TOPICS IN RELIABILITY OF SEQUENTIAL CIRCUITS

by

Who-Kee Chung

Submitted to the Department of Electrical Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Department of Electrical Engineering
Faculty of Pure and Applied Science
The University of Ottawa,
Ottawa, Canada.

March 1970

ABSTRACT

In this thesis more emphasis is given on the development of some useful models for reliability of sequential circuits suitable for engineering analysis and design purposes rather than on the development of abstract aspects of reliability theories.

A simplified matrix model of reliability based on Markov chains is developed and then applied to the reliability improvement based on error-correcting codes and redundancy techniques. Finally, a realization problem, best suited for reliability improvement, is considered.

ACKNOWLEDGEMENTS

The author is most grateful to his advisor, Professor G.S. Glinski, for his encouragement, understanding and guidance through out the course of the research.

He is indebted to Professor N.U. Ahmed for stimulating discussions and critical comment on various aspects of this work.

He also wishes to acknowledge the financial assistance of the National Research Council of Canada, and the Department of University affairs of the Province of Ontario.

TABLE OF CONTENTS

CHAPTER I.

INTRODUCTION

Each element in the physical realization of a system has a certain probability of erroneous operation. In 1956 this has been pointed out by von Neumann[1]. By adding redundant elements, he developed a multiplexing scheme for synthesizing reliable automata using unreliable components. In the same year, Moore and Shannon[2] have developed a method of constructing a reliable circuit from less reliable relays. In the past decade, various authors[3,4,5,6,7] have developed different schemes for the improvement of the overall reliability of switching systems. In accordance with the theory of von Neumann, the error probability of switching elements is assumed to be constant in time and independent of input probability distribution. However, not much work has been done on the role and characteristics of errors in the actual operation of the physical switching elements[8,9,10,11,12].

In order to build a switching circuit performing with a specified reliability, the latter must be estimated in advance. It is also necessary to include, in the design method, the actual error behavior of the physical switching components under the operating condition. In 1963 Udagawa, Inagaki[8] and Levin[9] introduced the probabilistic logical matrix (F-matrix, $[P_j]$), representing the probabilistic transformation between the input and output states of a probabilistic switching circuit, as a means of estimating the reliability of combinatorial circuit. A year later Udagawa and Inagaki[13] extended their previous work and introduced the idea of total state transition matrix, $[P_\Omega]$, for the estimation of the reliability of probabilistic sequential circuits, using Markov chain model and assuming that the input is a simple Markov process, The states of $[P_\Omega]$, $\Omega(k)$; k=0,1,..., are the direct product (Kronecker product)[14] of the input n-tuple, X(k), the state of the unit delay elements (assuming perfect)(state r-tuple, S(k)), and output m-tuple, Y(k), i.e.

$$\Omega(k)=X(k) \circledast S(k) \circledast Y(k)$$

Let $p_{i,j}$ be the conditional probability that the Markov chain model will be in state j when the given present state is i. Let u, v, and w be the decimal representations of the present input n-tuple, present state r-tuple and the present output m-tuple respectively. Also let u', v' and w' be the decimal representations of the next input n-tuple, next state r-tuple and next output m-tuple respectively, while X(k), X(k+1), S(k), S(k+1). Y(k), and. Y(k+1) are binary representations. $p_{i,j}$ was given by the following equation (13)

$$p_{i,j} = \frac{P\{Y(k+1)=w'/\ X(k+1)=u' \text{ and } S(k+1)=v'\}\ .\ P\{S(k+1)=v' \text{ and } Y(k)=w/\ X(k)=u \text{ and } S(k)=v\}\ .\ P\{X(k+1)=u'/\ X(k)=u\}}{P\{Y(k)=w/\ X(k)=u \text{ and } S(k)=v\}} \qquad (1.1)$$

where $i=u \times 2^{r+m} + v \times 2^{m} + w$; $j=u' \times 2^{r+m} + v' \times 2^{m} + w'$; $u, u' \in \{0, 1, \ldots, 2^{n}-1\}$; $v, v' \in \{0, 1, \ldots, 2^{r}-1\}$; $w, w' \in \{0, 1, \ldots, 2^{m}-1\}$; n, r and m are the number of the input, the state and the output variables of the sequential circuit respectively. $P\{Y(k+1)=w'/\ X(k+1)=u' \text{ and } S(k+1)=v'\}$ is obtained from the P-matrix representing the probabilistic transformation between the next input-next state and next output. $P\{S(k+1)=v' \text{ and } Y(k)=w/\ X(k)=u \text{ and } S(k)=v\}$ is obtained from the P-matrix representing the probabilistic transformation between the present input-present state and next state-present output. $P\{X(k+1)=u'/\ X(k)=u\}$ is obtained from the P-matrix representing the simple input Markov process. $P\{Y(k)=w/\ X(k)=u \text{ and } S(k)=v\}$ is obtained from the P-matrix representing the probabilistic transformation between the present input-present state and present output.

Since the state of this Markov chain model is the direct product of the input n-tuple, the state r-tuple and output m-tuple, the dimension of $[P_{\Omega}]$ depends on the number of input variables, the number of unit delay elements and the number of output variables. In a binary system, the

dimension of $[P_{\Omega}]$ is $2^{n+r+m} \times 2^{n+r+m}$. From equation (1.1), we note that $p_{i,j}$ cannot be obtained by using matrix operations. Because of the above a better Markov model for the probabilistic sequential circuit with a larger class of inputs is needed to be developed.

A simplified method of deriving the Markov models for the probabilistic sequential circuits with Bernoulli or Markov type of inputs is developed in Chapter IV and Chapter V. The elements of these Markov chain models can be derived by using matrix operations which are introduced in Chapter II, and the dimensions of the transition matrix of these models do not depend on the number of output variables. The dimension of the transition matrix of the Markov chain model for probabilistic sequential circuit with Bernoulli or Markov input is $(2^r+1) \times (2^r+1)$ or $(2^{nq+r}+1) \times (2^{nq+r}+1)$ respectively, where q is the order of the Markov chain input. The main idea of this method is to introduce a new column and a new row, denoted by e(k+1) and e(k) respectively, in the P-matrix of the probabilistic sequential circuit. This new column contains all the probabilities of erroneous state transitions and/or output states of the probabilistic sequential circuit. Then all the columns with headings of next state=v' and present output=w, $w \in \{0,1,\ldots,2^m-1\}$, are combined into one single column with heading of next state=v', $v' \in \{0,1,\ldots,2^r-1\}$. The new row and the new column are used to make the resultant matrix representing the Markovchain model to have an absorbing state. This absorbing state represents the failure state of the class of circuits considered.

Once the Markov chain model for the probabilistic sequential circuit with Bernoulli or Markov input is formed, the mean-time-to-first-error and its variance can be estimated by using the method based on the Markov chain theory [15]. It can also be obtained firstly by transforming the set of difference equations representing the Markov chain model into a set of algebraic equations utilizing the z-transform. Then the signal flow-graph technique is used to obtain the result. This method is developed in Chapter III.

In 1961 Armstrong [16] described a general error-correction procedure which could be used to improve the reliability of the combinatorial circuit part of a sequential circuit. In general any error appearing in the state variable will cause further state and/or output variables in the sequential circuit to be in error. If we do not want any such error to stay in the sequential circuit permanently, we have to correct them. In Chapter VI an algorithm of using error-correcting codes in the secondary state assignments (i.e. enlarging the state space) and output states (i.e. enlarging the output space) is presented. A certain pattern of errors appearing in the state can be prevented from propagation. The enlarged output states are then decoded by a reliable output decoder to extract the correct output states. The reliable output decoder consists of two layers. The first layer of the output decoder, which is used to extract the actual output states from the enlarged output states, consists of (2t+1) copies of decoding circuit. The Hamming distance between each pair of enlarged output states is equal to or greater than 2t+1, where t is the desired number of errors to be corrected. The output variables of the decoding circuits are fed to the majority gates, the second layer of the output decoder. The outputs of the majority gates are the final outputs of the sequential circuit. The probability of error of the resulting sequential circuit is a function of the probability of output variables to be one or zero and the reliability of the last components of the majority gates (assuming that the majority gates are constructed with diode gates) only. If the input diodes of the last components of the majority gates are replaced by t+1 diodes, then the reliability of the resulting sequential circuit can be made equal to one with respect to only t, or less, simultaneous errors occurred in it.

If we relax the reliability requirement, i.e. transient errors (errors which do not stay in the sequential circuit permanently) are allowed to occur it is better to realize every sequential machine by a minimum feedback front machine followed by a maximum feedback-free tail machine. In recent years, Hartmanis and Stearns [17] introduced the so-called

" partition method " and used it to test whether a given sequential circuit can be realized by a feedback-free machine or by a smallest feedback front machine followed by a largest feedback-free tail machine. The resulting machine has the state and output behavior of the given sequential machine. In Chapter VII, a method, successive transition matrices method, is developed for decomposing a given sequential machine into two submachines in series. The first submachine, minimum feedback front machine, contains a minimum number of feedback lines. The second submachine, maximum feedback-free tail machine, does not contain any feedback line.

In order to simplify the problems described, the following assumptions are made:

1. The errors are statistically independent.

2. All errors in the operations of components are transient. i.e. the erroneous operation of any component at time k is expected to operate correctly at time k+h, $h \geq 1$.

3. The component P-matrices are known and time independent.

4. The number of unit delay elements is equal to the number of state variables of the probabilistic sequential circuit. There is only one unit delay element in each feedback line.

5. Input, output and state variables are binary { 0, 1} .

6. Probability distributions of inputs are stationary.

7. The probabilistic sequential circuit operates on a synchronous discrete time scale.

CHAPTER II

FUNDAMENTALS

In this chapter we are going to introduce some material [8,10,11] which will be used in the following chapters.

2.1. Probabilistic Logical Matrix

Consider the combinatorial circuit shown in Fig. 2.1.

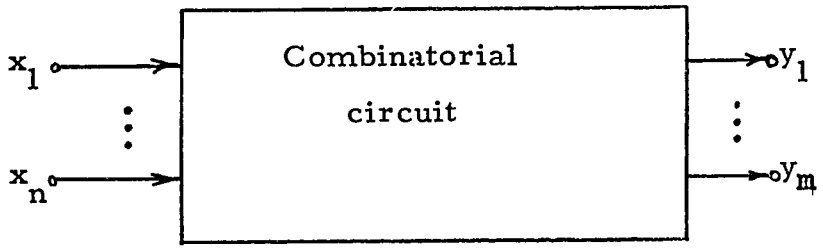

Fig. 2.1. A combinatorial circuit with n input and m output variables.

The probabilistic logical matrix is a transition matrix which represents the probabilistic transformation between the input and the output of a given switching circuit. In binary system, there are 2^n possible input states and 2^m possible output states for such a combinatorial circuit. Let $p_{i,j}$ be the conditional probability that the output state will be j when the input state is i. The collection of values of $p_{i,j}$ ($i=0,1,\ldots,2^n-1$; $j=0,1,\ldots,2^m-1$) determines the probabilistic properties and logical characteristic of the combinatorial circuit.

Let X(k) be the input state at time k, $X(k)=(x_1(k), x_2(k), \ldots, x_n(k))$, and Y(k) be the output state at the same time, $Y(k)=(y_1(k), \ldots, y_m(k))$, where $x_u(k), y_v(k) \in \{0, 1\}$, $u=1,2,\ldots,n$; $v=1,2,\ldots,m$. Arrange the $p_{i,j}$'s into a matrix with rows representing the input states and columns representing the output states as shown in equation (2.1).

$$[P] = \begin{array}{c} \diagdown\, Y(k) \\ X(k) \end{array} \begin{array}{c} 0\ldots0 \\ 0\ldots0 \\ 0\ldots1 \\ \vdots \\ 1\ldots1 \end{array} \begin{array}{c} 0\ldots0 \quad 0\ldots1 \quad \cdots \quad 1\ldots1 \\ \begin{bmatrix} p_{0,0} & p_{0,1} & \cdots & p_{0,2^m-1} \\ p_{1,0} & p_{1,1} & \cdots & p_{1,2^m-1} \\ \vdots & \vdots & & \vdots \\ p_{2^n-1,0} & p_{2^n-1,1} & \cdots & p_{2^n-1,2^m-1} \end{bmatrix} \end{array} \qquad (2.1)$$

This $2^n \times 2^m$ matrix is called probabilistic logical matrix (P-matrix, [P]). It has three properties

i). $1 \geq p_{i,j} \geq 0$ for all $i \in \{0,1,\ldots,2^n-1\}$: $j \in \{0,1,\ldots,2^m-1\}$.

ii). $$\sum_{j=0}^{2^m-1} p_{i,j} = 1 \qquad \text{for all } i.$$

iii). If p_i is the input probability distribution for the input state i; $i \in \{0,1,\ldots,2^n-1\}$, then

$$\sum_{i=o}^{2^n-1} \sum_{j=0}^{2^m-1} p_i \cdot p_{i,j} = 1$$

and $$\sum_{i=0}^{2^n-1} p_i = 1.$$

Now we can extend this idea to the sequential circuit shown in Fig. 2.2. $D_1 \ldots, D_r$ are unit delay elements.

