bar{g}_7$
1 0 1	1	0	0	1	1	0	1	1	$\bar{g}_{10}$
1 1 0	1	0	1	0	0	1	0	1	$\bar{g}_9$
1 1 1	1	1	1	1	0	1	1	1	$\bar{g}_{11}+\bar{g}_{M_2}$

(4.9)

$$E_{e_3} = E_y =$$

	F_1^3	F_2^3	
0 0 0	0	1	
0 0 1	0	1	
0 1 0	1	1	
0 1 1	1	1	$\bar{g}_1$
1 0 0	0	1	$\bar{g}_5$
1 0 1	0	1	
1 1 0	1	1	
1 1 1	1	1	

(4.10)

Thus, using formulation (4.7), one can determine, for each state-input tuple, the component malfunctions that can propagate to each output variable y_δ ($\delta = 1, 2, \ldots, \beta$) and each next-state variable q^+_ω ($\omega = 1, 2, \ldots, \gamma$).

5.5. Computing the entries of the probabilistic mapping matrix:

Having determined the propagated malfunctions, we can now compute the entries of the probabilistic mapping matrix by obtaining the joint probabilities of all output and next-state variables. This final step can be efficiently achieved by representing the propagated malfunctions as "error" sets. This subsection is therefore divided into two parts: first, the error sets are defined, and their properties are investigated; second, these general results are then applied in computing the entries of the probabilistic mapping matrix.

(i) error sets and their properties:

Within the present context, a sequential circuit is, in fact, analysed as a combinational circuit. Error sets and their properties will therefore be investigated with respect to a general combinational circuit, as shown in Fig. 4.9, where $Z = \{z_1, z_2, \ldots z_\nu\}$ is the set of output variables,

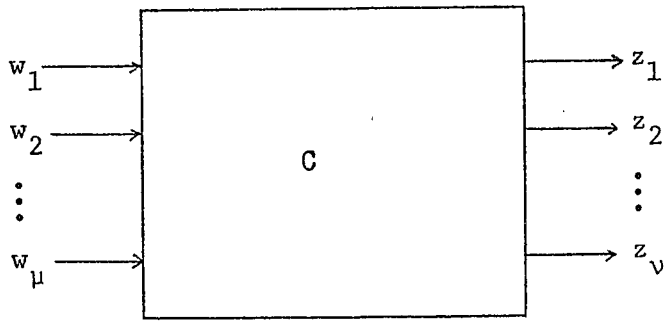


Fig. 4.9

$W = \{w_1, w_2, \ldots, w_\mu\}$ is the set of input variables, and each output variable z_i is a (Boolean) function, $f_i(W)$, of the input variables. Let Z^k denote the k^{th} ν- tuple ($k = 0, 1, \ldots, 2^\nu - 1$) of output variables; similarly, let W^j

denote the j^{th} μ - tuple ($j = 0, 1, \ldots, 2^{\mu}-1$) of input variables.

Definition 4.3: Error set $G_j(z_i)$, associated with output variable z_i, is defined as the set of component malfunctions that can propagate to variable z_i when W^j is applied to the circuit.

Example: the following error sets are easily obtained from matrix equations (4.8), (4.9), and (4.10):

$G_0(y) = \{\bar{g}_5\}$ $\qquad$ $G_7(y) = \{\bar{g}_1, \bar{g}_5\}$

$G_0(q_1^+) = \{\bar{g}_2, \bar{g}_4, \bar{g}_{M_1}\}$ $\qquad$ $G_7(q_1^+) = \{\bar{g}_1, \bar{g}_3, \bar{g}_4, \bar{g}_{M_1}\}$

$G_0(q_2^+) = \{\bar{g}_1, \bar{g}_7, \bar{g}_9, \bar{g}_{10}, \bar{g}_{11}, \bar{g}_{M_2}\}$ $\qquad$ $G_7(q_2^+) = \{\bar{g}_1, \bar{g}_2, \bar{g}_3, \bar{g}_4, \bar{g}_6, \bar{g}_8, \bar{g}_9, \bar{g}_{10}, \bar{g}_{11}, \bar{g}_{M_2}\}$

Definition 4.4: Probability $p(G_j(z_i))$, associated with error set $G_j(z_i) = \{\bar{g}_1, \bar{g}_2, \ldots, \bar{g}_d\}$, is defined as

$p(G_j(z_i)) = p(\bar{g}_1) + p(\bar{g}_2) + \ldots + p(\bar{g}_d)$, where

$p(G_j(z_i)) = 0$ if $G_j(z_i) = \emptyset$ (the empty set).

Using the above definitions, let us now consider the error set $G_j(Z^k)$ associated with output tuple Z^k when input tuple W^j is applied to the circuit.

Given that μ - tuple W^j is applied to circuit C, a unique ν - tuple, denoted as Z^{c_j}, is obtained if circuit C is errorless. However, let us assume that a component has malfunctioned and that output tuple Z^k is obtained. Furthermore, let us assume Z^k to be such that the value of a single variable, z_i, is in error. Then, by assumption 4.2.5, the occurrence of Z^k is due to anyone of the component malfunctions that can propagate to variable z_i, and only to variable z_i, when the input tuple W^j is applied to the circuit, i.e.

$$G_j(Z^k) = G_j(z_i) - \bigcup_{\substack{m=1 \\ m \neq i}}^{\nu} G_j(z_m),$$

where - denotes set difference, and $\bigcup$ denotes set union.

Similarly, if Z^k is such that the values of two variables, z_{i_1} and z_{i_2}, are in error, then the occurrence of Z^k is due to anyone of the component malfunctions that can propagate to <u>both</u> variables z_{i_1} and z_{i_2}, and only to

these two variables, when W^j is applied to the circuit, i.e.

$$G_j(Z^k) = (G_j(z_{i_1}) \bigcap G_j(z_{i_2})) - \bigcup_{\substack{m=1 \\ m \neq i_1 \\ m \neq i_2}}^{\nu} G_j(z_m).$$

By assumption 4.2.3, component malfunctions are statistically independent. Therefore, in each case, the probability of obtaining Z^k when W^j is applied is given as $p(Z^k/W^j) = p(G_j(Z^k))$.

The above results are easily extended to n variables in error as the following general property.

Property 4.1: Let Z^k correspond to an output ν - tuple for which the values of n variables, $z_{i_1}, z_{i_2}, \ldots, z_{i_n}$, $(1 \leq n \leq \nu)$ are in error when input μ - tuple W^j is applied to the circuit. Then, $p(Z^k/W^j) = p(G_j(Z^k))$, where

$$G_j(Z^k) = \bigcap_{t=1}^{n} G_j(z_{i_t}) - \bigcup_{\substack{m=1 \\ m \neq i_t}}^{\nu} G_j(z_m).$$

Note that, in the case where $n = \nu$, $G_j(Z^k) = \bigcap_{t=1}^{\nu} G_j(Z_t)$.

Using property 4.1, one can compute the probability of obtaining any erroneous output tuple Z^k, given that input tuple W^j is applied to the circuit.

Next, let us consider the probability of obtaining the correct ν - tuple Z^{c_j} when W^j is applied. An erroneous output tuple is obtained whenever anyone of the component malfunctions that can propagate to any variable z_i $(i = 1, 2, \ldots, \nu)$ does occur. Thus, the correct ν - tuple Z^{c_j} is obtained iff none of the component malfunctions of the set $\bigcup_{m=1}^{\nu} G_j(z_m)$ occur. Hence,

Property 4.2: $p(Z^{c_j}/W^j) = 1 - p(\bigcup_{m=1}^{\nu} G_j(z_m))$.

Furthermore, because of the deterministic nature of any given circuit C, a unique ν - tuple is obtained for each component malfunction that can occur when any particular μ - tuple W^j is applied to the circuit. Thus,

Property 4.3: $G_j(Z^{k_1}) \bigcap G_j(Z^{k_2}) = \emptyset$ for any $k_1 \neq k_2 \neq c_j$.

Moreover, for each input tuple W^j, any component malfunction in any set $G_j(Z^k)$, $k \neq c_j$, is contained in at least one error set $G_j(z_i)$, and any

malfunction in any set $G_j(z_i)$, i = 1, 2, ..., ν, is contained in a unique set $G_j(Z^k)$ (by property 4.3). Hence,

Property 4.4: $\bigcup_{\substack{k=0 \\ k\neq c_j}}^{2^\nu-1} G_j(Z^k) = \bigcup_{m=1}^{\nu} G_j(z_m)$.

Assertion 4.3: $\sum_{k=0}^{2^\nu-1} p(Z^k/W^j) = 1$.

proof: $\bigcup_{\substack{k=0 \\ k\neq c_j}}^{2^\nu-1} G_j(Z^k)$ represents a union of mutually exclusive sets (by property 4.3),

$$\therefore\ p\Big(\bigcup_{\substack{k=0 \\ k\neq c_j}}^{2^\nu-1} G_j(Z^k)\Big) = \sum_{\substack{k=0 \\ k\neq c_j}}^{2^\nu-1} p(G_j(Z^k)) = \sum_{\substack{k=0 \\ k\neq c_j}}^{2^\nu-1} p(Z^k/W^j) \text{ (by property 4.1)},$$

but $$p\Big(\bigcup_{\substack{k=0 \\ k\neq c_j}}^{2^\nu-1} G_j(Z^k)\Big) = p\Big(\bigcup_{m=1}^{\nu} G_j(z_m)\Big) \text{ (by property 4.4)},$$

thus, $$\sum_{k=0}^{2^\nu-1} p(Z^k/W^j) = p(Z^{c_j}/W^j) + \sum_{\substack{k=0 \\ k\neq c_j}}^{2^\nu-1} p(Z^k/W^j) = 1 - p\Big(\bigcup_{m=1}^{\nu} G_j(z_m)\Big) + p\Big(\bigcup_{m=1}^{\nu} G_j(z_m)\Big) \text{ (by prop. 4.2)}$$

$$= 1 \qquad \text{Q.E.D.}$$

Our next step is to extend the above properties to subsets of variables. Consider a subset $S \subset Z$, where $S = \{z_1, z_2, \ldots, z_\sigma\}$, $1 \leq \sigma \leq \nu$. Let S^t denote the t^{th} σ-tuple of variables $z_1, z_2, \ldots, z_\sigma$, where $t=0,1,\ldots, 2^\sigma-1$, and let S^{c_j} denote the correct σ - tuple obtained when μ - tuple W^j is applied to the circuit. Clearly, properties 4.1, 4.2, 4.3, 4.4 also apply to subset S. Hence,

Property 4.5: (a) $p(S^t/W^j) = p(G_j(S^t))$, $t \neq c_j$

(b) $p(S^{c_j}/W^j) = 1 - p\Big(\bigcup_{m=1}^{\sigma} G_j(z_m)\Big)$

(c) $G_j(S^t1) \bigcap G_j(S^t2) = \emptyset$ for any $t_1 \neq t_2 \neq c_j$

(d) $\bigcup_{\substack{t=0 \\ t \neq c_j}}^{2^\sigma-1} G_j(S^t) = \bigcup_{m=1}^{\sigma} G_j(z_m)$.

Assertion 4.4: $\sum_{t=0}^{2^\sigma-1} p(S^t/W^j) = 1$

proof: similar to that of assertion 4.3.

(ii) using error sets to compute the entries of the probabilistic mapping matrix:

Let us now use the general results obtained above to compute the probabilistic mapping matrix of a sequential circuit, as shown in Fig. 3.1, where

$X = \{x_1, x_2, \ldots, x_\alpha\}$ is the set of input variables,

$Y = \{y_1, y_2, \ldots, y_\beta\}$ is the set of output variables,

$Q = \{q_1, q_2, \ldots, q_\gamma\}$ is the set of present-state variables,

$Q^+ = \{q_1^+, q_2^+, \ldots, q_\gamma^+\}$ is the set of next-state variables.

The probabilistic entries, $p(s_j, v_r/s_i, u_p)$, where $s_j = Q^{+j}$, $j = 0, 1, \ldots, 2^\gamma-1$,

$s_i = Q^i$, $i = 0, 1, \ldots, 2^\gamma-1$,

$u_p = X^p$, $p = 0, 1, \ldots, 2^{\alpha-1}$,

and $v_r = Y^r$, $r = 0, 1, \ldots, 2^\beta-1$,

correspond to the joint probabilities associated with the set $Z = Q^+ \bigcup Y$ of all next-state and output variables.

Thus, letting $Z^{jr} = Q^{+j}\, Y^r$, the jr^{th} $(\gamma+\beta)$ - tuple of all next-state and output variables, where $jr = 0, 1, \ldots, 2^{\gamma+\beta}-1$,

and letting $W^{ip} = Q^i\, X^p$, the ip^{th} $(\gamma+\alpha)$ - tuple of all present-state and input variables, where $ip = 0, 1, \ldots, 2^{\gamma+\alpha}-1$,

$p(s_j, v_r/s_i, u_p) = p(Z^{jr}/W^{ip})$, as defined by properties 4.1 and 4.2.

Assertion 4.5: The probabilistic mapping matrix obtained with error sets (as defined above) is a stochastic matrix.

proof: follows directly from assertion 4.3.

However, by property 4.5, the error sets associated with the next-state γ - tuples Q^{+j}, $G_{ip}(Q^{+j})$, and the error sets associated with the output β - tuples Y^r, $G_{ip}(Y^r)$, can be obtained separately. Therefore, in computing the probabilistic mapping matrix, one can obtain, as intermediate results, the state-transition matrices and output matrices associated with the given sequential circuit, i.e.

$$p(s_j/s_i, u_p) = p(Q^{+j}/Q^iX^p)$$

$$= \begin{cases} p(G_{ip}(Q^{+j})), & \text{if } s_j \text{ corresponds to an erroneous next-state,} \\ 1-p(\bigcup_{m=1}^{\gamma} G_{ip}(q_m^+)), & \text{if } s_j \text{ corresponds to the correct next-state.} \end{cases}$$

$$p(v_r/s_i, u_p) = p(Y^r/Q^iX^p)$$

$$= \begin{cases} p(G_{ip}(Y^r)), & \text{if } v_r \text{ corresponds to an erroneous output,} \\ 1-p(\bigcup_{m=1}^{\beta} G_{ip}(y_m)), & \text{if } v_r \text{ corresponds to a correct output.} \end{cases}$$

The entries of the probabilistic mapping matrix can then be computed as follows:

$$p(s_j, v_r/s_i, u_p) = p(Q^{+j}Y^r/Q^iX^p)$$

$$= \begin{cases} p(G_{ip}(Q^{+j}) \bigcap G_{ip}(Y^r)), & \text{if } Q^{+j} \text{ and } Y^r \text{ both correspond to erroneous tuples,} \\ p(G_{ip}(Q^{+j}) - \bigcup_{m=1}^{\beta} G_{ip}(y_m)), & \text{if } Q^{+j} \text{ is erroneous and } Y^r \text{ is correct,} \\ p(G_{ip}(Y^r) - \bigcup_{n=1}^{\gamma} G_{ip}(q_n^+)), & \text{if } Q^{+j} \text{ is correct and } Y^r \text{ is erroneous,} \\ 1-p((\bigcup_{n=1}^{\gamma} G_{ip}(q_n^+)) \bigcup (\bigcup_{m=1}^{\beta} G_{ip}(y_m))), & \text{if } Q^{+j} \text{ and } Y^r \text{ both correspond to correct tuples.} \end{cases}$$

For our example, $U = \{u_0, u_1\}$, $V = \{v_0, v_1\}$, and $S = \{s_0, s_1, s_2, s_3\}$. Row (s_0, u_0) of the transition matrix associated with input symbol u_0 (as defined in Chapter II, section 3) is obtained as follows:

the correct next-state associated with state-input pair (s_0, u_0) is s_0, and the required error sets (obtained from equations (4.8) and (4.9)) are $G_0(q_1^+) = \{\bar{g}_2, \bar{g}_4, \bar{g}_{M_1}\}$ and $G_0(q_2^+) = \{\bar{g}_1, \bar{g}_7, \bar{g}_9, \bar{g}_{10}, \bar{g}_{11}, \bar{g}_{M_2}\}$; hence,

$$p(s_0/s_0, u_0) = 1 - p(G_0(q_1^+) \bigcup G_0(q_2^+))$$
$$= 1 - p(\bar{g}_1) - p(\bar{g}_2) - p(\bar{g}_4) - p(\bar{g}_7) - p(\bar{g}_9) - p(\bar{g}_{10}) - p(\bar{g}_{11}) - p(\bar{g}_{M_2})$$

$$p(s_1/s_0, u_0) = p(G_0(q_2^+) - G_0(q_1^+))$$
$$= p(\bar{g}_1) + p(\bar{g}_7) + p(\bar{g}_9) + p(\bar{g}_{10}) + p(\bar{g}_{11}) + p(\bar{g}_{M_2})$$

$$p(s_2/s_0, u_0) = p(G_0(q_1^+) - G_0(q_2^+)) = p(\bar{g}_2) + p(\bar{g}_4) + p(\bar{g}_{M_1})$$

$$p(s_3/s_0, u_0) = p(G_0(q_1^+) \bigcap G_0(q_2^+)) = 0$$

Similarly, row (s_0, u_0) of the output matrix associated with input symbol u_0 (as defined in Chapter II, section 3) is obtained as follows:

the correct output symbol associated with (s_0, u_0) is v_0, and the required error set is $G_0(y) = \{\bar{g}_5\}$ (obtained from equation (4.10)); hence,

$$p(v_0/s_0, u_0) = 1 - p(G_0(y)) = 1 - p(\bar{g}_5)$$

$$p(v_1/s_0, u_0) = p(G_0(y)) = p(\bar{g}_5)$$

Using the results obtained above for row (s_0, u_0), the corresponding row of the probabilistic mapping matrix can be obtained as shown above. Hence,

$$p(s_0, v_0/s_0, u_0) = 1 - p(\bar{g}_1) - p(\bar{g}_2) - p(\bar{g}_4) - p(\bar{g}_5) - p(\bar{g}_7) - p(\bar{g}_9) - p(\bar{g}_{10}) - p(\bar{g}_{11}) - p(\bar{g}_{M_1}) - p(\bar{g}_{M_2})$$

$$p(s_0, v_1/s_0, u_0) = p(\bar{g}_5)$$

$$p(s_1, v_0/s_0, u_0) = p(\bar{g}_1) + p(\bar{g}_7) + p(\bar{g}_9) + p(\bar{g}_{10}) + p(\bar{g}_{11}) + p(\bar{g}_{M_2})$$

$$p(s_1, v_1/s_0, u_0) = 0$$

$p(s_2, v_0/s_0, u_0) = p(\bar{g}_2)+p(\bar{g}_4)+p(\bar{g}_{M_1})$

$p(s_2, v_1/s_0, u_0) = 0$

$p(s_3, v_0/s_0, u_0) = 0$

$p(s_3, v_1/s_0, u_0) = 0$

The other rows of the state-transition and output matrices and the corresponding rows of the probabilistic mapping matrix are similarly computed.

Finally, the numerical entries of the probabilistic mapping matrix are obtained by replacing each malfunction probability, $p(\bar{g}_k)$, with its given value. In the case of input-independent malfunctions, each probability has a unique value, and the evaluation becomes very straightforward. In the case of input-dependent malfunctions, evaluation of $p(\bar{g}_k)$ requires a knowledge of the input values to component g_k when the corresponding state input pair (s_i, u_p) is applied to the circuit; these input values are immediately determined from the values of the corresponding propagation functions. Note that, in the latter case, a more direct evaluation could be achieved by replacing each malfunction probability with its highest value, regardless of the input values to the component, in which case the matrix entries would correspond to upper bounds of error probabilities.

For our example, the following probabilistic mapping matrix is obtained, assuming that the malfunction probabilities are input-independent and that

$$p(\bar{g}_k) = \begin{cases} a, & \text{for } k = 1, 2, \ldots, 11, \\ b, & \text{for } k = M_1, M_2. \end{cases}$$

$P =$

q_1q_2x \ $q_1^+q_2^+y$	000	001	010	011	100	101	110	111
000	1-8a-2b	a	5a+b	0	2a+b	0	0	0
001	3a+b	0	a	0	1-9a-2b	a	4a+b	0
010	5a+b	0	1-9a-2b	2a	0	0	2a+b	0
011	2a	a	a+b	0	2a+b	0	1-7a-2b	a
100	3a+b	0	a	0	1-9a-2b	a	4a+b	0
101	5a+b	0	1-9a-2b	a	a	0	2a+b	0
110	2a	a	a+b	0	2a+b	0	1-7a-2b	a
111	a	1-10a-2b	0	6a+b	0	b	a	2a

In conclusion of this section, the procedure is summarized as follows:

step 1: by inspection of the given sequential circuit, obtain an error propagation graph for each output variable and each next-state variable.

step 2: by inspection of each graph, determine the path-functions that govern the propagation of each component malfunction to each output and next-state variable (M-formulation).

step 3: in cases of reconvergent fanout, modify all sum-of-product path-functions corresponding to the component malfunctions which can propagate to a fanout point (expression F_{rf}).

step 4: compute the value of each path-function for all state-input tuples, as formulated with equation (4.7).

step 5: obtain, for every state-input tuple, the error sets associated with each next-state tuple and each output tuple. Combining these sets as described above, obtain the state-transition and output matrices, and thereupon compute the probabilistic mapping matrix.

6. Conclusion:

In computing the probabilistic mapping matrix of a given sequential circuit, the need to consider the probability associated with each and every state-output tuple for all values of input and present-state variables calls for fairly tedious computations. By contrast with the matrix and algebraic methods, the error propagation method developed in this chapter provides an efficient procedure whereby the amount of computations required at each step remains within reasonable proportion to the size and complexity of the given circuit.

As outlined at the beginning of the chapter, the error propagation method is based on the concept of path sensitizing. With this new approach to the problem, the simplifying assumption that at most one component may

malfunction at any time instant is included within the very concept of the method, and it thereby provides simplifications throughout the corresponding procedure. With the matrix method, this assumption is used only for simplifying the results of matrix operations while, with the algebraic method, it is used only at the very end of the procedure for eliminating certain products of malfunction variables.

Furthermore, the variety of computational tools used with the error propagation procedure provides greater simplification and increased efficiency at each step. The basic information regarding the propagation of component malfunctions is easily obtained by inspection of the error propagation graphs. Then, Boolean algebra is applied to efficiently determine the propagated malfunctions for all values of state and input variables. Finally, set theory proves to be very advantageous in computing the joint conditional probabilities corresponding to each entry of the probabilistic mapping matrix.