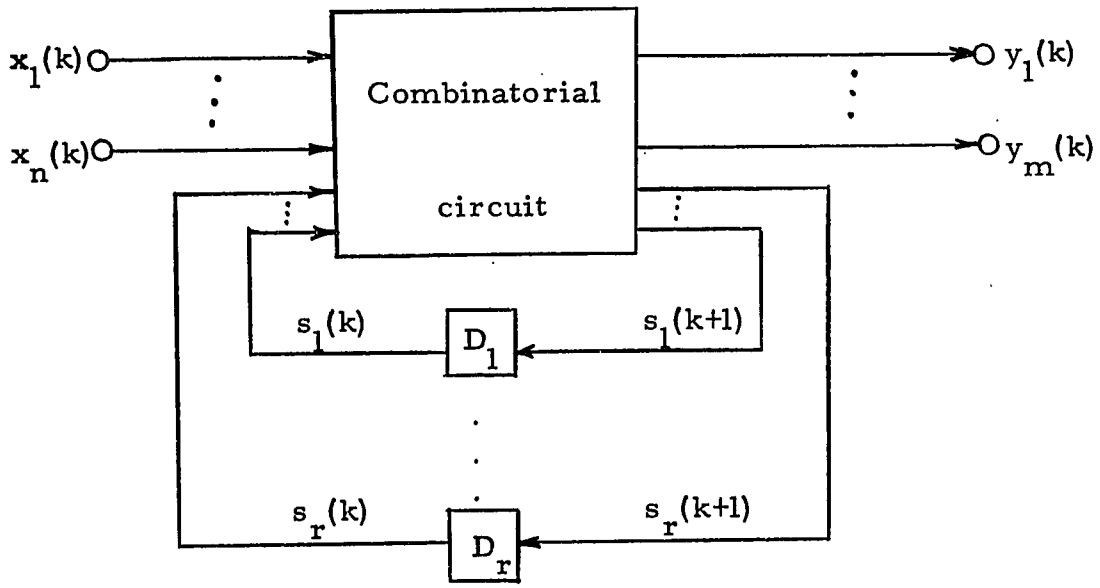

Fig. 2.2. A sequential circuit with n input, m output and r state variables.

Let $S(k+1)=(s_1(k+1),\ldots,s_r(k+1))$ and $S(k)=(s_1(k),\ldots,s_r(k))$ be the state r-tuples at time k+1 and k respectively. Then the P-matrix of the sequential circuit of Fig. 2.2 is

$$[P]= \begin{array}{c|c} S(k+1),Y(k) \;\backslash\; X(k),S(k) & \begin{array}{cccc} 0\ldots0 & 0\ldots1 & \ldots & 1\ldots1 \end{array} \\ \hline \begin{array}{c} 0\ldots0,0\ldots0 \\ 0\ldots0,0\ldots1 \\ \vdots \\ 1\ldots1,1\ldots1 \end{array} & \begin{bmatrix} p_{0,0} & p_{0,1} & \ldots & p_{0,2^{r+m}-1} \\ p_{1,0} & p_{1,1} & \ldots & p_{1,2^{r+m}-1} \\ \vdots & \vdots & \ldots & \vdots \\ p_{2^{n+r}-1,0} & p_{2^{n+r}-1,1} & \ldots & p_{2^{n+r}-1,2^{r+m}-1} \end{bmatrix} \end{array} \qquad (2.2)$$

where $(X(k),S(k))=(x_1(k),\ldots,x_n(k),s_1(k),\ldots,s_r(k))$ and $(S(k+1),\ Y(k))=(s_1(k+1),\ldots,s_r(k+1),y_1(k),\ldots,y_m(k))$

This P-matrix has several properties:

i). $1 \geq p_{i,j} \geq 0$ for all $i \in \{0,1,\ldots,2^{n+r}-1\}$; $j \in \{0,1,\ldots,2^{r+m}-1\}$.

ii).

$$\sum_{j=0}^{2^{r+m}-1} p_{i,j} = 1 \; ; \quad \text{for all } i$$

iii). Let p_u be the probability distribution for the present input n-tuple u and p_v be the probability distribution for the present state r-tuple v. Then

$$p_i = p_u \cdot p_v ; \quad u \in \{0,1,\ldots,2^n-1\}; \; v \in \{0,1,\ldots,2^r-1\}; \; i = u \times 2^r + v.$$

$$\sum_{u=0}^{2^n-1} p_u = 1 \; ; \quad \sum_{v=0}^{2^r-1} p_v = 1 \; ; \quad \sum_{i=0}^{2^{n+r}-1} p_i = 1$$

and

$$\sum_{i=0}^{2^{n+r}-1} \sum_{j=0}^{2^{r+m}-1} p_i \cdot p_{i,j} = 1$$

2.2. Series Combination of Circuits

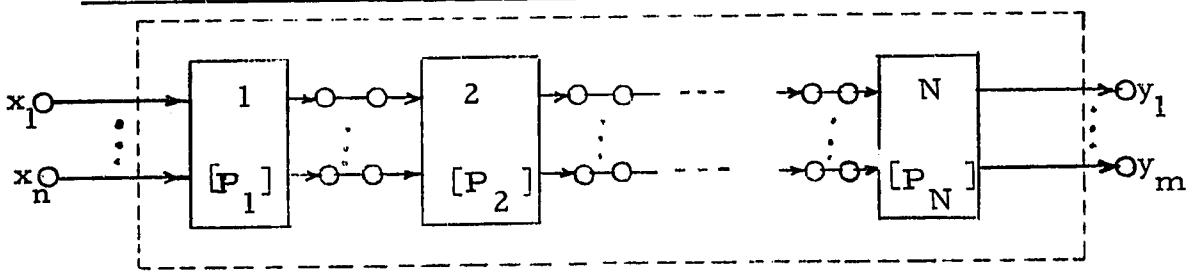

Fig. 2.3. Series combination of N circuits.

The P-matrix of a series combination of N circuits shown in Fig. 2.3 has n input and m output variables and is given by

$$[P] = [P_1] \cdot [P_2] \cdot \ldots \cdot [P_N] \qquad (2.3)$$

provided that the number of the output variables of the (i-1)-th switching circuit equals the number of the input variables of the i-th switching circuit and the column and row headings of $[P_{i-1}]$ and $[P_i]$ respectively are the same, $i=2,3,\ldots,N$. The dimension of the resultant P-matrix, $[P]$, is $2^n \times 2^m$. The operation "." is the conventional matrix

multiplication.

2.3. Parallel Combination of Circuits

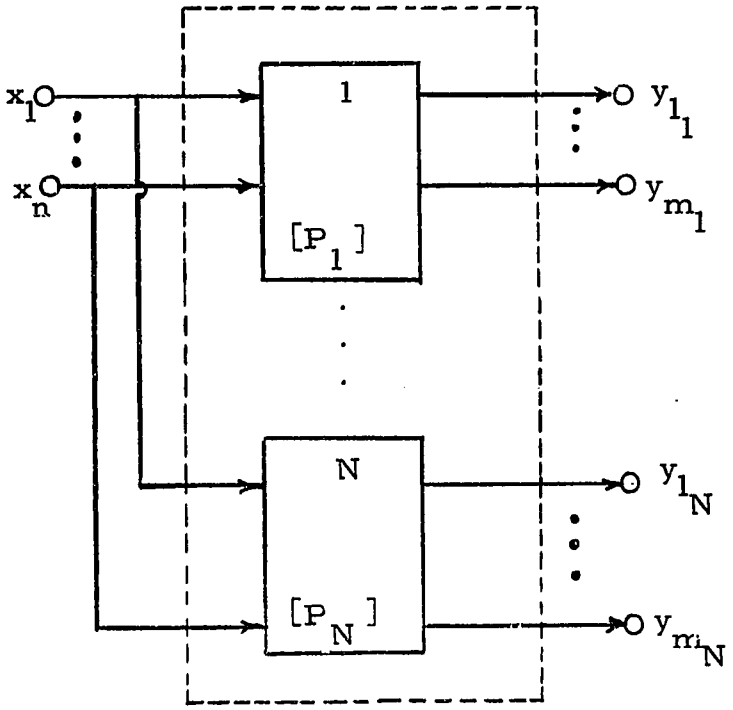

Fig. 2.4. Parallel combination of N circuits.

The P-matrix of a parallel combination of N circuits shown in Fig. 2.4 has n input and ($m_1 + m_2 + \ldots + m_N$) output variables and is given by

$$[P] = [P_1] * [P_2] * \ldots * [P_N] \tag{2.4}$$

where "*" denotes the row multiplication which is defined as

$$p_{i,j} = p\{j/i\} = \prod_{j_k=1}^{N} p_{i,j_k}$$

and $p_{i,j}$ is the (i, j) entry of $[P_{m_k}]$, p_{i,j_k} is the (i, j_k) entry of $[P_k]$
$i \in \{0,1,\ldots 2^n-1\}$ $j_k \in \{0,1,\ldots,2^{m_k}-1\}$ $k \in \{1,2,\ldots,N\}$;

$$j = j_1 \times 2^{m_2+m_3+\ldots+m_N} + \ldots + j_{N-1} \times 2^{m_N} + j_N$$

Example 2.3.1. One OR gate and one AND gate connected in parallel as shown in Fig. 2.5.

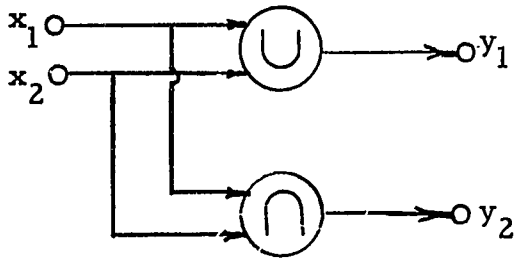

Fig. 2.5. Example 2.3.1.

The P-matrices of OR gate, $[P_U]$, and AND gate, $[P_\cap]$ are given as

$$[P_U] = \begin{array}{c|cc} x_1,x_2 \backslash y_1 & 0 & 1 \\ 0\ 0 & 1-d_0 & d_0 \\ 0\ 1 & d_1 & 1-d_1 \\ 1\ 0 & d_2 & 1-d_2 \\ 1\ 1 & d_3 & 1-d_3 \end{array} \quad ; \quad [P_\cap] = \begin{array}{c|cc} x_1,x_2 \backslash y_2 & 0 & 1 \\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 1 & 1-c_1 & c_1 \\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1 & c_3 & 1-c_3 \end{array} \qquad (2.5)$$

where c_i's, d_i's (i=0,1,2,3) are various error probabilities.

From equation (2.4) we have

$$[P] = [P_U] * [P_\cap]$$

$$= \begin{array}{c|cc} x_1,x_2 \backslash y_1 & 0 & 1 \\ 0\ 0 & 1-d_0 & d_0 \\ 0\ 1 & d_1 & 1-d_1 \\ 1\ 0 & d_2 & 1-d_2 \\ 1\ 1 & d_3 & 1-d_3 \end{array} * \begin{array}{c|cc} x_1,x_2 \backslash y_2 & 0 & 1 \\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 1 & 1-c_1 & c_1 \\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1 & c_3 & 1-c_3 \end{array}$$

$$
= \begin{array}{c|cccc}
x_1,x_2 \backslash y_1,y_2 & 0\,0 & 0\,1 & 1\,0 & 1\,1 \\
\hline
0\,0 & (1-d_0)(1-c_0) & (1-d_0)c_0 & d_0(1-c_0) & d_0 c_0 \\
0\,1 & d_1(1-c_1) & d_1 c_1 & (1-d_1)(1-c_1) & (1-d_1)c_1 \\
1\,0 & d_2(1-c_2) & d_2 c_2 & (1-d_2)(1-c_2) & (1-d_2)c_2 \\
1\,1 & d_3 c_3 & d_3(1-c_3) & (1-d_3)c_3 & (1-d_3)(1-c_3)
\end{array}
$$

Assume that c_i's and d_i's are very small, then the second and higher order product terms of d_i and c_i can be neglected. The P-matrix can be reduced to

$$
[P] = \begin{array}{c|cccc}
x_1,x_2 \backslash y_1,y_2 & 0\,0 & 0\,1 & 1\,0 & 1\,1 \\
\hline
0\,0 & 1-c_0-d_0 & c_0 & d_0 & 0 \\
0\,1 & d_1 & 0 & 1-c_1-d_1 & c_1 \\
1\,0 & d_2 & 0 & 1-c_2-d_2 & c_2 \\
1\,1 & 0 & d_3 & c_3 & 1-c_3-d_3
\end{array}
$$

2.4. " Side by Side" (Juxtaposing) Combination of Circuits

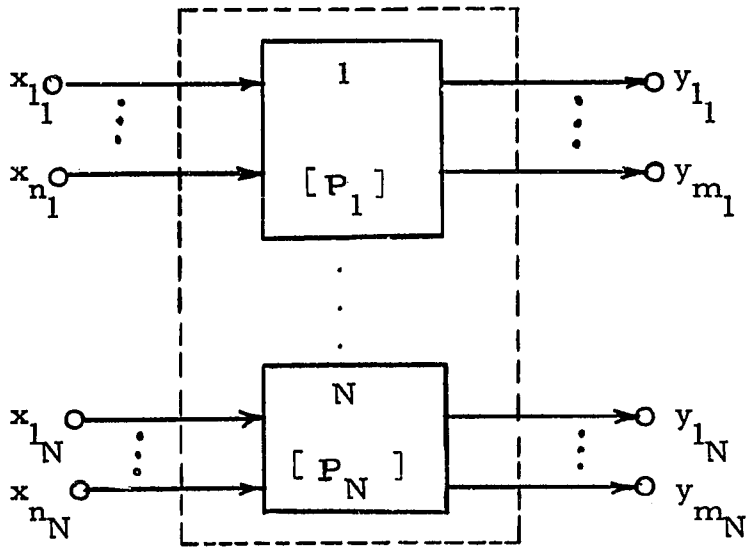

Fig. 2.6. " Side by Side" combination of N circuits

The P-matrix of a " side by side " combination of N circuits shown in Fig. 2.6 has ($n_1 + n_2 + \ldots + n_N$) input and ($m_1 + m_2 + \ldots + m_N$) output variables and is given by

$$[P] = [P_1] \otimes [P_2] \otimes \ldots \otimes [P_N] \qquad (2.6)$$

where the operation " $\otimes$ " denotes Kronecker product, introduced before.

2.5. Enlargement of P-matrix (Mixed Combination of Circuits)

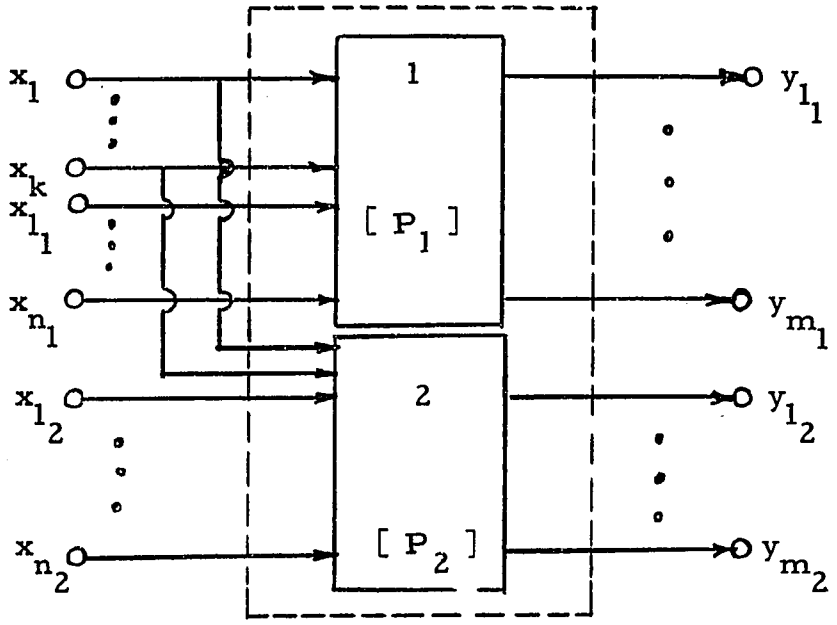

Fig. 2.7. Two switching circuits with some common input variables.

The switching circuits 1 and 2 of Fig. 2.7 are partially " in parallel" and " side by side", thus none of the cperations previously introduced can be used directly. By examining the combined circuit more carefully we find that the parallel operation can be applied if the P-matrices of both circuits 1 and 2, $[P_1]$ and $[P_2]$ respectively, are properly modified. The modified P-matrices will have $2^{k + n_1 + n_2}$ rows and the number of columns is unchanged. The row headings of the modified $[P_1]$ and $[P_2]$ are $X = (x_1, x_2, \ldots, x_k, x_{l_1}, x_{2_1}, \ldots, x_{n_1}, x_{l_2}, \ldots, x_{n_2})$.

The circuits in Fig. 2.7 can be redrawn as shown in Fig. 2.8.

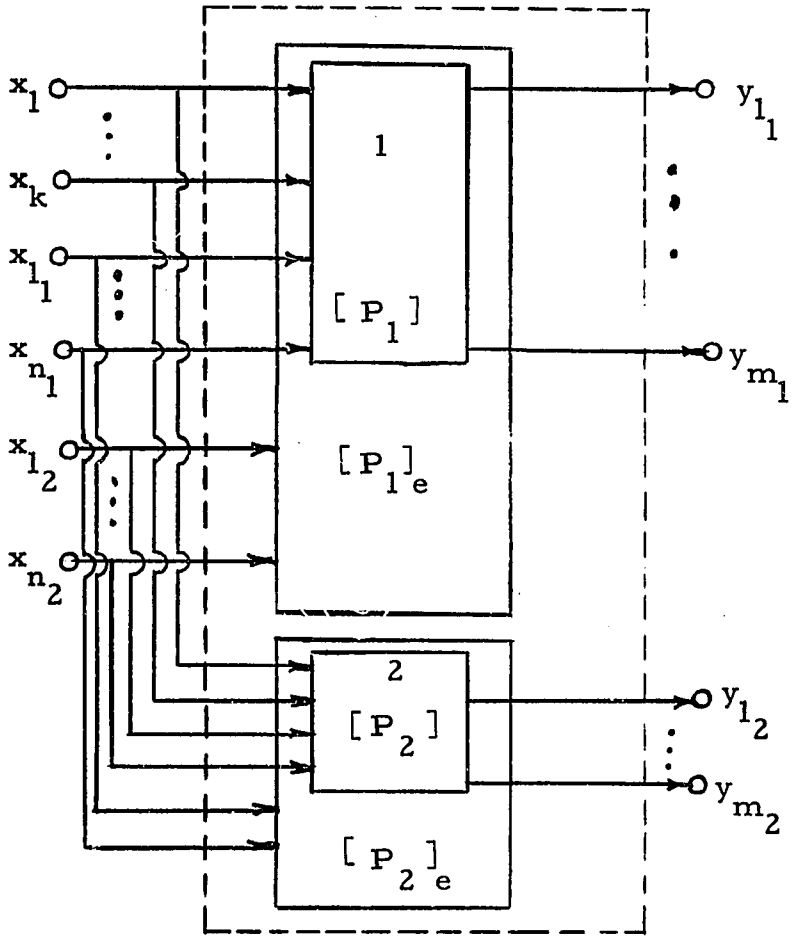

Fig. 2.8. Another circuit arrangement of Fig. 2.7.

From Fig. 2.7 we note that the input variables $x_{1_2}, \ldots,$ and x_{n_2} do not affect the output states of circuit 1. Similarly the input variables $x_{1_1}, \ldots,$ and x_{n_1} do not affect the output states of circuit 2. Thus the entries in $[P_1]_e$, the modified P-matrix of circuit 1, are the same as the entries in $[P_1]$ when the valuations of $x_1, x_2, \ldots, x_k, x_{1_1}, \ldots,$ and x_{n_1} are the same for both $[P_1]$ and $[P_1]_e$. Similarly we can obtain the modified P-matrix of circuit 2, $[P_2]_e$. We call this operation an "enlargement" of P-matrix. Once the enlargement operation is introduced, the P-matrix of the combined circuit in Fig. 2.7 is

$$[P] = [P_1]_e * [P_2]_e \tag{2.7}$$

Let us consider the following example.

Example 2.5.1. The switching circuit is shown in Fig. 2.9.

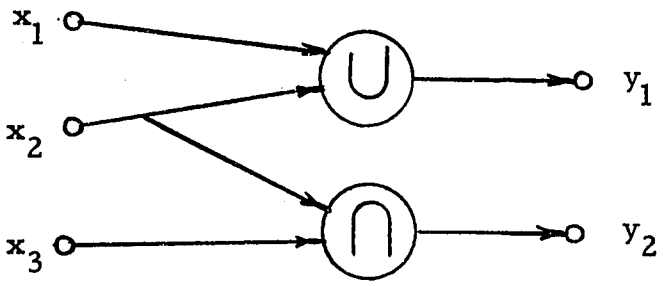

Fig. 2.9. Example 2.5.1.

The OR and And gates in Fig. 2.9 correspond to the circuits 1 and2 in Fig. 2.8 respectively, also $k=n_1=n_2=m_1=m_2=1$.

The P-matrices of OR and AND gates, $[P_U]$ and $[P_\cap]$ respectively, are given as

$$[P_U] = \begin{array}{c|cc} y_1 \,/\, x_1,x_2 & 0 & 1 \\ \hline 0\ 0 & 1-d_0 & d_0 \\ 0\ 1 & d_1 & 1-d_1 \\ 1\ 0 & d_2 & 1-d_2 \\ 1\ 1 & d_3 & 1-d_3 \end{array} ; \quad [P_\cap] = \begin{array}{c|cc} y_2 \,/\, x_2,x_3 & 0 & 1 \\ \hline 0\ 0 & 1-c_0 & c_0 \\ 0\ 1 & 1-c_1 & c_1 \\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1 & c_3 & 1-c_3 \end{array}$$

The enlarged $[P_U]$ and $[P_\cap]$ are

$$[P_{U}]_e = \begin{array}{c|cc} x_1,x_2,x_3 \backslash y_1 & 0 & 1 \\ \hline 0\ 0\ 0 & 1-d_0 & d_0 \\ 0\ 0\ 1 & 1-d_0 & d_0 \\ 0\ 1\ 0 & d_1 & 1-d_1 \\ 0\ 1\ 1 & d_1 & 1-d_1 \\ 1\ 0\ 0 & d_2 & 1-d_2 \\ 1\ 0\ 1 & d_2 & 1-d_2 \\ 1\ 1\ 0 & d_3 & 1-d_3 \\ 1\ 1\ 1 & d_3 & 1-d_3 \end{array} ; \quad [P_{\cap}]_e = \begin{array}{c|cc} x_1,x_2,x_3 \backslash y_2 & 0 & 1 \\ \hline 0\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 0\ 1 & 1-c_1 & c_1 \\ 0\ 1\ 0 & 1-c_2 & c_2 \\ 0\ 1\ 1 & c_3 & 1-c_3 \\ 1\ 0\ 0 & 1-c_0 & c_0 \\ 1\ 0\ 1 & 1-c_1 & c_1 \\ 1\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1\ 1 & c_3 & 1-c_3 \end{array}$$

Assuming, as usual, that c_i's and d_i's ($i = 0, 1, 2, 3$) are very small, the second and higher order product terms of c_i and d_i can be neglected, and the P-matrix of the combined circuit of fig. 2.9 is

$$[P] = [P_{U}]_e * [P_{\cap}]_e$$

$$= \begin{array}{c|cccc} x_1,x_2,x_3 \backslash y_1,y_2 & 0\ 0 & 0\ 1 & 1\ 0 & 1\ 1 \\ \hline 0\ 0\ 0 & 1-c_0-d_0 & c_0 & d_0 & 0 \\ 0\ 0\ 1 & 1-c_1-d_0 & c_1 & d_0 & 0 \\ 0\ 1\ 0 & d_1 & 0 & 1-c_2-d_1 & c_2 \\ 0\ 1\ 1 & 0 & d_1 & c_3 & 1-c_3-d_1 \\ 1\ 0\ 0 & d_2 & 0 & 1-c_0-d_2 & c_0 \\ 1\ 0\ 1 & d_2 & 0 & 1-c_1-d_2 & c_1 \\ 1\ 1\ 0 & d_3 & 0 & 1-c_2-d_3 & c_2 \\ 1\ 1\ 1 & 0 & d_3 & c_3 & 1-c_3-d_3 \end{array}$$

2.6. Probabilistic Sequential Circuits

From the definition of sequential circuit (Mealy's model) [18]:

$$S(k+1) = F(X(k), S(k)) \qquad (2.8)$$

$$Y(k) = G(X(k), S(k)) \qquad (2.9)$$

In a deterministic sequential circuit, there is a one to one mapping between (X(k), S(k)) and (S(k+1), Y(k)). This does not hold for a probabilistic case. The mapping becomes one to many. For a given (X(k), S(k)) there is now a probability p<1 but, for practical case (when the error probability is very small), p ≅ 1 of producing correct (S(k+1), Y(k)), as specified by the equation (2.8) and (2.9).