By contrast, the algebraic method resorts exclusively to Boolean algebra. As shown in the Appendix, the Boolean functions that must be computed for the output and next-state variables (of our example) become very complex and accordingly difficult to simplify. Furthermore, the joint probabilities corresponding to the entries of matrix P are very tediously obtained through Boolean products of all these complex functions.

Similarly, the matrix method resorts exclusively to matrix operations. The resulting procedure consists of two basic steps: the partitioning of the circuit and the computation of the block matrices and of the overall matrix. As the circuit increases in complexity, the partitioning step becomes a problem in itself, since it must be "properly" achieved in order to maintain the required computations at a minimum. By contrast, the error propagation graphs always remain straightforward tree structures. Furthermore, the matrix procedure offers

no simplification other than the deletion of products of malfunction probabilities. As shown in the Appendix, the computations involved in obtaining the block matrices and the overall matrix become extremely tedious, and the matrices reach very large sizes. A good example of the "brute-force" nature of the matrix method is obtained with respect to output variable y, corresponding to the output of component g_5 of our example. Although y appears as a simple line through four blocks of the partitioned circuit, the computations required with each one of these blocks must be duplicated in order to account for y in the probabilistic mapping matrix. With the error propagation procedure, variable y is represented with a very simple graph and a correspondingly simple path-function, and it is easily accounted for with the final computations involving set theory.

In view of the above discussion, it is clear that the error propagation method has provided a more efficient procedure for computing the probabilistic mapping matrix. In addition, with its graphical representation the method conveys much more insight into the problem of circuit reliability and, as shall be discussed in Chapter VI, it thereby offers a meaningful approach to many other related problems.

CHAPTER V

FAULT DETECTION IN COMBINATIONAL CIRCUITS

1. Introduction:

The purpose of this chapter is to extend the results obtained in Chapter IV to the detection of faults in combinational circuits. As will be shown in Section 2, this becomes straightforward, since the error propagation method was developed from the concept of path-sensitizing, which was originally introduced for the purpose of fault detection in combinational circuits [14]. However, some modifications will need to be introduced, since the problems of fault detection and of reliability analysis differ in certain respects.

In the case of sequential circuits, fault detection involves the additional problem of state identification whenever the state variables of the circuit are not observable. Otherwise, if the state variables are observable, or better, if the feedback lines of the circuit can be physically disconnected and the present-state variables accessed as inputs, then fault detection can be achieved with procedures that apply to combinational circuits. In this latter case, the results obtained with the error propagation method can be easily extended to sequential circuit fault detection, since, in computing the probabilistic mapping matrix, a sequential circuit is actually analysed as a combinational circuit.

At this point, one may question the usefulness of extending the error propagation method to fault detection when a variety of techniques have already been developed for that purpose [16]. For example, Armstrong [19] has developed a path-sensitizing procedure, and Kilin To [18] has provided an efficient technique with his fault-folding approach. However, fault detection alone gives no precise

information about the reliability of the circuit, when, in fact, these two problems are very closely related, i.e. reliability analysis provides a measure of how "good" the circuit is (and hence, of how often it should be tested), while fault detection determines whether or not the circuit is operating properly. By contrast, the error propagation method provides a unified approach to these two problems, since, as will be described in Section 3, it can be used for fault detection as well as for reliability analysis.

Another point to consider is the testing philosophy associated with fault detection. In general, fault detection procedures aim at obtaining a minimal set of test sequences that detect all possible faults in the circuit. However, as the circuit increases in size and in the number of input-output lines, these complete test sets become fairly large, and testing time and cost can become excessive. In that case, if the number of tests can be significantly reduced by testing only for those faults that are more likely to occur, testing time and expense can be reduced by spacing the complete test set at longer intervals and making the limited tests at shorter intervals. Hence, those faults that are more likely to occur are tested for more frequently, while the faults unlikely to occur are less often tested for. With respect to this testing philosophy, the error propagation method provides a very flexible approach to obtaining limited test sets, as will be shown in Section 4.

2. Fault Detection and the Error Propagation Method:

The reliability problem analysed in Chapter IV consisted essentially in determining the (temporary) component malfunctions that could propagate to the output terminals of the circuit for every input tuple (state-input pair). By contrast, fault detection aims at generating only a subset of input tuples that will test for all possible faults. In order to detect the presence of a fault, its effect must be allowed to propagate to an observable output of the

circuit, and this is precisely the information provided by the error propagation method in computing the probabilistic mapping matrix. However, since the two problems differ as to the nature of the errors involved, some modifications must be effected in order to achieve fault detection.

First, let us consider fault detection at the component level. In Chapter IV, the error propagation method was developed with respect to temporary component malfunctions, whereby there was no need to distinguish between an error at the input and an error at the output of the component, or to distinguish between the component erroneously outputing a ONE and the component erroneously outputing a ZERO. By contrast, fault detection is concerned with permanent faults, which are generally considered as a component input or output either stuck-at-zero (s-a-0) or stuck-at-one (s-a-1). In testing for these faults, it becomes important to distinguish between input faults and output faults, and between s-a-0 and s-a-1 faults, because the corresponding tests are different. For example, consider a two-input AND gate, as shown in Fig. 5.1. In testing for a particular fault, the input tuple, referred to as a test tuple, must be

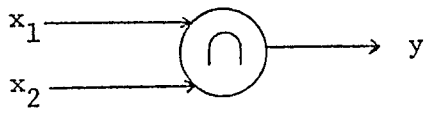


Fig. 5.1

such that the value of y differs according to whether the fault is present or not. Tests for all faults associated with the above AND gate are given in Table 5.1. As seen from the table, a fault in x_1 is detected at output y iff

		FAULTS					
TEST TUPLE		x_1		x_2		y	
x_1	x_2	s-a-0	s-a-1	s-a-0	s-a-1	s-a-0	s-a-1
0	0	-	-	-	-	-	TEST
0	1	-	TEST	-	-	-	TEST
1	0	-	-	-	TEST	-	TEST
1	1	TEST	-	TEST	-	TEST	-

TABLE 5.1

$x_2 = 1$, and a fault in x_2 is detected at output y iff $x_1 = 1$. Hence, these input tests correspond exactly to the error propagation graph obtained for an AND gate. Similarly, it can be shown that this correspondence holds for the other types of components.

In extending fault detection to the circuit level, each fault in the circuit could be tested individually, as in Table 5.1, by determining those (primary) input tuples that allow the effect of the fault to propagate to an observable output. However, that would be equivalent to obtaining a fault table for the circuit. As described by Kohavi [17], the fault table has a row associated with each input tuple, and a column associated with each possible fault, listing all those input tuples which can detect the corresponding fault. The table may be simplified by using the following property: if every test for a fault i is also a test for fault j, then the column corresponding to fault j can be removed. Finding a minimal set of tests that detect all faults in the circuit will then consist in determining a minimal set of rows that "cover" the fault table. As a first step in finding the smallest cover, one determines the "essential" tests, where a test is said to be essential if it is the only test available for detecting a particular fault. The problem is then reduced to obtaining a minimal set of non-essential tests that detect all remaining faults. The fault table thus provides a tool for finding a minimal set of test tuples, but as the circuit increases in size, the procedure becomes impracticable because of the size of the table. In order to obtain a more efficient procedure, only those tests which detect a large number of faults should be considered, as is the case with the path-sensitizing technique.

The concept of path-sensitizing is based on the following two properties:

Property 5.1: Those input tuples that test for s-a-0 and s-a-1 faults at any single input of a component also provide tests for s-a-0 and s-a-1 faults at the output of the component.

Property 5.1 can be easily verified with Table 5.1 in the case of an AND gate.

Property 5.2: If the output from a component g_i is used as an input to another component g_j, then test tuples for detecting s-a-0 and s-a-1 faults at the output of g_i will also detect s-a-0 and s-a-1 faults at the corresponding input of g_j.

Hence, in testing a path $f_i g_1 g_2 \cdots g_k f_j$ from an input f_i to an output f_j, as shown in Fig. 5.2, it is sufficient to test for s-a-0 and s-a-1 faults at input

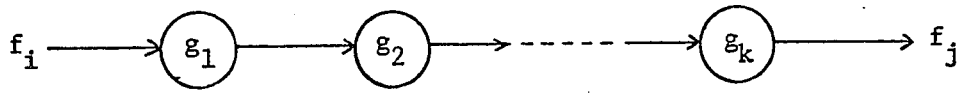


Fig. 5.2

f_i in order to test for s-a-0 and s-a-1 faults at all inputs and outputs encountered on the path, provided that all other inputs to components $g_1, g_2, \ldots, g_k$ allow a fault at f_i to propagate to output f_j. Clearly, the error propagation graph (shown in Fig. 5.3) corresponding to path $f_i g_1 g_2 \cdots g_k f_j$

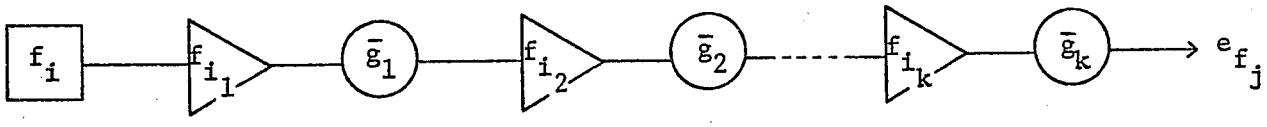


Fig. 5.3

specifies the required propagation conditions with path function $F_i = f_{i_1} f_{i_2} \cdots f_{i_k}$, i.e. the Boolean product of all propagation functions encountered on the path. The square node associated with f_i is referred to as a "test" node, indicating that the path is tested at f_i. Note that, although output f_j must correspond to an observable output of the circuit in order to detect all faults in the above

path, input f_i need not be a primary input. In the case where the test node does correspond to a primary input, the path is referred to as primary path.

Let us now consider how the error propagation graphs can be used to test for all faults in a given circuit. Within the context of reliability analysis, input and present-state variables were assumed errorless, and errors at the inputs of a component were included as a general component malfunction. However, within the context of fault detection, component inputs corresponding to primary inputs of the circuit can also become s-a-0 or s-a-1, and corresponding test nodes (and their associated propagation functions) must therefore be added to the graphs. These modified error propagation graphs then indicate all paths through which each and every fault can propagate on to the observable outputs of the circuit.

A first attempt at obtaining a set of tests for detecting all faults in the circuit would be to test all primary paths. However, such a procedure suffers from the following drawbacks. In testing some of the paths, especially those which involve a relatively large number of propagation functions, it may occur that none of the input tuples which satisfy the corresponding path-function will provide a test for the s-a-0 and/or s-a-1 faults at the primary input, in which case some of the faults in the path will not be detected. Furthermore, in some cases where a component output fans out to two or more components, the subpath which precedes the branch point will be unnecessarily tested more than once. Finally, in the case of reconvergent fanout, the input tuples found for testing one of the fanout paths may allow the faults at or before the fanout point to propagate through the other path and "desensitize" the former path at the point of reconvergence, in which case these faults will not be detected.

A better approach to the problem of detecting all faults is obtained with "path splitting". Consider the two primary paths shown in Fig. 5.4, where f_2,

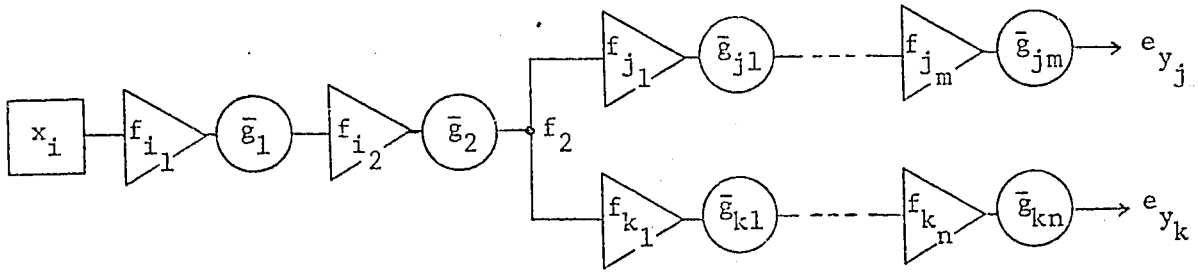


Fig. 5.4

output of component g_2, is referred to as the branch point. If both of these paths are tested at the primary input x_i, then subpath $x_i g_1 g_2 f_2$ will be unnecessarily tested twice. With path splitting, only one of these two paths is tested at x_i, while the other path is split and tested at the branch point f_2. The choice of which path to split rests on the following observations. In general, the larger the number of propagation functions considered with a path, the more restricted the choice of input tuples for testing the path. Furthermore, if a path cannot be tested because of conflicting propagation functions, path splitting will generally provide a test by reducing the number of propagation functions that need to be considered. Therefore, the longer path (i.e. the path which involves the larger number of propagation functions) will be generally chosen as the path to split, except in the case where the shorter path cannot be tested at the primary input. However, note that this method of selection cannot guarantee the minimality of the resulting set of tests. A more complete method would consist in testing each primary path and then choosing the splitting that provides the larger choice of test tuples; but since this method can neither guarantee minimality while requiring more computations, it will not be considered any further.

By comparison with our first attempt, the path splitting approach reduces redundant path testing, i.e. it reduces the number of subpaths which are unnecessarily tested more than once (except for those subpaths which follow a point of convergence of two or more paths, in which case redundant testing

cannot be avoided). Accordingly, the amount of computations needed to obtain a set of test tuples will be reduced, especially in cases where relatively long paths can be shortened with path splitting.

Furthermore, the set of tests obtained with the path splitting approach will detect all faults that can be detected, since those paths which previously involved conflicting propagation functions will generally become testable through path splitting. However, in the case where the circuit includes redundancies, some of the faults cannot be detected because their effect does not propagate to any observable output for any of the input tuples. In testing paths which include such undetectable faults, path splitting can be used to exclude these faults and test for the remaining faults of the path, as will be shown in the next section.

Path splitting also provides a better approach to the problem of reconvergent fanout. Whenever those faults that can propagate to the fanout point are tested with a path which does not include the point of reconvergence, as shown schematically in Fig. 5.5, the problem of path desensitizing can be

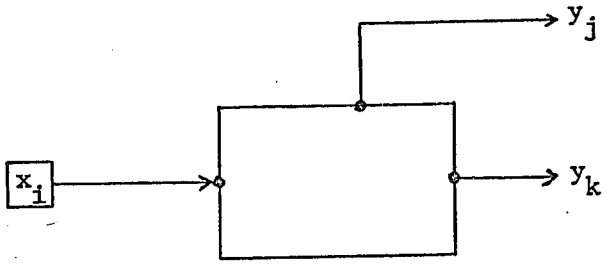


Fig. 5.5

avoided. The remaining subpaths of the reconvergent fanout can then be tested from the branch points only. However, in the case where an alternate path (as described above) is not available, the detection of faults at or before the fanout point requires special consideration, as will be discussed in the next section.

Finally, note that, since some paths are reduced, the correspondingly simpler path-functions will usually offer a greater choice of test tuples. Hence, by comparison with our first attempt, the number of required test tuples will often be reduced with the path splitting approach.

3. Procedure for Obtaining a Complete Test Set:

This section aims at developing a procedure for detecting all faults in a combinational circuit, using the results obtained in computing the probabilistic mapping matrix as outlined in Chapter IV. The procedure will be illustrated with the sequential circuit considered in the Appendix, assuming that it can be tested as a combinational circuit by disconnecting the feedback lines through the memory.

3.1 Selection and testing of the primary paths:

As outlined in the previous section, each primary input to each component must be tested for s-a-0 and s-a-1 faults, and with the path splitting approach, only one path needs to be tested for each one of these inputs. For our example, eight primary inputs must be considered, i.e. input x to components g_1 and g_2, input q_1 to components g_1 and g_2, and input q_2 to components g_5, g_6, g_7, and g_{10}.

The first step is therefore to choose and test an appropriate set of primary paths. In order to achieve this selection, the error propagation graphs of the circuit are used to indicate all possible paths associated with each primary input to each component.

Example: all primary paths originating at input x to component g_1 are shown in Fig. 5.6.

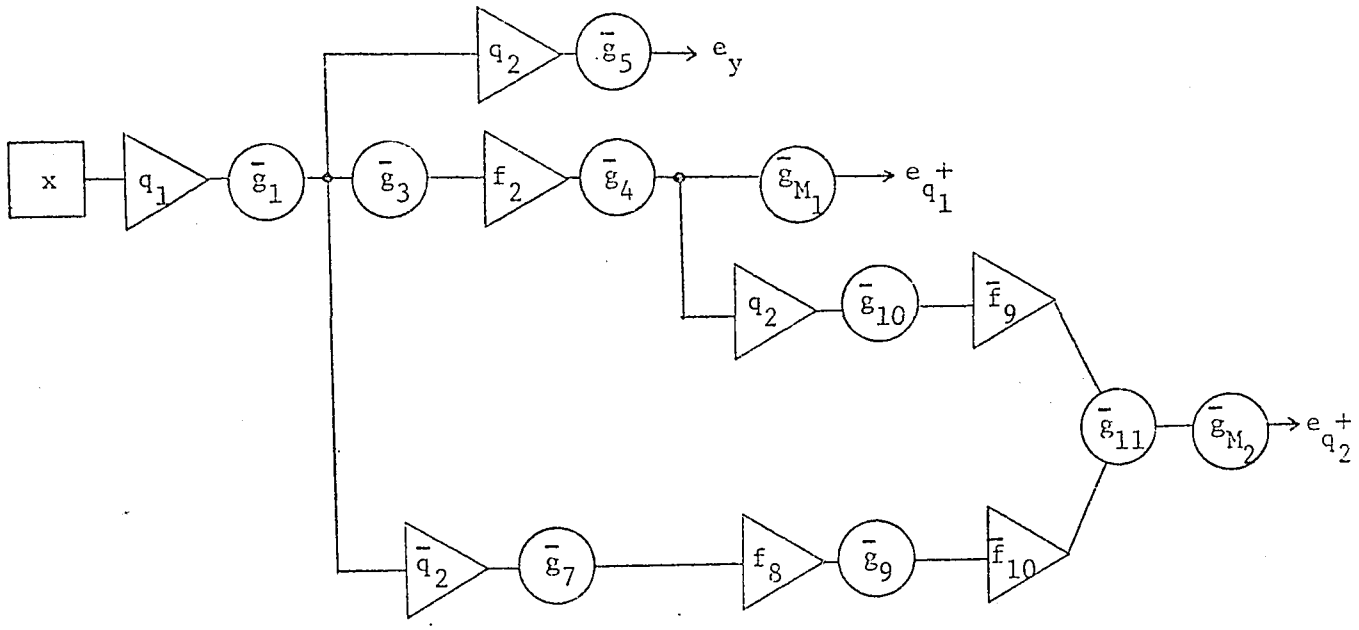


Fig. 5.6

The first choice corresponds to the shortest path which does not include both the fanout point and the reconvergence point of a reconvergent fanout. From Fig. 5.6, one of two paths can be chosen: xg_1g_5y or $xg_1g_3g_4g_{M_1}q_1^+$. Primary path $xg_1g_3g_4g_{M_1}q_1^+$ is arbitrarily chosen.

However, in the case of a primary input for which all paths do include both the fanout point and reconvergence point, the possibility of path desensitizing must be investigated. First, one determines those input tuples for which a fault at the output of the first component of the path can propagate through the reconvergence point onto an observable output; these tuples were previously obtained in evaluating the path-function associated with that first component (Chapter IV, section 5.4). If these input tuples can now be used to detect s-a-0 and s-a-1 faults at the corresponding primary input, then both branches of the reconvergent fanout can be simultaneously tested. Otherwise, each branch will need to be tested separately, and the chosen primary path will include the shorter branch.

After choosing a particular primary path, as shown in Fig. 5.7, the next step is to obtain a complete test for it. All faults in the path

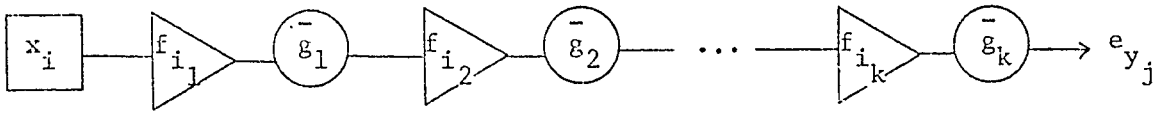


Fig. 5.7

can be detected with two input tuples. One test tuple is obtained with Boolean function $x_i F_i$ for testing x_i s-a-0, and the other one is obtained with Boolean function $\bar{x}_i F_i$ for testing x_i s-a-1, where $F_i = f_{i_1} f_{i_2} \ldots f_{i_k}$ is the corresponding path function. Note that, in each case, the Boolean function may correspond to a sum of input tuples. Also note that this computation is very straightforward, since most of the path-functions have already been obtained while computing the probabilistic mapping matrix.

However, if a complete test cannot be obtained for a chosen primary path, i.e. if $x_i F_i = 0$ and/or $\bar{x}_i F_i = 0$, then an alternate path must be chosen. In the case where no other path is available, the failure to provide a complete test with respect to the corresponding primary input is due to one or more redundancies in the circuit. In that case, some of the faults will never propagate to any observable output and hence, no test can be obtained for their detection. However, the remaining faults of the path which are detectable must be tested for, and for a path as shown in Fig. 5.7, this can be achieved by successively testing at outputs of components $g_1, g_2, \ldots, g_k$ until a complete test is obtained for the remaining part of the path.

For our example, the selected primary paths and their associated test tuples are listed in Table 5.2, which is referred to as a test table.