If the result of the actual operation of the probabilistic sequential circuit differs from the result implied by the equations (2.8) and (2.9), there is an error in the probabilistic sequential circuit.
Reliability of a probabilistic sequential circuit will be considered in terms of the mean-time-to-first-error, T_m, which is the average time for an error to occur for the first time in the probabilistic sequential circuit.

CHAPTER III

A METHOD FOR FINDING THE RELIABILITY OF TIME-DISCRETE SYSTEMS BY USING Z-TRANSFORM AND SIGNAL FLOW-GRAPH TECHNIQUES

In contrast to the Laplace transform and signal flow-graph techniques as applied to continuous systems [19], the z-transform method and signal flow-graph techniques are here used to find the mean-time-to-first-error, T_m, and variance, σ^2, of time-discrete systems [10].

For any time-discrete system which is represented by a q-th order Markov chain ($q \geq 1$), a set of difference equations can be found

$$Q(k+1) = Q(k).[P_T'] \tag{3.1}$$

where $Q(k)=(q_0(k) \quad q_1(k) \quad \ldots \quad q_g(k))$ is a row vector of probabilities of successful states, $[P_T']$ is for transitions between successful states. and $q_i(k)$ is the probability that the system at the k-th time step is in state i; $i \in \{0, 1, \ldots, g\}$; $g+1$ is the number of successful states of the systems.

Since the signal flow-graph technique is applicable only to the solution of linear algebraic equations, in order to apply the signal flow-graph technique to find the reliability of time-discrete system, the set of difference equations must first be transformed into a set of algebraic equations. This can be accomplished by applying z-transform. Taking the z-transform [20] of equation (3.1) we obtain

$$Q(z) = z.Q(z).[P_T'] + Q(0) \tag{3.2}$$

where $Q(z)=(q_0(z) \quad q_1(z) \quad \ldots \quad q_g(z))$, $Q(0)=(q_0(0) \quad q_1(0) \quad \ldots \quad q_g(0))$

is a row vector of initial probabilities, and $q_i(z) = \sum_{k=0}^{\infty} q_i(k) . z^k = \mathcal{Z}(q_i(k))$; $i \in \{0,1,\ldots,g\}$.

Let p(k) be the probability that the system will fail at the k-th time step, then

$$F(m) = \sum_{k=0}^{m} p(k) = \text{probability that the system will fail in the time interval between 0-th and m-th step.}$$

Reliability is the probability of a system performing its purpose adequately for the period of time interval under the operating condition encounted [12, 21]. Thus

$$R(m) = 1-F(m) = \sum_{k=m+1}^{\infty} p(k) = \text{reliability of the system at time m}$$
$$= \sum_{i=0}^{g} q_i(m)$$

From the above relation, we can express the p(k) in terms of R(k), we have

$$p(k) = R(k-1) - R(k)$$

The mean-time-to-first-error, T_m, (first moment of the system) [21, 22] is

$$T_m = \sum_{k=0}^{\infty} k.(R(k-1) - R(k))$$

Let

$$R(z) = \mathcal{Z}(R(k)) = \sum_{k=0}^{\infty} R(k) . z^k = \sum_{k=0}^{\infty} \sum_{i=0}^{g} q_i(k) . z^k = \sum_{i=0}^{g} q_i(z)$$

Since

$$\frac{d}{dz}\left(\sum_{k=0}^{\infty} R(k-1) . z^k - \sum_{k=0}^{\infty} R(k) . z^k \right)\Bigg|_{z=1} = \sum_{k=0}^{\infty} k . R(k-1) - \sum_{k=0}^{\infty} k . R(k)$$

$$= \sum_{k=0}^{\infty} k.(R(k-1) - R(k))$$

then

$$T_m = \frac{d}{dz}(z.R(z) - R(z))\Big|_{z=1} = R(z)\Big|_{z=1} = \sum_{i=0}^{g} q_i(z)\Big|_{z=1} \qquad (3.3)$$

In order to find the variance, the second moment, m^2, of the system must first be found

$$m^2 = \sum_{k=0}^{\infty} k^2.(R(k-1) - R(k))$$

Following the same argument as before, we obtain

$$m^2 = \frac{d^2}{dz^2}(z.R(z) - R(z))\Big|_{z=1} + \frac{d}{dz}(z.R(z) - R(z))\Big|_{z=1}$$

$$= 2.\frac{dR(z)}{dz}\Big|_{z=1} + R(z)\Big|_{z=1} \qquad (3.4)$$

and variance, σ^2, is

$$\sigma^2 = m^2 - T_m^2 \qquad (3.5)$$

We may apply signal flow-graph technique [23] to equation (3.2) with $q_i(0)=1$ and $q_j(0)=0$; $j, i \in \{0,1,\ldots,g\}; j \neq i$, to find $q_\omega(z)/q_i(0)=1$, then T_m's and σ^2's follow from equations (3.3) and (3.5). The symbol ω signifies any of the possible successful (the total number of these is (g+1)) states and $q_\omega(z)/q_i(0)=1$ is the z-transform of the probability of the system being in state ω if it starts at state i.

According to the Mason's rule [23], we have

$$q_i(z) = \frac{\sum_k G_k.\Delta_k}{\Delta} \qquad i : \text{all the successful states}$$

where

$$\Delta = 1 - \sum_m P_{m_1} + \sum_m P_{m_2} - \ldots$$

P_{m_r} = gain product of the m-th possible combination of r nontouching loops.

Δ_k = the value of Δ for that part of the graph not touching the k-th forward path.

G_k = gain of the k-th forward path.

Let us consider a simple example to illustrate the convenience of this method for finding the reliability of probabilistic sequential circuit.

Example 3.1. Consider a time-discrete system with the following transition matrix between successful states, g=1.

$$[P_T'] = \begin{array}{c} S(k+1)\backslash S(k) \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & \quad 1 \end{array} \\ \begin{bmatrix} 0.4 & 0.5 \\ 0.35 & 0.5 \end{bmatrix} \end{array}$$

The state diagram of $[P_T']$ is shown in Fig. 3.1.

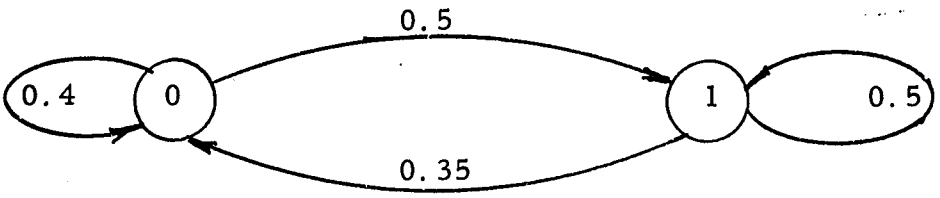

Fig. 3.1. The state diagram of $[P_T']$

The flow-graph of the z-transform of $[P_T']$ is shown in Fig. 3.2.

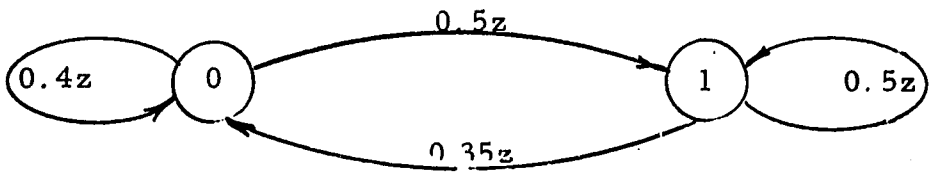

Fig. 3.2. The flow-graph of $[P_T']$.

We now apply the signal flow-graph technique to find T_m's and σ^2's.

$$\Delta = 1 - 0.4z - 0.5z - 0.5 \times 0.35z^2 + 0.4 \times 0.5z^2$$
$$= 1 - 0.9z + 0.025z^2$$

Assuming that the system starts at "0" state, we have

$$q_0(z) = (1-0.5z)/\Delta$$

and $$q_1(z) = 0.5z/\Delta$$

Then we have

$$T_m = (q_0(z) + q_1(z))\Big|_{z=1} = 8$$

$$\frac{dR(z)}{dz}\Big|_{z=1} = \frac{-(-0.9 + 0.05z)}{(1-0.9z + 0.025z^2)^2}\Big|_{z=1} = 54.4$$

and

$$\sigma^2 = 2 \times 54.4 + 8 - 64 = 52.8$$

Assuming that the system starts at "1" state, we have

$$q_0(z) = 0.35z/\Delta$$

and $$q_1(z) = (1-0.4z)/\Delta$$

Then

$$T_m = 7.6$$

and

$$\sigma^2 = 52.4$$

In matrix form, we have

$$T_m = \begin{matrix} S(0) \text{ is } 0 \\ \\ S(0) \text{ is } 1 \end{matrix} \begin{bmatrix} 8 \\ \\ 7.6 \end{bmatrix} \quad \text{and} \quad \sigma^2 = \begin{matrix} 0 \\ \\ 1 \end{matrix} \begin{bmatrix} 52.8 \\ \\ 52.4 \end{bmatrix}$$

CHAPTER IV

A MARKOV MODEL FOR RELIABILITY OF PROBABILISTIC SEQUENTIAL CIRCUITS WITH BERNOULLI INPUTS

The simplest Markov model of probabilistic sequential circuit is obtained when the input sequence can be expressed as Bernoulli process, i.e. the probability of present input n-tuple u does not depend on the past input n-tuples, $u \in \{0, 1, \ldots, 2^n-1\}$, where n is the number of input variables. The dimension of this Markov model is $(2^r + 1) \times (2^r + 1)$ [10], where r is the number of state variables of the probabilistic sequential circuit.

In this chapter we are going to derive, first, the Markov model of the probabilistic sequential circuit with Bernoulli inputs. This will be followed by a calculation of the reliability.

4.1. Methods of Obtaining the P-matrix of Combinatorial Circuit Part of Probabilistic Sequential Circuits.

There are two systematic ways of finding the P-matrix of the combinatorial circuit part of a probabilistic sequential circuit, $[P_c]$.

4.1.1. Method 1---Parallel Partitioning Method

In this method we partition the combinatorial circuit part of the probabilistic sequential circuit into ω blocks, where ω is the total number of output and state variables of the probabilistic sequential circuit (refer to Fig. 2.2; $\omega=m+r$). The i-th block will contain all the components between the present input-present state and the i-th output or next state variable ($i=1, 2, \ldots, \omega$). The procedure of obtaining $[P_c]$ is the following:

Step 1. Assign different subscripts to different error probabilities for every component in the probabilistic sequential circuit. (For example: There are four error probabilities c_0, c_1, c_2, and c_3 in a two inputs AND gate as shown in equation 2.5.)

Step 2. Calculate the P-matrix between the present input-present state and one output or next state variable. There are ω such P-matrices, $[P_1]$, $[P_2]$, ... , and $[P_\omega]$.

Step 3. Since these ω P-matrices have the same row headings, then they are in parallel. We can combine these ω P-matrices with parallel operation. The approximate P-matrix of the combinatorial circuit part of the probabilistic sequential circuit, denoted by $[P_c]_p$, is

$$[P_c]_p = [P_1] * [P_2] * \ldots * [P_\omega]$$

Step 4. If any error probability appears more than once in the entry of the correct transition in any row in $[P_c]_p$, we must make this error probability appear only once in that entry. This error probability must also be deleted from all the other columns. However, it must be rearranged to appear in the proper column (in the same row). The proper column to which this error probability appears is determined by the column of the correct transition and the ones in which it appeared previously. After this modification, $[P_c]_p$ becomes $[P_c]$, the exact P-matrix of the combinatorial circuit part of the probabilistic sequential circuit.