PRIMARY PATH	PATH-FUNCTION	s-a-0 TEST	s-a-1 TEST
p_1: $xg_1g_3g_4g_{M_1}q_1^+$	q_1f_2	$xq_1q_2+x\bar{q}_1\bar{q}_2$	$\bar{x}q_1q_2+\bar{x}q_1\bar{q}_2$
p_2: $q_1g_1g_3g_4g_{M_1}q_1^+$	xf_2	$xq_1q_2+x\bar{q}_1\bar{q}_2$	$\bar{x}q_1q_2+\bar{x}q_1\bar{q}_2$
p_3: $xg_2g_4g_{M_1}q_1^+$	$\bar{q}_1f_3$	$x\bar{q}_1q_2+x\bar{q}_1\bar{q}_2$	$\bar{x}\bar{q}_1q_2+\bar{x}\bar{q}_1\bar{q}_2$
p_4: $q_1g_2g_4g_{M_1}q_1^+$	$\bar{x}f_3$	$\bar{x}q_1q_2+\bar{x}q_1\bar{q}_2$	$\bar{x}\bar{q}_1q_2+\bar{x}\bar{q}_1\bar{q}_2$
p_5: q_2g_5y	f_1	xq_1q_2	$xq_1\bar{q}_2$
p_6: $q_2g_6g_8g_9g_{11}g_{M_2}q_2^+$	$f_2f_7\bar{f}_{10}$	xq_1q_2	$xq_1\bar{q}_2$
p_7: $q_2g_7g_9g_{11}g_{M_2}q_2^+$	$\bar{f}_1f_8\bar{f}_{10}$	$\bar{x}\bar{q}_1q_2$	$\bar{x}q_1\bar{q}_2+\bar{x}\bar{q}_1\bar{q}_2+x\bar{q}_1\bar{q}_2$
p_8: $q_2g_{10}g_{11}g_{M_2}q_2^+$	$f_4\bar{f}_9$	$x\bar{q}_1q_2+\bar{x}q_1q_2$	$x\bar{q}_1\bar{q}_2+\bar{x}q_1\bar{q}_2$

Table 5.2

Note that, in a test table, those input tuples which appear as single test tuples correspond to essential tests; the remaining tuples which appear in sums of test tuples are non-essential tests.

3.2 Testing the remaining paths:

In order to detect all faults in the circuit, one must test all those paths which originate at branch points and which have not been tested within a primary path. Since each branch point can be considered as a primary input with respect to the paths that remain to be tested, the corresponding testing is achieved as in the case of primary paths. Note that if a remaining path fans out at a further branch point, then it is analysed in exactly the same way as in the case of a primary path, i.e. the selection and testing of associated paths is achieved as described above.

For our example, the branch points that need to be tested correspond to f_1, f_2 and f_4. The remaining paths and their test tuples are listed in Table 5.3.

REMAINING PATHS	PATH-FUNCTIONS	s-a-0 TEST	s-a-1 TEST
p_9: $f_1 g_5 y$	q_2	xq_1q_2	$\bar{x}q_1q_2+x\bar{\bar{q}}_1q_2+x\bar{q}_1q_2$
p_{10}: $f_1 g_7 g_9 g_{11} g_{M_2} q_2^+$	$\bar{q}_2 f_8 \bar{f}_{10}$	$xq_1\bar{q}_2$	$\bar{x}q_1\bar{q}_2+x\bar{\bar{q}}_1\bar{q}_2+x\bar{q}_1\bar{q}_2$
p_{11}: $f_2 g_6 g_8 g_9 g_{11} g_{M_2} q_2^+$	$q_2 f_7 \bar{f}_{10}$	xq_1q_2	$\bar{x}\bar{q}_1q_2$
p_{12}: $f_4 g_{10} g_{11} g_{M_2} q_2^+$	$q_2 \bar{f}_9$	$x\bar{q}_1q_2+\bar{x}q_1q_2$	xq_1q_2

Table 5.3

3.3 Obtaining the complete set of tests:

The complete set of test tuples must be such that each path considered in the above test tables is tested for both s-a-0 and s-a-1 faults. As outlined in the previous section, the essential tests are determined first, since they must be all included in the set. The problem is then reduced to obtaining the smallest set of non-essential tests that covers all those path faults that have not been tested by essential tests. For simple circuits, this last step can be achieved by inspection of the test tables, but for larger circuits, a tabular technique similar to the procedure associated with the prime implicant chart (for reducing Boolean functions) can be used.

For our example, inspection of the test tables 5.2 and 5.3 reveals that xq_1q_2, $xq_1\bar{q}_2$, and $\bar{x}\bar{q}_1q_2$ are essential test tuples. The smallest set of non-essential tests can then be obtained with Table 5.4, where the X's indicate which path faults (not tested by essential tests) can be tested with each non-essential tuple.

NON-ESSENTIAL TEST TUPLES	PATH FAULTS NOT TESTED BY ESSENTIAL TESTS							
	p_1-a-1	p_2-a-1	p_3-a-0	p_4-a-0	p_7-a-1	p_8-a-0	p_8-a-1	p_{10}-a-1	p_{12}-a-0
$x\bar{q}_1q_2$		X	X			X			X
$x\bar{q}_1\bar{q}_2$		X	X		X		X	X	
$\bar{x}q_1q_2$	X			X		X			X
$\bar{x}q_1\bar{q}_2$	X			X	X		X	X	
$\bar{x}\bar{q}_1\bar{q}_2$					X			X	

Table 5.4

In fact, Table 5.4 is similar to a fault table (as described in the previous section), and it is thereby reduced by removing the columns corresponding to p_3-a-0, p_4-a-0, p_7-a-1, p_{10}-a-1, and p_{12}-a-0. Is is then very easy to see that the smallest set of non-essential tests can be either $\{x\bar{q}_1q_2, \bar{x}q_1\bar{q}_2\}$ or$\{x\bar{q}_1\bar{q}_2, \bar{x}q_1q_2\}$. Hence, one complete set of tests corresponds to $\{xq_1q_2, xq_1\bar{q}_2, \bar{x}\bar{q}_1q_2, x\bar{q}_1\bar{q}_2, \bar{x}q_1q_2\}$. Note that, for our example, the circuit contains no redundancies, and all possible faults can be detected with the above set of tests. Furthermore, referring back to the selection of primary paths discussed in sub section 3.1, note that the selection of path xg_1g_5y would have led to a complete test set of six input tuples, whereas the selection of path $xg_1g_3g_4g_{M_1}q_1^+$ has led to the above set of five tuples. Hence, as mentioned in the previous section, the above procedure cannot guarantee the minimality of the resulting test set.

4. Selective Testing:

The purpose of this section is to illustrate the flexibility of the error propagation method in obtaining limited test sets as discussed at the beginning of this chapter.

In general, the assumptions that can be made as to the relatively higher probability of occurrence of certain faults will depend on the technology used in realizing the circuit. However, in some cases, these assumptions may also relate to the structure of the circuit or to the relative reliability of its components. A large variety of assumptions can thus be made in obtaining limited test sets, but only two specific cases of selective testing are considered in this section. A more general discussion is deferred to the conclusion of the chapter.

Case 1: Testing only for output faults.

With a number of technologies, especially in the case of integrated circuits, it is observed that a very high percentage of faults detected inside a circuit occur at the output of its components. Technologically, this is usually explained by the fact that the output components of a gate carry a heavier load than the input components and are thereby more prone to becoming faulty. We will therefore describe how the procedure developed in the previous section can be easily modified for obtaining a set of tests under the assumption that only output faults can occur. This modification will be illustrated with the example considered in the previous section.

With the present assumption, there is no need to test for faults at any of the primary inputs to the circuit. Hence, the corresponding test nodes need not be added to the error propagation graphs. The primary test nodes are now associated with the functions corresponding to the outputs of only those components whose inputs are all primary inputs. These outputs will be referred to as primary outputs. Note that the output of a component whose inputs are not all primary inputs is not classified as a primary output, because it will be tested within a path associated with a primary output. For our example, there are only two primary outputs, f_1 and f_2, and the corresponding graphs are shown in Fig. 5.8.

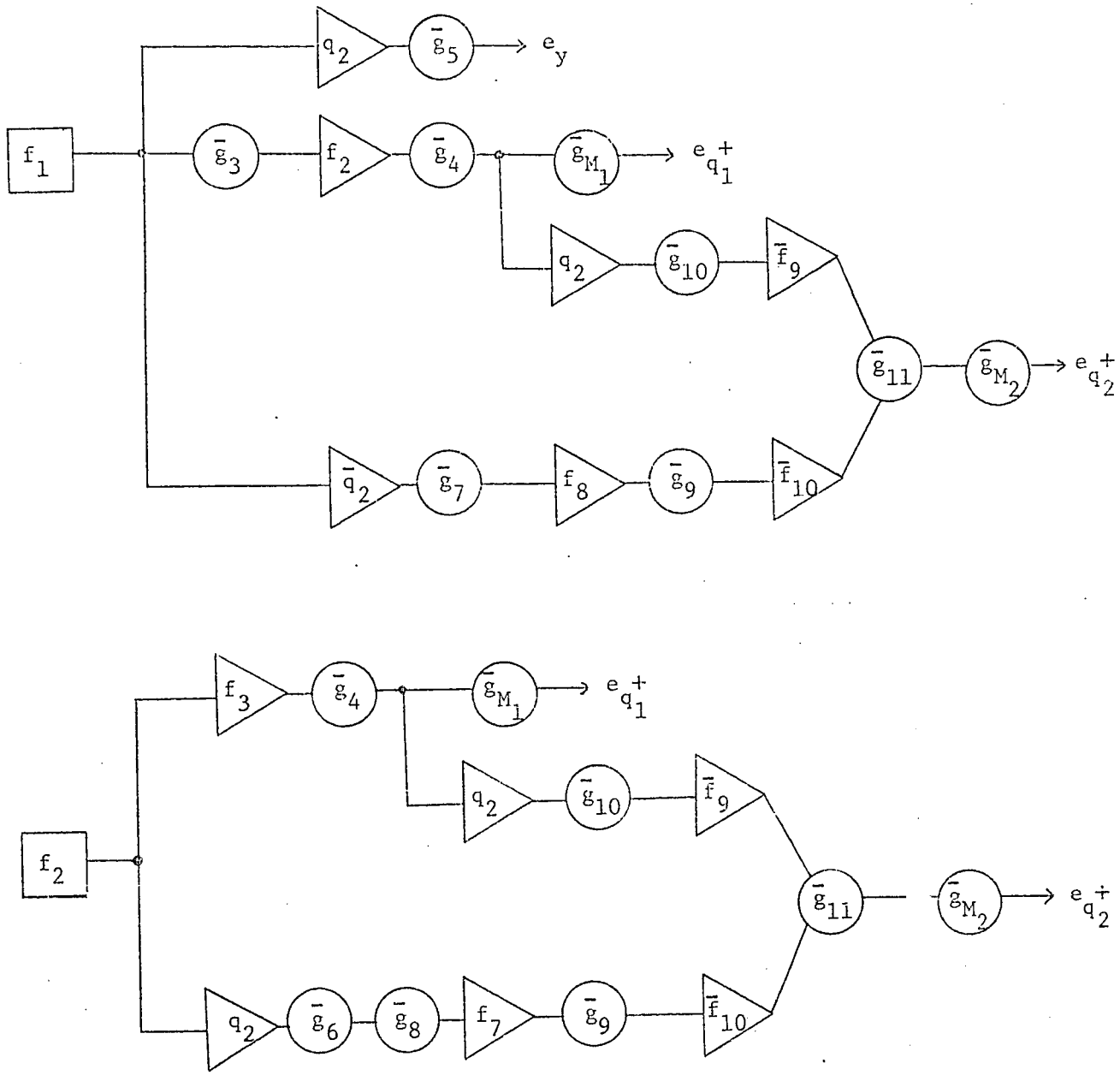


Fig. 5.8

As in the previous section, the first step consists in choosing and testing an appropriate set of primary paths, but in this case, fewer paths need to be chosen. In testing for all faults in the circuit, a primary path had to be selected for each primary input to each component, whereas under the present assumption, a single primary path needs to be chosen for each primary output, regardless of its degree of fanout. The subsequent selection and

testing of these paths is achieved exactly as described in the previous section. For our example, the resulting primary paths and their test tuples are listed in Table 5.5.

PRIMARY PATH	PATH-FUNCTION	s-a-0 TEST	s-a-1 TEST
p_1: $f_1 g_5 y$	q_2	xq_1q_2	$\bar{x}q_1q_2+x\bar{\bar{q}}_1q_2+x\bar{q}_1q_2$
p_2: $f_2 g_4 g_{M_1} q_1^+$	f_3	$x\bar{q}_1q_2+x\bar{q}_1\bar{q}_2+\bar{x}q_1q_2+\bar{x}q_1\bar{q}_2$	$\bar{x}\bar{q}_1q_2+\bar{x}\bar{\bar{q}}_1\bar{q}_2$

Table 5.5

In order to test for all remaining output faults, the procedure remains fairly well the same as in the previous section, i.e. all those paths which originate at fanout points of the circuit and which have not been tested within a primary path must now be considered. However, since none of the component inputs need to be tested, each one of the remaining paths is tested at the output of the first component which immediately follows the fanout point. Again, each one of these test points can be considered as a primary output, and the selection and testing of these remaining paths is achieved as in the case of the above primary paths. For our example, the points that must be tested correspond to outputs f_3, f_6, f_7, and f_{10}. The corresponding paths and their test tuples are listed in Table 5.6.

REMAINING PATHS	PATH-FUNCTION	s-a-0 TEST	s-a-1 TEST
p_3: $f_3 g_4 g_{10} g_{11} g_{M_2} q_2^+$	$f_2 q_2 \bar{f}_9$	$\bar{x}q_1q_2+x\bar{q}_1q_2$	xq_1q_2
p_4: $f_6 g_8 g_9 g_{11} g_{M_2} q_2^+$	$f_7 \bar{f}_{10}$	xq_1q_2	$\bar{x}\bar{\bar{q}}_1q_2+xq_1\bar{q}_2$
p_5: $f_7 g_9 g_{11} g_{M_2} q_2^+$	$f_8 \bar{f}_{10}$	$xq_1\bar{q}_2+\bar{x}\bar{\bar{q}}_1q_2$	$\bar{x}\bar{\bar{q}}_1\bar{q}_2+\bar{x}q_1\bar{q}_2+x\bar{q}_1\bar{q}_2$

Table 5.6

Finally, the set of tests for detecting all output faults is obtained from the test tables, as described in the previous section, i.e. the essential tests are determined first, and a smallest set of non-essential tests is then obtained. For our example, a number of limited test sets can be obtained, each one containing four input tuples. One possible set corresponds to $\{xq_1q_2, xq_1\bar{q}_2, x\bar{q}_1q_2, \bar{x}\bar{q}_1\bar{q}_2\}$; one could also choose the set $\{xq_1q_2, \bar{x}\bar{q}_1q_2, x\bar{q}_1\bar{q}_2, \bar{x}q_1q_2\}$, which is a subset of the complete set obtained in the previous section. Note that, in this case, there is again no guarantee that the smallest test set obtained with the above procedure will be minimal.

For this example, the number of tuples in the complete test set was reduced by only one in testing only for output faults. However, for larger circuits, a significant reduction will be generally obtained, since the number of paths to consider becomes much smaller and the number of input tuples available for testing each path is usually larger.

Case 2: Testing only the outputs with fanout

As a second example, we will show how the error propagation method can be used in generating a set of tests under the assumption that faults can occur only at the outputs with fanout. Note that this second case is an example of selective testing which relates to the structure of the circuit. Physically, one would expect faults at the outputs with fanout to occur more frequently, because the corresponding output components are more heavily loaded than those for outputs with no fanout. Furthermore, it becomes important to test for these specific faults, since they can propagate to more parts of the circuit.

In this case, the path-functions obtained in computing the probabilistic mapping matrix can be used directly for generating a minimal set of tests. With the method of analysis described in Chapter IV, a path-function was obtained for

each output fault (i.e. component malfunction) with respect to each output variable to which it could propagate. Hence, all input tuples that can test for s-a-0 and s-a-1 faults at each output with fanout can be easily determined by considering the sum of all corresponding path-functions. Clearly, this is equivalent to testing all paths through which the output fault can propagate onto an observable output. The minimal set of tests is then obtained by finding the smallest cover of test tuples that will detect all output faults considered.

For our example, only the outputs from components g_1, g_2, and g_4 need to be tested. For each one of these outputs, all input tuples for testing s-a-0 and s-a-1 faults are determined from the corresponding path-functions considered in equations (4.8), (4.9), and (4.10). The resulting test tuples are given in Table 5.7,

OUTPUT	PATH-FUNCTION	s-a-0 TESTS	s-a-1 TESTS
f_1	F_1	$xq_1q_2+x\bar{q}_1\bar{q}_2$	$\bar{x}q_1q_2+\bar{x}q_1\bar{q}_2+\bar{\bar{x}}q_1q_2+$ $\bar{\bar{x}}\bar{q}_1\bar{q}_2+x\bar{q}_1q_2+x\bar{q}_1\bar{q}_2$
f_2	F_2	$x\bar{q}_1q_2+x\bar{q}_1\bar{q}_2+\bar{x}q_1q_2+$ $\bar{x}q_1\bar{q}_2+xq_1q_2$	$\bar{\bar{x}}\bar{q}_1q_2+\bar{\bar{x}}q_1\bar{q}_2$
f_4	F_4	$x\bar{q}_1q_2+x\bar{q}_1\bar{q}_2+$ $\bar{x}q_1q_2+\bar{x}q_1\bar{q}_2$	$xq_1q_2+xq_1\bar{q}_2+$ $\bar{\bar{x}}q_1q_2+\bar{\bar{x}}q_1\bar{q}_2$

Table 5.7

where $F_1 = F_1^1 + F_1^2 + F_1^3 = (x+q_1) + (x+q_1+\bar{q}_2) + q_2 = 1$

$F_2 = F_2^1 + F_2^2 = (\bar{x}+\bar{q}_1) + xq_1q_2$

$F_4 = F_3^1 + F_3^2 = 1 + q_2(x+q_1) = 1.$

Using a fault table similar to Table 5.4, a number of minimal sets can be obtained, each one containing three test tuples. One minimal set can be chosen as $\{xq_1q_2, \bar{x}\bar{q}_1q_2, x\bar{q}_1\bar{q}_2\}$, which corresponds to a subset of the complete test set obtained in Section 3.

5. Conclusion:

The main purpose of this chapter was to suggest procedures whereby the results obtained in computing the probabilistic mapping matrix could be used for generating various sets of fault detection tests. The error propagation method has proved to be a very efficient tool in providing a unified approach to the problem of reliability analysis and fault detection; as was shown in sections 3 and 4, little additional work was needed in extending the results of Chapter IV to the generation of test sets. However, the resulting testing procedures would not be used for fault detection alone, as they are relatively inefficient in dealing only with that problem. Deriving a more efficient procedure for generating minimal sets of tests will require more basic modifications of the error propagation method, and it is therefore suggested as a topic for further research.

Nevertheless, with its use of the error propagation graphs, the method again brings much insight into the problem, as its major advantage lies in the flexible approach that it provides with respect to selective testing. In fact, for the purpose of reducing testing time and expense, the error propagation method can be used in "tailoring" the fault detection procedure to the characteristics of the circuit, whether the corresponding assumptions relate to the technology used in realizing the circuit, to the structure of the circuit, or to the relative reliability of its components.

As an example of a technological assumption, we demonstrated, in Section 4, how the procedure can be easily modified for testing only for output

faults, and why, in most cases, a significant reduction can be obtained in the number of test tuples. With some technologies another possible assumption may correspond to the fact that the faults associated with each type of gate are either s-a-0 or s-a-1, but not both, in which case the error propagation method will again be useful for reducing the number of test tuples.

With respect to the structure of the circuit, the faults corresponding to outputs with high fanout and outputs feeding long circuit lines will be generally assumed more likely to occur. In the previous section, we showed how the error propagation method can be used directly in generating a minimal set of tests for detecting all faults occurring at the outputs with high fan-out. From another point of view, it may also be useful to test only for those faults that propagate most often to the output terminals. These faults can be readily identified with the probabilistic mapping matrix by determining which component malfunctions appear most often in the matrix, and the corresponding set of tests can again be easily obtained with the error propagation method. Similarly, in the case where the simplifying assumption relates to the reliability of the circuit components, the error propagation method can be easily applied in testing only for the faults associated with a determined subset of less reliable components.

Although the above considerations generally apply to circuits operating within a system, the concept of selective testing can also be very useful with respect to testing at the manufacturing stage. A large amount of testing is needed to detect the many faults that occur during the manufacturing process, and to our knowledge, no formal method has been suggested for ordering the tests according to the testing philosophy discussed in this chapter. In this respect, the error propagation method can provide a very flexible procedure for selecting those tests that detect the faults most likely to occur and thereby reduce significantly testing time and cost, especially when the limited test set can be chosen as a subset of the complete test set.

CHAPTER VI

CONCLUSION AND SUGGESTIONS FOR FURTHER RESEARCH

The main purpose of this thesis was to develop an efficient procedure for computing the probabilistic mapping matrix of a sequential circuit whose components have a known probability of malfunctioning. As shown in Chapter IV, the path-sensitizing approach to this problem led to a very convenient graphical representation for the propagation of errors within a circuit, and it thereby provided much insight into the physical nature of the problem. Together with the variety of mathematical tools that were used in developing the procedure, the resulting error propagation method was able to overcome the computational limitations that arose with the matrix and algebraic methods in dealing with larger circuits. Indeed, the increased efficiency of the error propagation procedure was clearly illustrated with respect to the example given in the Appendix.

Furthermore, since the error propagation method is based on the concept of path sensitizing, the results obtained in Chapter IV were easily extended to the detection of faults in combinational circuits. In that respect, the purpose of Chapter V was not so much to develop a very efficient procedure, but rather to investigate some of the problems associated with fault detection and to suggest how the error propagation method could be modified in dealing with these problems. Again, the error propagation graphs emphasized the physical aspects of the problems considered, and a very flexible approach was obtained with respect to the concept of selective testing, i.e. the generation of limited test sets for the detection of subsets of faults, according to technological and structural considerations of the circuit.

Hence, with respect to the general topic of circuit reliability, the most significant advantage of the method of analysis developed in this thesis

is undoubtedly the physical insight that it provides into various related problems. In fact, the error propagation method constitutes an introductory step towards the more challenging problems of fault analysis and the synthesis of more reliable circuits. We will therefore conclude the present study with a few suggestions for further research, and we will show in what way the error propagation method of analysis provides a meaningful approach to each suggested topic.