The justification for step 4 is the following: Each input signal passes through each component only once at each time interval. The probability of having an error in the output of each component is, therefore, at the most equal to the error probability of that component for each present input-present state. If the error probability appears in the correct transition entry more than once in any row, this is due to the fact that the component which produces this error probability has more than one, say n'. output lines, where $n' \geq 2$. When we calculate the P-matrices. we consider the component with n' lines as n' identical components. Thus, at the most, there will be n' error probabilities in the

correct transition in any row. Actually there is only one component with n' output lines which, if it is in error, produces error in n', at the most, output and/ or state variables at the same instant of time. If no error probability appears more than once in any entry of correct transition in any row in $[P_{c\,p}]$ then $[P_c]=[P_{c\,p}]$.

Let us consider the following example:

Example 4.1.1.

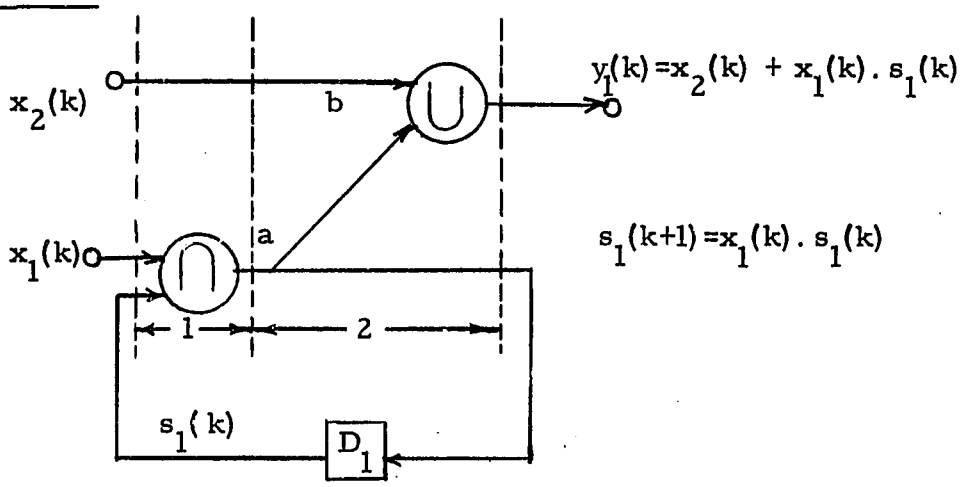

Fig. 4.1. Example 4.1.1.

There are only one output and one state variables in Fig. 4.1, i.e. $\omega=2$. The P-matrices of the AND and OR gates are given as

$$[P_{\cap}] = \begin{array}{cc|cc} a & & 0 & 1 \\ x_1(k), s_1(k) & & & \\ \hline 0 & 0 & 1-c_0 & c_0 \\ 0 & 1 & 1-c_1 & c_1 \\ 1 & 0 & 1-c_2 & c_2 \\ 1 & 1 & c_3 & 1-c_3 \end{array} \quad ; \quad [P_U] = \begin{array}{cc|cc} y_1(k) & & 0 & 1 \\ a, b & & & \\ \hline 0 & 0 & 1-d_0 & d_0 \\ 0 & 1 & d_1 & 1-d_1 \\ 1 & 0 & d_2 & 1-d_2 \\ 1 & 1 & d_3 & 1-d_3 \end{array}$$

The P-matrices of blocks 1 and 2 are

$$[P_1] = \begin{array}{c|cc} x_1(k), s_1(k) \backslash s_1(k+1) & 0 & 1 \\ 0\ 0 & 1-d_0 & d_0 \\ 0\ 1 & d_1 & 1-d_1 \\ 1\ 0 & d_2 & 1-d_2 \\ 1\ 1 & d_3 & 1-d_3 \end{array}_e = \begin{array}{c|cc} x_1(k), x_2(k), s_1(k) \backslash s_1(k+1) & 0 & 1 \\ 0\ 0\ 0 & 1-d_0 & d_0 \\ 0\ 0\ 1 & d_1 & 1-d_1 \\ 0\ 1\ 0 & 1-d_0 & d_0 \\ 0\ 1\ 1 & d_1 & 1-d_1 \\ 1\ 0\ 0 & d_2 & 1-d_2 \\ 1\ 0\ 1 & d_3 & 1-d_3 \\ 1\ 1\ 0 & d_2 & 1-d_2 \\ 1\ 1\ 1 & d_3 & 1-d_3 \end{array}$$

$$[P_2] = \left\{ \begin{array}{c|cc} x_1(k), s_1(k) \backslash a & 0 & 1 \\ 0\ 0 & 1-d_0 & d_0 \\ 0\ 1 & d_1 & 1-d_1 \\ 1\ 0 & d_2 & 1-d_2 \\ 1\ 1 & d_3 & 1-d_3 \end{array}_e \right. * \begin{array}{c|cc} x_2(k) \backslash b & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}_e \left. \bullet \begin{array}{c|cc} a, b \backslash y_1(k) & 0 & 1 \\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 1 & 1-c_1 & c_1 \\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1 & c_3 & 1-c_3 \end{array} \right\}$$

$$= \begin{array}{c|cc} x_1(k), x_2(k), s_1(k) \backslash y_1(k) & 0 & 1 \\ 0\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 0\ 1 & 1-c_2 & c_2 \\ 0\ 1\ 0 & 1-c_1-d_0 & c_1+d_0 \\ 0\ 1\ 1 & d_1+c_3 & 1-c_3-d_1 \\ 1\ 0\ 0 & 1-c_2 & c_2 \\ 1\ 0\ 1 & 1-c_2 & c_2 \\ 1\ 1\ 0 & d_2+c_3 & 1-c_3-d_2 \\ 1\ 1\ 1 & d_3+c_3 & 1-c_3-d_3 \end{array}$$

The approximate P-matrix of the combinatorial circuit part of the probabilistic sequential circuit of Fig. 4.1 is

$$[P_{cp}] = [P_1] * [P_2]$$

$x_1(k), x_2(k), s_1(k)$ \ $s_1(k+1), y_1(k)$	0 0	0 1	1 0	1 1
0 0 0	$1-c_0-d_0$	c_0	d_0	0
0 0 1	d_1	0	$1-c_2-d_1$	c_2
0 1 0	$1-c_1-2d_0$	c_1+d_0	d_0	0
0 1 1	0	d_1	d_1+c_3	$1-c_3-2d_1$
1 0 0	d_2	0	$1-c_2-d_2$	c_2
1 0 1	d_3	0	$1-c_2-d_3$	c_2
1 1 0	0	d_2	d_2+c_3	$1-c_3-2d_2$
1 1 1	0	d_3	d_3+c_3	$1-c_3-2d_3$

Since there are $2d_0$, $2d_1$, $2d_2$, and $2d_3$ in the correct entries in the second, third, sixth and seventh rows respectively, we have to modify this $[P_{cp}]$ to get the exact $[P_c]$. For example, in the second row there is a probability of $1-c_1-2d_0$ in the 0-th column (the entry of correct state transition and correct output state for this row), d_0+c_1 and d_0 appear in the first and second columns respectively. According to step 4, we have to reduce the coefficient of d_0 in the 0-th column (in the second row) to 1, delete d_0 from the first and second columns, and place d_0 in the third column in the same row. Since the column heading for the correct transition is 00, and d_0 appears in columns with headings 01 and 10, (i.e. the erroneous operation of the OR gate, when $x_1(k)=0$, $x_2(k)=1$ and $s_1(k)=0$, will cause both $s_1(k+1)$ and $y_1(k)$ to be in error), thus the d_0 must appear in the third column

instead of appearing in the first and second columns. Since there is only one OR gate which is in error, thus $2d_0$ is reduced to d_0 in the entry of correct state transition and correct output state. Similarly we have to modify the third, the sixth and the seventh rows of $[P_c]_p$. The exact $[P_c]$ is

$[P_c] =$

$x_1(k), x_2(k), s_1(k)$	$s_1(k+1), y_1(k)$ = 0 0	0 1	1 0	1 1
0 0 0	$1-c_0-d_0$	c_0	d_0	0
0 0 1	d_1	0	$1-d_1-c_2$	c_2
0 1 0	$1-c_1-d_0$	c_1	0	d_0
0 1 1	d_1	0	c_3	$1-c_3-d_1$
1 0 0	d_2	0	$1-c_2-d_2$	c_2
1 0 1	d_3	0	$1-c_2-d_3$	c_2
1 1 0	d_2	0	c_3	$1-c_3-d_2$
1 1 1	d_3	0	c_3	$1-c_3-d_3$

4.1.2. Method 2---Series Partitioning Method

Here we partition the combinatorial circuit part of a probabilistic sequential circuit into w blocks in series, where w is the number of components in the longest path between the input-state and the output-next state terminals (refer to Fig. 4.1; w=2, the number of components in the path between $(x_1(k), s_1(k))$ and $y_1(k)$). The procedure of obtaining $[P_c]$ is as follows:

Step 1. Partition the combinatorial circuit into w blocks in series. There is at the most one component in each path in each block. If any path in any block does not contain any component, then an identity matrix of dimension 2 x 2 is inserted.

Step 2. Find the P-matrix of each block by using the parallel,

" side by side" and/or " enlargement" operations. The ordering of the input variables of i-th block must be the same as the ordering of the output variables of (i-1)-th block, i=2, 3, . . . , w . We number the blocks in the ascending order as the direction of the input signal passing through the combinatorial circuit part of the probabilistic sequential circuit. There are w such P-matrices, $[P_1]$, $[P_2]$, . . ., and $[P_w]$.

Step 3. Since these w P-matrices are in series, we can apply the series operation to these P-matrices to get $[P_c]$. Thus

$$[P_c]=[P_1].[P_2]. \ldots .[P_w]$$

Consider the previous example (Example 4.1.1) again. There are two components between the present input-present state and present output terminals, thus w=2. Partition the circuit as shown in Fig. 4.1. The P-matrices of blocks 1 and 2 are

a / $x_1(k), s_1(k)$	0	1
0	$1-d_0$	d_0
1	d_1	$1-d_1$
2	d_2	$1-d_2$
3	d_3	$1-d_3$

$[P_1]=$ *

b / $x_2(k)$	0	1
0	1	0
1	0	1

$x_1(k), x_2(k), s_1(k)$ \ a, b	0 0	0 1	1 0	1 1
0 0 0	$1-d_0$	0	d_0	0
0 0 1	d_1	0	$1-d_1$	0
0 1 0	0	$1-d_0$	0	d_0
0 1 1	0	d_1	0	$1-d_1$
1 0 0	d_2	0	$1-d_2$	0
1 0 1	d_3	0	$1-d_3$	0
1 1 0	0	d_2	0	$1-d_2$
1 1 1	0	d_3	0	$1-d_3$

$$[P_2] = \begin{array}{c|cc} s_1(k+1) \backslash a & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} * \begin{array}{c|cc} y_1(k) \backslash a, b & 0 & 1 \\ 0\,0 & 1-c_0 & c_0 \\ 0\,1 & 1-c_1 & c_1 \\ 1\,0 & 1-c_2 & c_2 \\ 1\,1 & c_3 & 1-c_3 \end{array}$$

$s_1(k+1), y_1(k)$ \ a, b	0 0	0 1	1 0	1 1
0 0	$1-c_0$	c_0	0	0
0 1	$1-c_1$	c_1	0	0
1 0	0	0	$1-c_2$	c_2
1 1	0	0	c_3	$1-c_3$

The P-matrix of the combinatorial circuit part of the probabilistic sequential circuit of Fig. 4.1 is given by

$$[P_c] = [P_1] \cdot [P_2]$$

and the result is the same as the one previously derived by method 1.

Comparing these two methods, we have :

	Method 1	Method 2
Advantage	The dimension of $[P_i]$'s, $i=1,\ldots,\omega$, are relatively small, thus the matrix operations can be easily performed by hand.	The resultant P-matrix, $[P_c]$, is the P-matrix of the combinatorial circuit part of the probabilistic sequential circuit, regardless of whether there is any component which has more than one output line, or not.
Disadvantage	The $[P_c]_p$ may need modification in order to get the exact $[P_c]$, if any component has more than one output line.	The dimension of $[P_i]$'s, $i=1,\ldots,w$, are relatively large when the number of lines in each block is large.

4.2. P-matrix of a Probabilistic Sequential Circuit

Consider the probabilistic sequential circuit shown in Fig. 2.2. Assume that the P-matrices of the combinatorial circuit and unit delay elements, $[P_c]$ and $[P_D]_i$, $i=1,2,\ldots,r$, respectively. are given as follows:

$$[P_{D_i}] = \begin{array}{c|cc} s_i(k) \,/\, s_i(k+1) & 0 & 1 \\ \hline 0 & 1-a_i & a_i \\ 1 & b_i & 1-b_i \end{array}$$

and

$$[P_c] = \begin{array}{c|c} S(k+1), Y(k) \;\diagdown\; X(k), S(k) & \begin{matrix} 0\ldots0,0\ldots0 & \cdots & 1\ldots1,1\ldots1 \end{matrix} \\ \hline \begin{matrix} 0\ldots0,0\ldots0 \\ 0\ldots0,0\ldots1 \\ \vdots \\ 1\ldots1,1\ldots1 \end{matrix} & \begin{bmatrix} p_{0,0} & \cdots & p_{0,2^{r+m}-1} \\ p_{1,0} & \cdots & p_{1,2^{r+m}-1} \\ \vdots & & \vdots \\ p_{2^{n+r}-1,0} & & p_{2^{n+r}-1,2^{r+m}-1} \end{bmatrix} \end{array}$$

The state and block diagrams of the unit delay element are shown in Fig. 4.2a and Fig. 4.2b respectively.

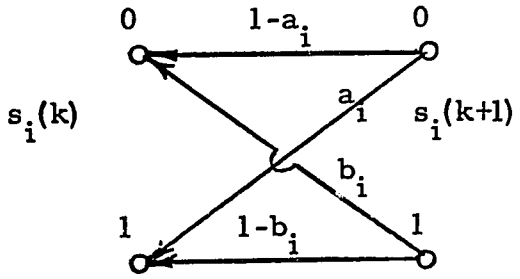

Fig. 4.2a. State diagram of i-th unit delay element.

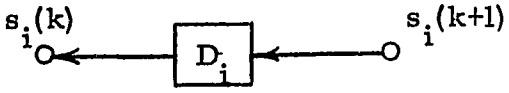

Fig. 4.2b. Block diagram of i-th delay element.

The state diagram of Fig. 4.2a can be decomposed into two parts in series [24]. The first part is the noisy channel with probabilities as in the actual unit delay element, and the second part is the perfect unit delay element. The state diagram of Fig. 4.2a becomes

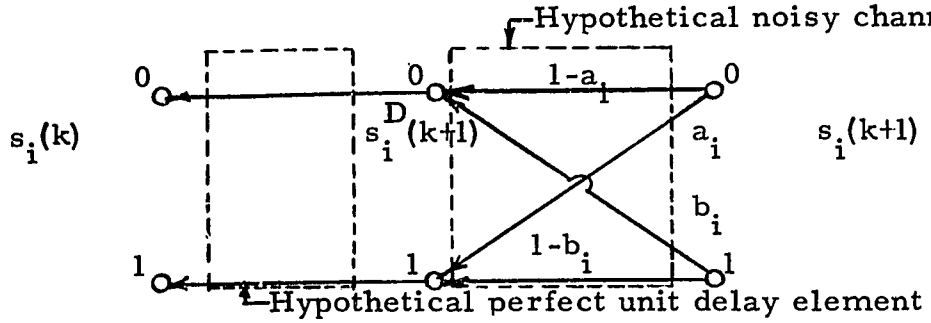

Fig. 4.3. State diagram of i-th unit delay element with two parts in series.

The block diagram of Fig. 4.2b becomes

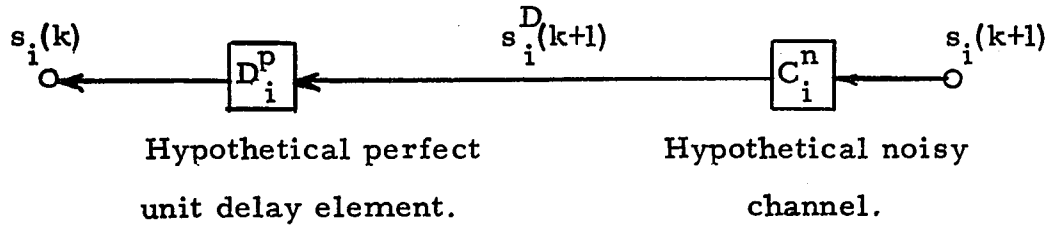

Fig. 4.4. Block diagram of i-th unit delay element with two parts in series.

The $\mathbf{P}$-matrices of the hypothetical noisy channel and the hypothetical perfect unit delay element of the i-th unit delay element are denoted as $[\mathbf{P}_n]_i$ and $[\mathbf{P}_u]_i$ respectively.

$$[\mathbf{P}_n]_i = \begin{array}{c|cc} s_i(k) \backslash s_i^D(k+1) & 0 & 1 \\ \hline 0 & 1-a_i & a_i \\ 1 & b_i & 1-b_i \end{array}$$

and

$$[\mathbf{P}_u]_i = \begin{array}{c|cc} s_i(k) \backslash s_i^D(k+1) & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \qquad i=1,2,\ldots,r.$$

The block diagram of the modified prcbabilistic sequential circuit, of Fig. 2.2, is shown in Fig. 4.5.

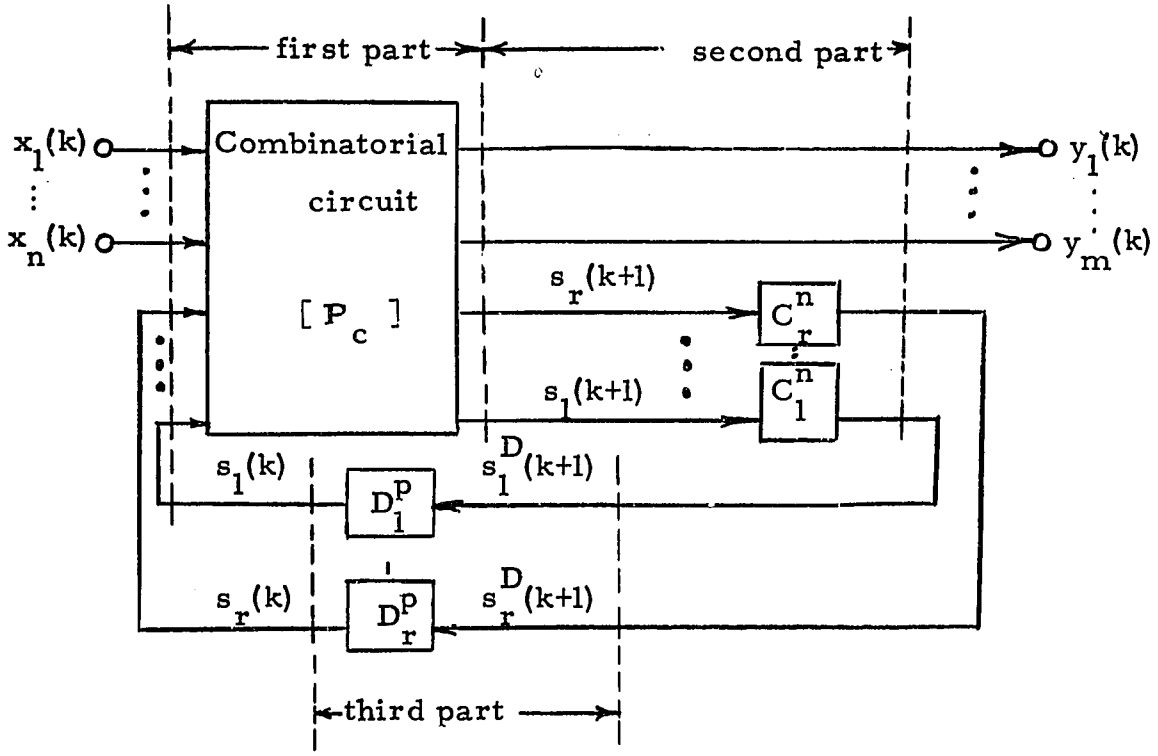

Fig. 4.5. Modified probabilistic sequential circuit.

Now we can say that the probabilistic sequential circuit consists of three parts in series: 1. the combinatorial circuit, 2. the output lines and the hypothetical noisy channels, and, 3. the hypothetical perfect unit delay elements. Thus the P-matrix of the probabilistic sequential circuit, denoted by $[P_s]$, is

$$[P_s] = [P_c].[P_{ny}]$$

provided that the column headings of $[P_c]$ are the same as the row headings of $[P_{ny}]$, where $[P_{ny}]$ denotes the P-matrix obtained from the second part of the probabilistic sequential circuit. Also

$$S^D(k+1) = H(S(k+1)) \tag{4.1}$$

where H is the probabilistic transformation between the S(k+1) and $S^D(k+1)=(s_1^D(k+1), \ldots, s_r^D(k+1))$, and is obtained from the direct product of $[P_{n_i}]$'s, i=1,2,...,r.

4.3. Method of Obtaining the Modified P-matrix of Probabilistic Sequential Circuits

From equations (2.8), (2.9) and (4.1), and the explanation that follows these equations, the following statements are true.

1. The probability of the correct next state r-tuple $S^D(k+1)$ depends on the present input n-tuple and the present state r-tuple only.

2. The error in the actual circuit operation of the probabilistic sequential circuit causes the output m-tuple and/or the next state r-tuple to differ from the answer provided by the equations (2.9) and (4.1) for a given present input n-tuple and present state r-tuple.

We are now in a position to derive the modified P-matrix, denoted by $[P_{ec}]$, of a probabilistic sequential circuit, represented by $[P_s]$. Let the P-matrix of a probabilistic sequential circuit with n=m=r=1 (actually a binary counter) be

$$[P_s]=\begin{array}{c|cccc} X(k),S(k) \backslash S^D(k+1),Y(k) & 0\,0 & 0\,1 & 1\,0 & 1\,1 \\ \hline 0\;0 & \underline{p_{0,0}} & p_{0,1} & p_{0,2} & p_{0,3} \\ 0\;1 & p_{1,0} & p_{1,1} & \underline{p_{1,2}} & p_{1,3} \\ 1\;0 & p_{2,0} & p_{2,1} & \underline{p_{2,2}} & p_{2,3} \\ 1\;1 & p_{3,0} & \underline{p_{3,1}} & p_{3,2} & p_{3,3} \end{array}$$

Assume that the underlined entries are the probabilities for the correct state transitions and correct output m-tuples.