Topics for Further Reasearch:

A. With respect to the nature and characteristics of the probabilistic mapping matrix:

1) Computation of the reduced probabilistic matrix associated with an asynchronous sequential circuit:

As outlined in Chapter II, the probabilistic mapping matrix is characterized differently with respect to an asynchronous sequential circuit, because the states are not all stable as the "interval" of circuit operation can extend over more than one state transition. However, since the error propagation method provides all the probabilistic information pertaining to each single state transition, the reduced matrix can be computed by extending the results obtained in Chapter IV. Two different approaches can be considered according to the following assumptions:

(i) only a single error may occur during each state transition (whether the states are stable or transient)

(ii) only a single error may occur during a transition between any two stable states.

2) Analysis of the characteristics of the probabilistic mapping matrix in the case where gate delays in a sequential circuit are not negligible with respect to the memory delay:

This characterization is highly dependent upon the structure of the circuit. Since the error propagation graphs indicate which signals propagate to the output and next-state terminals of the circuit, and since they also indicate the number of components through which these signals propagate, the error propagation method should be useful in coping with this problem. Note that this topic is closely related to the concept of sequential circuit realization without memory delay, as in the case of iterative networks.

B. With respect to the computation of the probabilistic mapping matrix:

1) Modular analysis of large sequential circuits:

In this study, we achieved the computation of the probabilistic mapping matrix for a given circuit as a whole. However, when the circuit is very large, such a method of analysis becomes prohibitive, and the circuit must be considered from a modular point of view. In that case, two different approaches are feasible:

(a) Functional analysis of the circuit: this approach consists essentially in extending the error propagation graph developed at the component level to a "functional" graph defined at the level of interconnected circuit modules. In this case, the circuit is assumed to be realized with a number of functional modules, where each module corresponds to some Boolean function. The functional graph is obtained by defining a node for each module and by determining the propagation functions which govern error propagation along the interconnections between modules. The probabilistic mapping matrix corresponding to the entire

circuit could then be computed by extending the error propagation method of analysis to these functional graphs. Note that such an approach should prove to be particularly suitable with respect to integrated circuit realizations.

(b) modular partitioning of the circuit: in the case where a very large circuit is not realized with functional modules as described above, or in the case where the circuit consists of interconnected sequential circuits, a different approach must be adopted by partitioning the circuit into subcircuit modules. The error propagation procedure can be applied to compute the probabilistic mapping matrix of each module, and the overall matrix of the circuit can then be obtained by defining suitable operations on these matrices, according to the interconnections between modules.

2) Reliability analysis with respect to permanent faults:

As outlined in Chapters III and IV, the probabilistic mapping matrix is computed with respect to temporary errors. However, from the point of view of fault detection, more significant information about the reliability of the circuit would be obtained if the analysis was achieved with respect to permanent faults.

The major problem here arises from the fact that temporary errors and permanent faults have different effects on the correct operation of the circuit. Whereas a temporary error may not cause the circuit to operate incorrectly, a permanent fault will eventually lead to erroneous circuit operation. A possible approach to this problem may be to consider time-varying fault probabilities.

C. With respect to fault detection and fault diagnosis:

1) Developing an efficient procedure for fault detection:

The fault detection procedure developed in Chapter V was a simple extension of the results obtained with the error propagation method in computing the probabilistic mapping matrix. As mentioned in the conclusion to that chapter, the procedure may not be very efficient for fault detection alone, and it may not produce minimal sets of tests. However, there is a strong indication that the error propagation method could be further modified in order to derive an efficient procedure for generating minimal sets of tests.

2) Fault diagnosis of circuits:

By contrast with fault detection tests, which only tell us whether the circuit is faulty or failure-free, diagnostic tests provide additional information as to the faults which could be responsible for the occurrence of the detected errors. In computing the entries of the probabilistic mapping matrix, the error propagation method provides such information with the error sets that are obtained for each input tuple applied to the circuit. For large circuits, especially those which are realized with integrated circuits, the diagnostic resolution is generally limited to identifying the faulty module. The error propagation could therefore provide a valuable approach to the problem of fault diagnosis.

D. With respect to the synthesis of reliable circuits:

1) Reliable design of Boolean functions:

In general, a Boolean function can be realized in a number of ways. In analysing the reliability of these implementations with the error propagation method, great physical insight could be obtained towards relating circuit reliability to circuit structure, thereby providing a meaningful approach to the problem of reliable circuit synthesis. Further-

more, the error propagation method will readily identify those components whose faults propagate most often to the output of the circuit. The overall reliability of a Boolean function circuit could therefore be increased by improving the reliability of these components. Note that such a study is particularly significant in view of the growing trend to design circuits using standard subcircuits.

2) Design of circuits that are easy to test:

Again, with the physical insight that it brings into the problem of fault detection, the error propagation method could be very useful in characterizing circuit structures that are easy to test. Furthermore, it could also provide a meaningful approach with respect to the concept of adding special gates and/or special inputs to the circuit for the purpose of improving testing efficiency.

APPENDIX

AN EXAMPLE

The matrix and algebraic methods for computing the probabilistic mapping matrix are illustrated with the example shown in Fig. A.1. The sequential circuit is realized with two-input AND-OR-NOT logic, and its design equations are:

$$y = xq_1q_2$$

$$q_1^+ = x \oplus q_1$$

$$q_2^+ = xq_1\bar{q}_2 + \bar{x}q_1q_2 + \bar{q}_1q_2$$

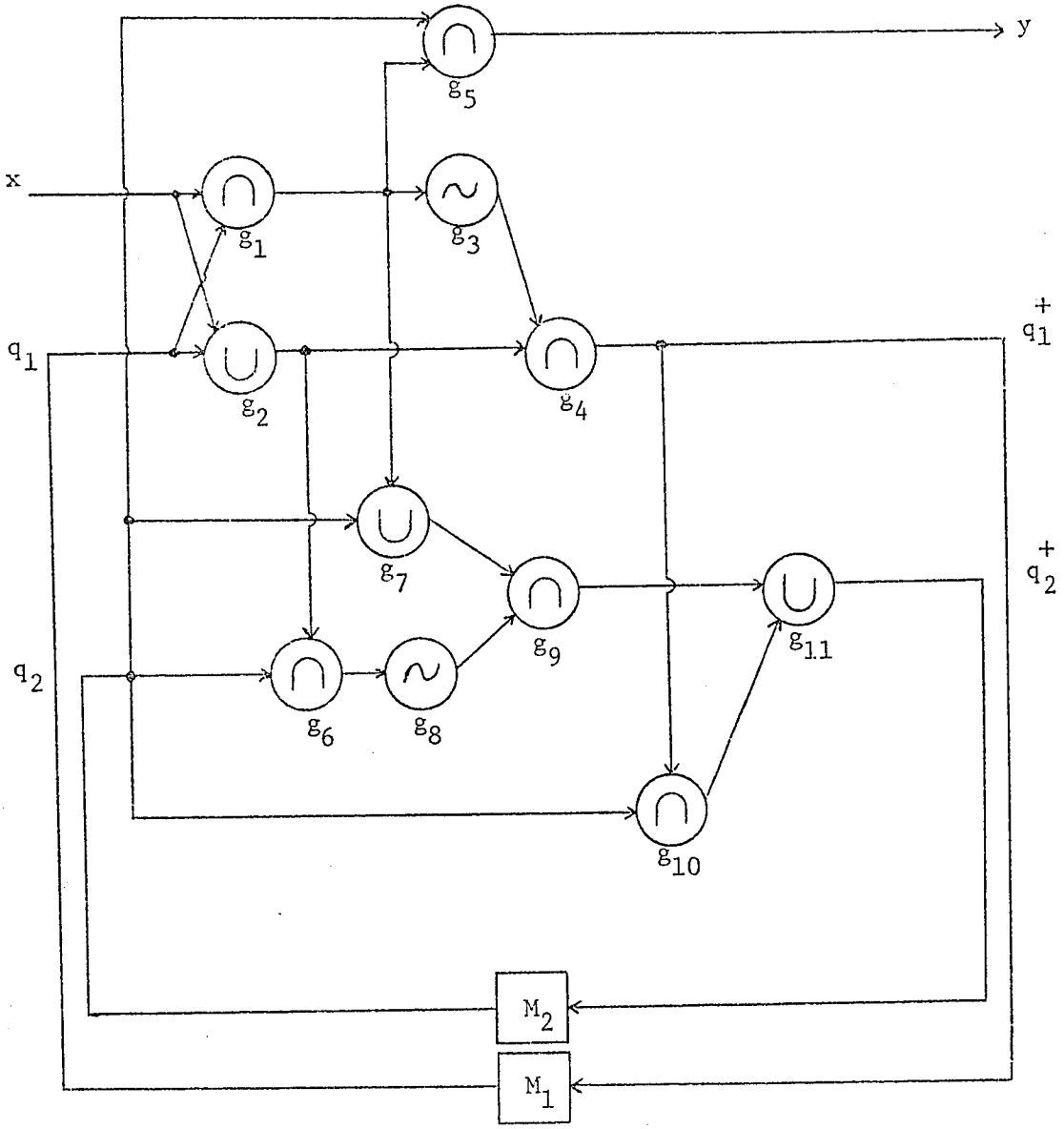


Fig. A.1

1. Computing the probabilistic mapping matrix with the matrix method:

For this example, the series-partitioning method is chosen, because it is more efficient than the parallel-partitioning procedure. Hence, the first step is to partition the circuit into six blocks in series, as shown in Fig. A.2, since the longest path in the circuit contains six components.

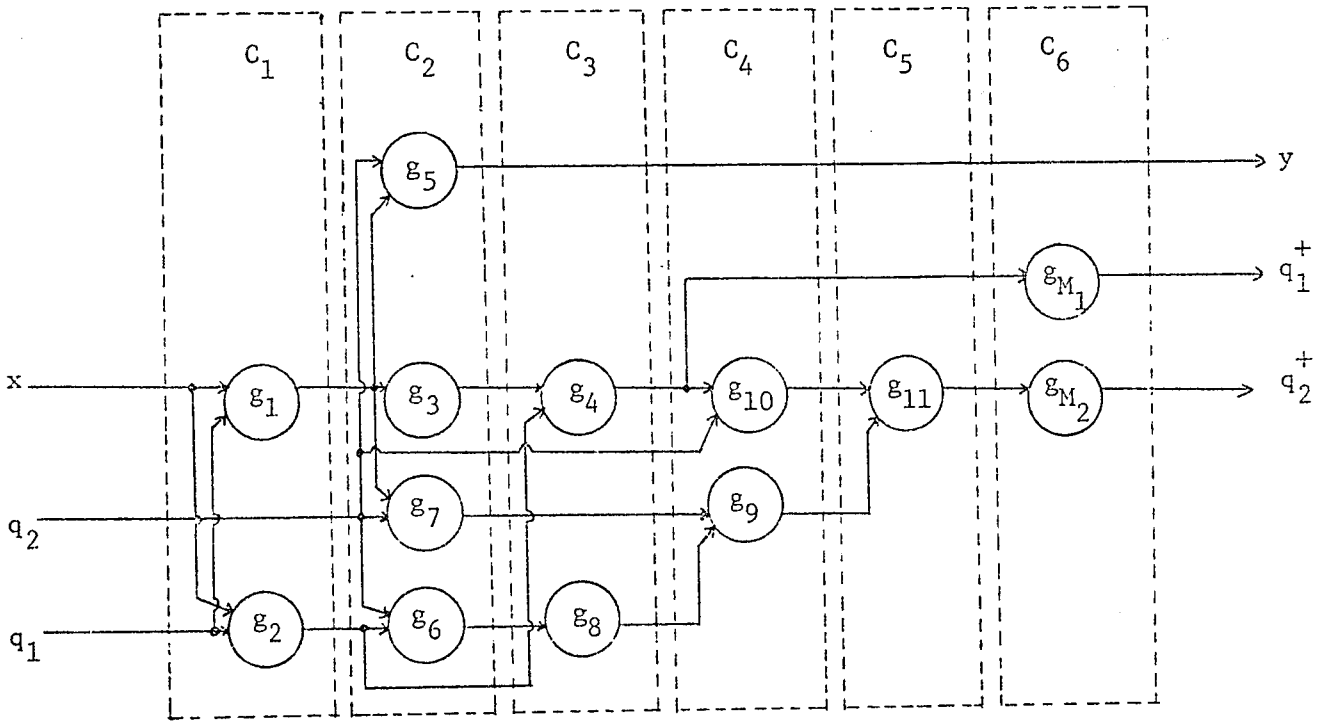


Fig. A.2

The next step is to compute matrix P_i associated with each circuit block C_i ($i = 1, 2, 3, 4, 5, 6$). The computation of each matrix is done by using the matrix operations defined in Chapter III. As an example, let us compute matrix P_1. Assume that the error matrices associated with components g_1 and g_2 are as follows:

$$P_{g_1} = \begin{array}{c|cc} x_1x_2 \backslash f_1 & 0 & 1 \\ \hline 00 & 1-a_0 & a_0 \\ 01 & 1-a_1 & a_1 \\ 10 & 1-a_2 & a_2 \\ 11 & a_3 & 1-a_3 \end{array}$$

$$P_{g_2} = \begin{array}{c|cc} x_1x_2 \backslash f_2 & 0 & 1 \\ \hline 00 & 1-b_0 & b_0 \\ 01 & b_1 & 1-b_1 \\ 10 & b_2 & 1-b_2 \\ 11 & b_3 & 1-b_3 \end{array}$$

The matrix associated with the line corresponding to q_2 is simply given as

$$P_{q_2} = \begin{array}{c|cc} q_2 \backslash q_2 & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}$$. Hence, matrix P_1 is computed as follows:

$$P_1 = (P_{g_1} * P_{g_2}) \otimes P_{q_2},$$

$$= \begin{array}{c|cccc} f_1 f_2 \backslash xq_1 & 00 & 01 & 10 & 11 \\ \hline 00 & 1-a_0-b_0 & b_0 & a_0 & 0 \\ 01 & b_1 & 1-a_1-b_1 & 0 & a_1 \\ 10 & b_2 & 1-a_2-b_2 & 0 & a_2 \\ 11 & 0 & a_3 & b_3 & 1-a_3-b_3 \end{array} \otimes \begin{array}{c|cc} q_2 \backslash q_2 & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}$$

$$= \begin{array}{c|cccccccc} f_1 q f_2 \backslash xq_1 q_2 & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ \hline 000 & 1-a_0-b_0 & b_0 & 0 & 0 & a_0 & 0 & 0 & 0 \\ 001 & 0 & 0 & 1-a_0-b_0 & b_0 & 0 & 0 & a_0 & 0 \\ 010 & b_1 & 1-a_1-b_1 & 0 & 0 & 0 & a_1 & 0 & 0 \\ 011 & 0 & 0 & b_1 & 1-a_1-b_1 & 0 & 0 & 0 & a_1 \\ 100 & b_2 & 1-a_2-b_2 & 0 & 0 & 0 & a_2 & 0 & 0 \\ 101 & 0 & 0 & b_2 & 1-a_2-b_2 & 0 & 0 & 0 & a_2 \\ 110 & 0 & a_3 & 0 & 0 & b_3 & 1-a_3-b_3 & 0 & 0 \\ 111 & 0 & 0 & 0 & a_3 & 0 & 0 & b_3 & 1-a_3-b \end{array}$$

Similarly, one can compute matrices P_2, P_3, P_4, P_5, P_6, where P_2 is an 8x64 matrix, P_3 a 64x32 matrix, P_4 a 32x16 matrix, P_5 a 16x8 matrix, and P_6 an 8x8 matrix. Finally, the P-matrix of the overall circuit is computed by conventional matrix multiplication, i.e. $P = P_1 \cdot P_2 \cdot P_3 \cdot P_4 \cdot P_5 \cdot P_6$

2. Computing the probabilistic mapping matrix with the algebraic method:

The first step is to express variables y, q_1^+, q_2^+ as error functions of the variables x, q_1, q_2 and the variables associated with the components of the circuit. Using the basic output probability functions given in Table 3.3, this first step can be achieved by obtaining an error function for each component output as follows:

$$f_1 = xq_1 \oplus \bar{g}_1$$

$$f_2 = (x+q_1) \oplus \bar{g}_2$$

$$f_3 = f_1 \oplus g_3 = xq_1 \oplus \bar{g}_1 \oplus g_3$$

$$f_4 = f_2 f_3 \oplus g_4 = ((x+q_1) \oplus \bar{g}_2)(xq_1 \oplus \bar{g}_1 \oplus g_3) \oplus \bar{g}_4$$

$$f_5 = f_1 q_2 \oplus \bar{g}_5 = (xq_1 \oplus \bar{g}_1)q_2 \oplus \bar{g}_5$$

$$f_6 = q_2 f_2 \oplus \bar{g}_6 = q_2((x+q_1) \oplus \bar{g}_2) \oplus \bar{g}_6$$

$$f_7 = (q_2+f_1) \oplus \bar{g}_7 = (q_2+(xq_1 \oplus \bar{g}_1)) \oplus \bar{g}_7$$

$$f_8 = f_6 \oplus g_8 = q_2((x+q_1) \oplus \bar{g}_2) \oplus \bar{g}_6 \oplus g_8$$

$$f_9 = f_7 f_8 \oplus \bar{g}_9 = ((q_2+(xq_1 \oplus \bar{g}_1)) \oplus \bar{g}_7)(q_2((x+q_1) \oplus \bar{g}_2) \oplus \bar{g}_6 \oplus g_8) \oplus \bar{g}_9$$

$$f_{10} = q_2 f_4 \oplus \bar{g}_{10} = q_2(((x+q_1) \oplus \bar{g}_2)(xq_1 \oplus \bar{g}_1 \oplus g_3) \oplus \bar{g}_4) \oplus \bar{g}_{10}$$

$$f_{11} = (f_9+f_{10}) \oplus \bar{g}_{11} = (((q_2+(xq_1 \oplus \bar{g}_1)) \oplus \bar{g}_7)(q_2((x+q_1) \oplus \bar{g}_2) \oplus \bar{g}_6 \oplus g_8) \oplus \bar{g}_9)((q_2(((x+q_1) \oplus \bar{g}_2)(xq_1 \oplus \bar{g}_1 \oplus g_3) \oplus \bar{g}_4) \oplus \bar{g}_{10}) \oplus \bar{g}_{11}$$

and $y = f_5$

$$q_1^+ = f_4 \oplus \bar{g}_{M_1}$$

$$q_2^+ = f_{11} \oplus \bar{g}_{M_2}$$

The next step is then to consider the joint probabilities of variables y, q_1^+, and q_2^+. This consists in computing Boolean products of all error functions obtained for these variables. For example, the joint probability associated with the state-output tuple $(q_1^+ , q_2^+ , y) = (1 , 1 , 1)$ corresponds to the Boolean product $f_5(f_4 \oplus \bar{g}_{M_1})(f_{11} \oplus \bar{g}_{M_2})$. Note that functions f_5, f_4,

and f_{11} should be simplified before computing these products.

In order to compute the entries of the probabilistic mapping matrix, the final step consists in evaluating each joint probability function for all values of input-state variables. The resulting expressions will then correspond to sums of products of component variables. Under the assumption that at most one component may malfunction at any time instant, all products involving two or more component variables in complemented form can be discarded. Hence, the matrix entry will be obtained as the sum of probabilities associated with the remaining products of component variables.

REFERENCES

1. T.L. Booth, Sequential Machines and Automata Theory, J. Wiley and Sons, 1967.

2. J.N. Warfield, "Switching networks as models of discrete stochastic processes", in Applied Automata Theory, J.T. Tou (Ed.), Academic Press, pp 81-123, 1968.

3. M.O. Rabin, "Probabilistic automata", Information and Control, vol. 6, pp 230-245, 1963.

4. C.V. Page, "Strong stability problems for probabilistic sequential machines", Information and Control, vol. 15, pp 487-509, Dec. 1969.

5. A. Salomaa, Theory of Automata, Pergamon Press, 1969.

6. J.T. Tou, "Stochastic automata and discrete systems theory", in Applied Automata Theory, J.T. Tou (Ed.), Academic Press, pp 55-80, 1968.

7. W.K. Chung, "Topics in reliability of sequential circuits", Ph.D. dissertation submitted at the Dept. of Elect. Eng., U. of Ottawa, Ottawa, Ont., March 1970.

8. I.N. Chen, "Analysis and reliability estimation for probabilistic switching circuits", IEEE Transactions on Reliability, vol. R-20, no. 1, pp 36-38, Feb. 1971.

9. K. Udagawa and Y. Inagaki, "Probabilistic studies of logical circuits and the synthesis of reliable circuit using probabilistic logical matrix", J. Inst. Elec. Commun. Eng. Jap., vol. 46, no. 11, pp 195-206, Nov. 1963.

10. V.I. Levin, "Probabilistic analysis of combinational systems and their reliability", Eng. Cybernetics, no. 6, pp 78-84, Nov.-Dec. 1964.

11. G.S. Glinski and W.K. Chung, "A Markov model for the reliability of probabilistic sequential circuits with Bernoulli inputs", Proc. 6th Annual Allerton Conf. on Circuit and System Theory, U. of Illinois, pp 568-577, Oct. 1968.

12. G.S. Glinski and W.K. Chung, "A Markov model for the reliability of probabilistic sequential circuits with Markov inputs", Proc. 7th Annual Allerton Conf. on Circuit and System Theory, U. of Illinois, pp 690-691, Oct. 1969.

13. E.F. Moore, "Gedanken-experiments on sequential machines", in Automata Studies, C.E. Shannon and J. McCarthy (Eds.), vol. 34, Princeton Univ. Press. 1956.

14. R.D. Eldred, "Test routines based on symbolic logical statements", J. ACM, vol. 6, no. 1, Jan. 1959.

15. M.A. Arbib, Theories of Abstract Automata, Prentice-Hall, 1969.

16. H.Y. Chang, E.G. Manning, and G. Metze, Fault Diagnosis of Digital Systems, J. Wiley and Sons, 1970.

17. Z. Kohavi, Switching and Finite Automata Theory, McGraw-Hill, 1970.

18. K. To, "Fault folding: a unified approach to fault analysis in combinational logic networks", paper presented at the 1971 IEEE Comp. Conf., Fault Diagnosis Session, Sept. 1971.

19. D.B. Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic nets", IEEE Transactions on Electronic Computers, EC-15, pp 66-73, Feb. 1966.

VITA

Name:		Pierre J. Des Marais
Place of Birth:		Quebec, Quebec, Canada
Date of Birth:		Jan. 9, 1949.
Education:	Secondary	U. of Ottawa High-School
	University	U. of Ottawa
		B.Sc. (Computer Science), 1970.