Let us introduce a new column and a new row in the $[P_s]$ with headings e(k+1) and e(k) respectively. Since we are going to find the mean number of times that the probabilistic sequential circuit remains in its correct next state r-tuples and correct present output m-tuples before producing an error, we make $p_{e(k),e(k+1)}=1$, $p_{e(k),j}=0$ and $p_{i,e(k+1)}=r_i$; $j \in \{0,1,\ldots,2^{m+r}-1\}$; $i \in \{0,1,\ldots,2^{n+r}-1\}$. The r_i includes all the probabilities of erroneous state transitions and/or erroneous output m-tuples when $(X(k),S(k))=i$. The probabilities

of erroneous state transitions and/or erroneous output m-tuples, which previously appeared in the entries other than the entries of correct state transitions and correct output m-tuples, must also be dropped. Once the new column and new row are introduced, the $[P_s]$ becomes

$$[P_s]_m = (X(k), S(k))\begin{array}{c} \\ e(k) \\ 00 \\ 01 \\ 10 \\ 11 \end{array}\begin{array}{c} \overbrace{}^{S^D(k+1),\,Y(k)} \\ e(k+1) \quad 00 \quad 01 \quad 10 \quad 11 \end{array}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1-p_{0,0} & p_{0,0} & 0 & 0 & 0 \\ 1-p_{1,2} & 0 & 0 & p_{1,2} & 0 \\ 1-p_{2,2} & 0 & 0 & p_{2,2} & 0 \\ 1-p_{3,1} & 0 & p_{3,1} & 0 & 0 \end{bmatrix}$$

From $[P_s]_m$, we know the probability of having the correct next state r-tuple and the correct present output m-tuple for a given present input n-tuple--present state r-tuple. The correct output m-tuple for a given present input n-tuple--present state r-tuple can be found by using equation (2.9). For each present input n-tuple--present state r-tuple, there is a probability of producing one and only one correct next state r-tuple and present output m-tuple. But the present output m-tuple does not affect the probabilistic sequential circuit at time k+1, k+2,.... Thus we can combine all the columns with headings $(S^D(k+1)=v'$ and $Y(k)=w)$, $w \in \{0,1,\ldots,2^m-1\}$, in $[P_s]_m$ into one column with heading $S^D(k+1)=v'$, $v' \in \{0,1,\ldots,2^r-1\}$. The resulting matrix is $[P_{ec}]$ which has a dimension of $(2^{n+r}+1) \times (2^r+1)$. For the example considered the $[P_{ec}]$ is

$$
[P_{ec}] = (X(k), S(k))
\begin{array}{c}
\\
e(k) \\
0\,0 \\
0\,1 \\
1\,0 \\
1\,1
\end{array}
\begin{array}{c}
\overbrace{}^{S^D(k+1)} \\
e(k+1) \quad 0 \qquad\qquad 1 \\
\left[\begin{array}{lll}
1 & 0 & 0 \\
1-p_{0,0} & p_{0,0} & 0 \\
1-p_{1,2} & 0 & p_{1,2} \\
1-p_{2,2} & 0 & p_{2,2} \\
1-p_{3,1} & p_{3,1} & 0
\end{array}\right]
\end{array}
$$

4.4. Markov Model for Probabilistic Sequential Circuits with Bernoulli Inputs

For a probabilistic sequential circuit as shown in Fig. 2.2 the corresponding P-matrix, $[P_s]$, is a $2^{n+r} \times 2^{r+m}$ matrix

$$
[P_s] = \begin{array}{c}
\\
0\ldots0,0\ldots0 \\
0\ldots0,0\ldots1 \\
\vdots \\
0\ldots0,1\ldots1 \\
0\ldots1,0\ldots0 \\
\vdots \\
0\ldots1,1\ldots1 \\
\vdots \\
1\ldots1,0\ldots0 \\
\vdots \\
1\ldots1,1\ldots1
\end{array}
\begin{array}{c}
\begin{array}{|c|}\hline S^D(k+1), Y(k) \\ \hline X(k), S(k) \\ \hline \end{array} \\
\begin{array}{llll}
0\ldots0,0\ldots0 & 0\ldots0,0\ldots1 & \ldots & 1\ldots1,1\ldots1
\end{array} \\
\left[\begin{array}{llll}
p_{0,0} & p_{0,1} & \ldots & p_{0,2^{r+m}-1} \\
p_{1,0} & p_{1,1} & \ldots & p_{1,2^{r+m}-1} \\
\vdots & \vdots & & \vdots \\
p_{2^r-1,0} & p_{2^r-1,1} & \ldots & p_{2^r-1.2^{r+m}-1} \\
p_{2^r,0} & p_{2^r,1} & \ldots & p_{2^r,2^{r+m}-1} \\
\vdots & \vdots & & \vdots \\
p_{2^{r+1}-1,0} & p_{2^{r+1},1} & \ldots & p_{2^{r+1}-1,2^{r+m}-1} \\
\vdots & \vdots & & \vdots \\
p_{2^{n+r}-2^r,0} & p_{2^{n+r}-2^r,1} & \ldots & p_{2^{n+r}-1.2^{r+m}-1} \\
\vdots & \vdots & & \vdots \\
p_{2^{n+r}-1,0} & p_{2^{n+r}-1,1} & & p_{2^{n+r}-1.2^{r+m}-1}
\end{array}\right]
\end{array}
$$

Assume that the underlined entries in $[P_s]$ correspond to the

probabilities of correct input-state and output-next state transition of the probabilistic sequential circuit. The modified P-matrix of this $[P_s]$ is

$$[P_{ec}] =$$

$e(k)$ \ $e(k+1)$	$S^D(k+1)$: $0\ldots0$	$\ldots$	$1\ldots1$
$0\ldots0,0\ldots0$	1	0	0
	$1-p_{0.1}$	$p_{0,1}$	0
$0\ldots0,0\ldots1$	$1-p_{1,2^{r+m}-1}$	0	$p_{1,2^{r+m}-1}$
$0\ldots0,1\ldots1$	$1-p_{2^r-1,0}$	$p_{2^r-1,0}$	0
$0\ldots1,0\ldots0$	$1-p_{2^r,0}$	$p_{2^r,0}$	0
$0\ldots1,1\ldots1$	$1-p_{2^{r+1}-1,1}$	$p_{2^{r+1}-1,1}$	0
$1\ldots1,0\ldots0$	$1-p_{2^{n+r}-2^r,2^{r+m}-1}$	0	$p_{2^{n+r}-2^r,2^{r+m}-1}$
$1\ldots1,1\ldots1$	$1-p_{2^{n+r}-1,2^{r+m}-1}$	0	$p_{2^{n+r}-1,2^{r+m}-1}$

($X(k), S(k)$)

Since we are dealing with Bernoulli inputs, we decompose the $[P_{ec}]$ into 2^n submatrices, $[P(0)], \ldots, [P(2^n-1)]$ with respect to $X(k)=X=u$ for the convenience of combining them into a single matrix as we shall see later. Each of these submatrices, $P\{e(k+1), S^D(k+1) / e(k), S(k) / X=u\} = [P(u)]$; $u \in \{0,1,\ldots,2^n-1\}$, is a $(2^r+1) \times (2^r+1)$ matrix. Thus

$$[P(0)] =$$

$S(k)$ \ $e(k)$ \ $e(k+1)$	$S^D(k+1)$: $0\ldots0$	$\ldots$	$1\ldots1$
	1	0	0
$0\ldots0$	$1-p_{0,1}$	$p_{0,1}$	0
$0\ldots1$	$1-p_{1,2^{r+m}-1}$	0	$p_{1,2^{r+m}-1}$
$1\ldots1$	$1-p_{2^r-1,0}$	$p_{2^r-1,0}$	0

$$[P(1)] = S(k)\begin{cases} 0\ldots0 \\ \vdots \\ 1\ldots1 \end{cases}\begin{matrix} e(k) \\ \\ \\ \end{matrix}\overset{\overset{e(k+1)}{\overbrace{0\ldots0}}\quad\ldots\quad\overbrace{1\ldots1}^{S^D(k+1)}}{\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 1-p_{2^r,0} & p_{2^r,0} & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 1-p_{2^{r+1}-1,1} & p_{2^{r+1}-1,1} & \ldots & 0 \end{bmatrix}}$$

$$[P(2^n-1)] = S(k)\begin{cases} 0\ldots0 \\ \vdots \\ 1\ldots1 \end{cases}\begin{matrix} e(k) \\ \\ \\ \end{matrix}\overset{\overset{e(k+1)}{\overbrace{0\ldots0}}\quad\ldots\quad\overbrace{1\ldots1}^{S^D(k+1)}}{\begin{bmatrix} 1 & 0 & \ldots & 0 \\ 1-p_{2^{n+r}-2^r,2^{r+m}-1} & 0 & \ldots & p_{2^{n+r}-2^r,2^{r+m}-1} \\ \vdots & \vdots & & \vdots \\ 1-p_{2^{n+r}-1,2^{r+m}-1} & 0 & \ldots & p_{2^{n+r}-1,2^{r+m}-1} \end{bmatrix}}$$

Let $[P_p(u)]$ be the transition matrix of correct operations of the probabilistic sequential circuit corresponding to the input n-tuple u. Also let $[P_e(u)]$ be the transition matrix of erroneous operations of the probabilistic sequential circuit corresponding to the input n-tuple u. The dimensions of $[P_p(u)]$ and $[P_e(u)]$ are $(2^r \times 2^r)$ and $(2^r \times 1)$ respectively.

Since the input n-tuples are mutually independent, on the basis of total probability formula [22, 25, 26], the total transition matrix for the probabilistic sequential circuits with this class of inputs, denoted by $[P_{TB}]$, is

$$[P_{TB}] = \sum_{u=0}^{2^n-1} p_u . [P(u)]$$

$$= S(k) \begin{matrix} \\ 0\ldots0 \\ \vdots \\ 1\ldots1 \end{matrix} \begin{matrix} \overbrace{}^{S^D(k+1)} \\ 0\ldots0 \ldots 1\ldots1 \end{matrix}$$

$$\begin{matrix} e(k+1) \\ e(k) \end{matrix} \begin{bmatrix} 1 & 0 \ldots 0 \\ \begin{bmatrix} \sum_{u=0}^{2^n-1} p_u . [P_e(u)] \end{bmatrix} & \begin{bmatrix} \sum_{u=0}^{2^n-1} p_u . [P_p(u)] \end{bmatrix} \end{bmatrix}$$

$$= S(k) \begin{matrix} \\ 0\ldots0 \\ \vdots \\ 1\ldots1 \end{matrix} \begin{matrix} \overbrace{}^{S^D(k+1)} \\ 0\ldots0 \ldots 1\ldots1 \end{matrix}$$

$$\begin{matrix} e(k+1) \\ e(k) \end{matrix} \begin{bmatrix} 1 & 0 \ldots 0 \\ \begin{bmatrix} R_T \end{bmatrix} & \begin{bmatrix} P_T' \end{bmatrix} \end{bmatrix} \qquad (4.2)$$

where $[R_T] = \sum_{u=0}^{2^n-1} p_u . [P_e(u)]$ and $[P_T'] = \sum_{u=0}^{2^n-1} p_u . [P_p(u)]$

From equation (4.2), we know that $[P_{TB}]$ is a stochastic matrix. The probability that the probabilistic sequential circuit is in state v' at time k+1 is

$$p_{v'}(k+1) = \sum_{u=0}^{2^n-1} \sum_{v=e(k)}^{2^r-1} p_v(k) . p_u . p_{v,v'}(u)$$

where $p_v(k)$ is the probability that the probabilistic sequential circuit is in state v at time k, and $p_{v,v'}(u)$ is the (v, v') entry of $[P(u)]$; $v \in \{e(k), 0, 1, \ldots, 2^r-1\}$ and $v' \in \{e(k+1), 0, 1, \ldots, 2^r-1\}$.

Since the probabilistic sequential circuit has Bernoulli inputs and $p_{v,v'}(u)$ are fully defined by v, v' and u, then from equation (4.2), the state transitions of the probabilistic sequential circuit with this

class of inputs are described by a Markov chain with transition matrix $[P_{TB}]^{(27)}$.

From the above analysis, we have the following theorem:

Theorem 4.1. The behavior of the probabilistic sequential circuit with Bernoulli inputs can be expressed as a Markov chain, and its state transition probability is governed by $[P_{TB}]$. The present states of the Markov chain are e(k) and S(k). The next states are e(k+1) and $S^D(k+1)$, where S(k), $S^D(k+1) \in \{0, 1, \ldots, 2^r - 1\}$.

4.5. Algorithm for Finding the Reliability of Probabilistic Sequential Circuits with Bernoulli Inputs

The flow chart of an algorithm for finding the reliability of probabilistic sequential circuits with Bernoulli inputs is shown in Fig. 4.6. It is a convenient summary of our previous more detailed discussion.

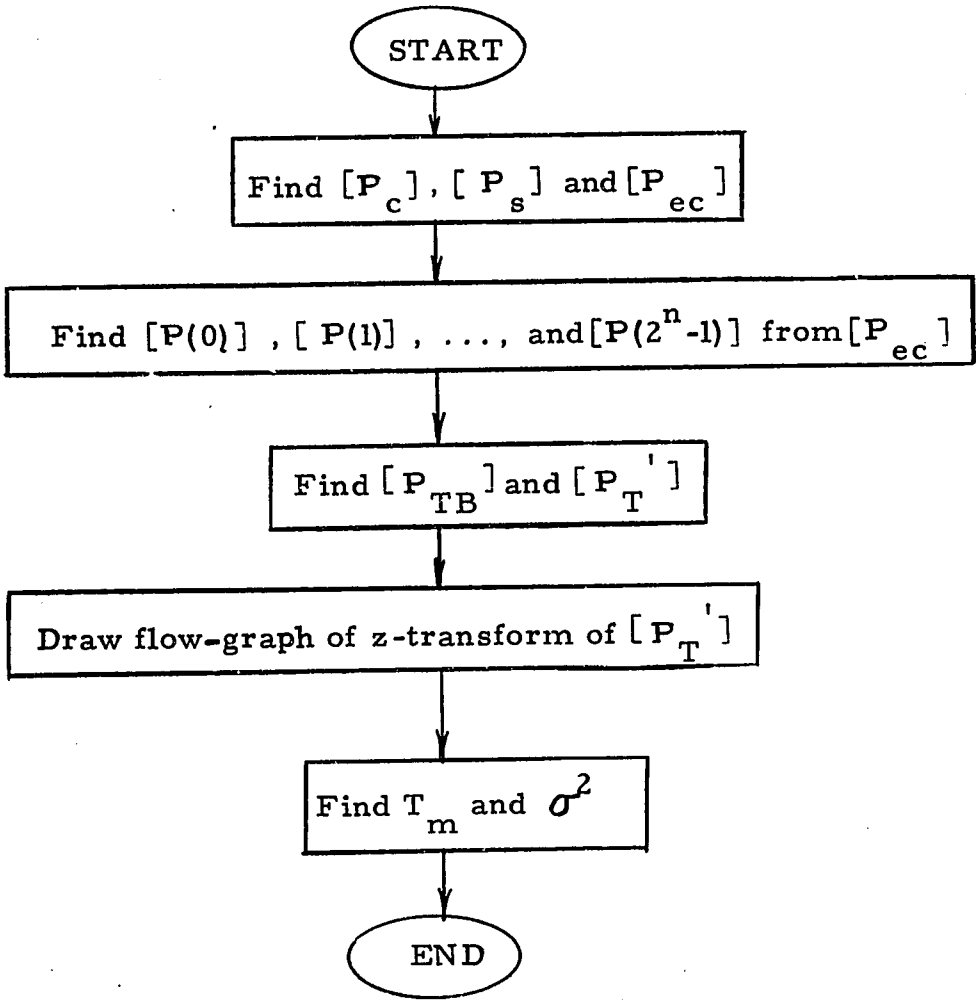

Fig. 4.6. The flow chart of algorithm for finding the reliability of probabilistic sequential circuits with Bernoulli inputs.

4.6. Example

In order to illustrate the method presented in this chapter, an example of a binary counter is considered as shown in Fig. 4.7.

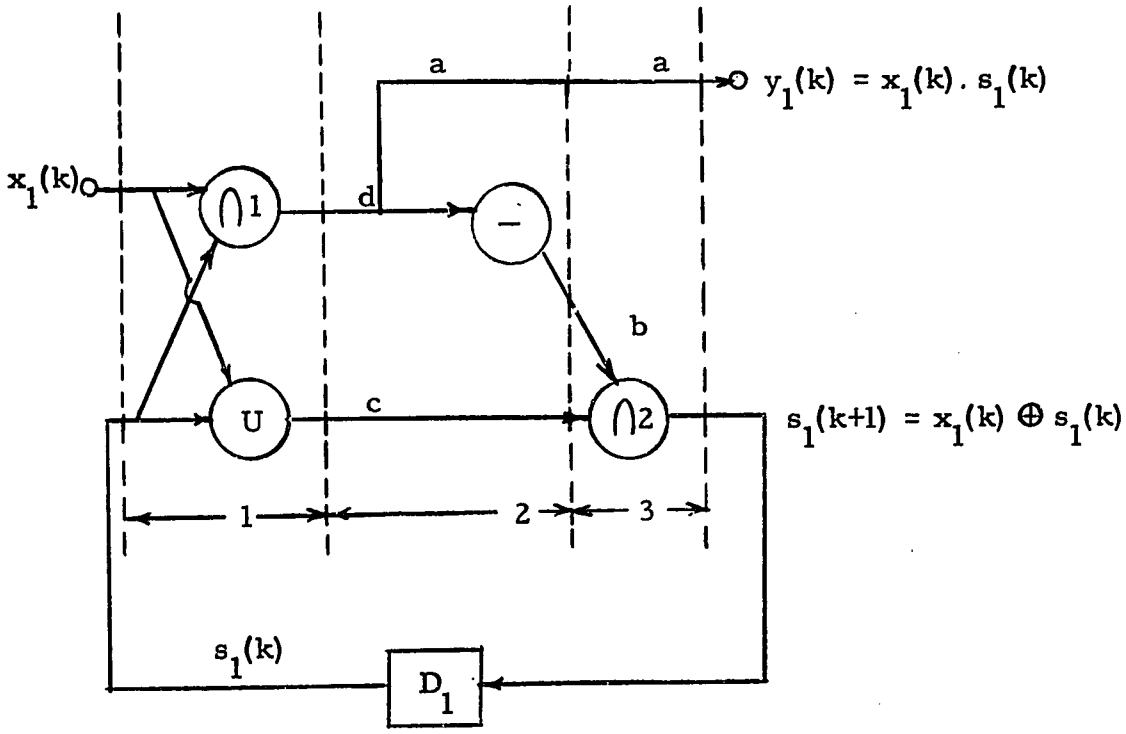

Fig. 4.7. Binary counter.

where the symbol " $\oplus$ " denotes the exclusive-or operation.

Assume that the component P-matrices of AND1 ($\cap 1$), AND2($\cap 2$), OR(U), NOT(−) and unit delay element(D_1), denoted by $[P_{\cap 1}]$, $[P_{\cap 2}]$, $[P_U]$, $[P_-]$and $[P_D]$ respectively, are given as

$$[P_{\cap 1}] = \begin{array}{c|cc} \text{Input} \backslash \text{Output} & 0 & 1 \\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 1 & 1-c_1 & c_1 \\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1 & c_3 & 1-c_3 \end{array} ; \quad [P_{\cap 2}] = \begin{array}{c|cc} \text{Input} \backslash \text{Output} & 0 & 1 \\ 0\ 0 & 1-c_4 & c_4 \\ 0\ 1 & 1-c_5 & c_5 \\ 1\ 0 & 1-c_6 & c_6 \\ 1\ 1 & c_7 & 1-c_7 \end{array}$$

$$[P_U] = \begin{array}{c|cc} \text{Input} \backslash \text{Output} & 0 & 1 \\ \hline 0\ 0 & d_0 & 1-d_0 \\ 0\ 1 & d_1 & 1-d_1 \\ 1\ 0 & d_2 & 1-d_2 \\ 1\ 1 & d_3 & 1-d_3 \end{array} ;$$

$$[P_-] = \begin{array}{c|cc} \text{Input} \backslash \text{Output} & 0 & 1 \\ \hline 0 & e_0 & 1-e_0 \\ 1 & 1-e_1 & e_1 \end{array}$$

and

$$[P_D] = \begin{array}{c|cc} s_1(k) \backslash s_1(k+1) & 0 & 1 \\ \hline 0 & 1-a_1 & a_1 \\ 1 & b_1 & 1-b_1 \end{array} = [P_n]$$

Since there is only one input variable, one state variable and one output variable in this example, then $S(k+1)=s_1(k+1)$, $S(k)=s_1(k)$. $X(k)=x_1(k)$ and $Y(k)=y_1(k)$. Assume that c_i's, d_i's, e_i's. where i ranges from 0 to 7, a_1 and b_1 are very small, so that the second and higher order product terms of c_i, d_i, e_i, a_1 and b_1 can be neglected. We will determine $[P_c]$ by series partitioning method (see Section 4.1.2).

Since there are three components in the longest path, the path between $(X(k), S(k))$ and $s_1(k+1)$ terminals, in Fig. 4.7, w is equal to 3. We partition the combinatorial circuit part of Fig. 4.7 into three blocks in series as shown by the dotted lines . Then

$$[P_1] = \begin{array}{c|cc} X(k),S(k) \backslash d & 0 & 1 \\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 1 & 1-c_1 & c_1 \\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1 & c_3 & 1-c_3 \end{array} * \begin{array}{c|cc} X(k),S(k) \backslash c & 0 & 1 \\ 0\ 0 & 1-d_0 & d_0 \\ 0\ 1 & d_1 & 1-d_1 \\ 1\ 0 & d_2 & 1-d_2 \\ 1\ 1 & d_3 & 1-d_3 \end{array}$$

$$= \begin{array}{c|cccc} X(k),S(k) \backslash d,c & 0\ 0 & 0\ 1 & 1\ 0 & 1\ 1 \\ 0\ 0 & 1-c_0-d_0 & d_0 & c_0 & 0 \\ 0\ 1 & d_1 & 1-c_1-d_1 & 0 & c_1 \\ 1\ 0 & d_2 & 1-c_2-d_2 & 0 & c_2 \\ 1\ 1 & 0 & c_3 & d_3 & 1-c_3-d_3 \end{array}$$

$$[P_2] = \left(\begin{array}{c|cc} b \backslash d & 0 & 1 \\ 0 & e_0 & 1-e_0 \\ 1 & 1-e_1 & e_1 \end{array} \otimes \begin{array}{c|cc} c \backslash c & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \right) * \begin{array}{c|cc} a \backslash d & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}_e$$

$$= \begin{array}{c|cccccccc} d,c \backslash b,c,a & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\ 0\ 0 & e_0 & 0 & 0 & 0 & 1-e_0 & 0 & 0 & 0 \\ 0\ 1 & 0 & 0 & e_0 & 0 & 0 & 0 & 1-e_0 & 0 \\ 1\ 0 & 0 & 1-e_1 & 0 & 0 & 0 & e_1 & 0 & 0 \\ 1\ 1 & 0 & 0 & 0 & 1-e_1 & 0 & 0 & 0 & e_1 \end{array}$$

and

$$[P_3] = \begin{array}{c|cc} S(k+1) \backslash b,c & 0 & 1 \\ \hline 0\ 0 & 1-c_4 & c_4 \\ 0\ 1 & 1-c_5 & c_5 \\ 1\ 0 & 1-c_6 & c_6 \\ 1\ 0 & c_7 & 1-c_7 \end{array} \otimes \begin{array}{c|cc} Y(k) \backslash a & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}$$

$$= \begin{array}{c|cccc} S(k+1),Y(k) \backslash b,c,a & 0\ 0 & 0\ 1 & 1\ 0 & 1\ 1 \\ \hline 0\ 0\ 0 & 1-c_4 & 0 & c_4 & 0 \\ 0\ 0\ 1 & 0 & 1-c_4 & 0 & c_4 \\ 0\ 1\ 0 & 1-c_4 & 0 & c_5 & 0 \\ 0\ 1\ 1 & 0 & 1-c_5 & 0 & c_5 \\ 1\ 0\ 0 & 1-c_6 & 0 & c_6 & 0 \\ 1\ 0\ 1 & 0 & 1-c_6 & 0 & c_6 \\ 1\ 1\ 0 & c_7 & 0 & 1-c_7 & 0 \\ 1\ 1\ 1 & 0 & c_7 & 0 & 1-c_7 \end{array}$$

Then the P-matrix of the combinatorial circuit part of the binary counter is

$$[P_c] = [P_1] . [P_2] . [P_3]$$

S(k+1), Y(k) / X(k), S(k)	0 0	0 1	1 0	1 1
0 0	$1-c_0-c_6-d_0$	c_0	d_0+c_6	0
0 1	$e_0+d_1+c_7$	c_1	$1-c_1-c_7-d_1-e_0$	0
1 0	$e_0+d_2+c_7$	c_2	$1-c_2-c_7-d_2-e_0$	0
1 1	0	$1-c_3-c_5-e_1$	c_3	c_5+e_1

Thus the P-matrix of the binary counter is

$$[P_s]=[P_c]\cdot[P_n]\otimes\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

S^D(k+1), Y(k) / X(k), S(k)	0 0	0 1	1 0	1 1
0 0	$1-a_1-c_0-c_6-d_0$	c_0	$d_0+c_6+a_1$	0
0 1	$e_0+d_1+c_7+b_1$	c_1	$1-e_0-c_1-c_7-d_1-b_1$	0
1 0	$e_0+d_2+c_7+b_1$	c_2	$1-e_0-c_2-c_7-d_2-b_1$	0
1 1	0	$1-a_1-c_3-c_5-e_1$	c_3	$e_1+c_5+a_1$

Assume that $c_i=d_i=e_i=b_1=a_1=a$, $i=0,1,\ldots,7$. If we compare the entries of $[P_s]$ with the switching equations of the binary counter, we find that the modified P-matrix of this binary counter is

$$
[P_{ec}] = (X(k), S(k)) \begin{array}{c} \\ e(k) \\ \left\{ \begin{array}{l} 0\ 0 \\ 0\ 1 \\ 1\ 0 \\ 1\ 1 \end{array} \right. \end{array} \begin{array}{c} \overbrace{\begin{array}{ccc} e(k+1) & & S^D(k+1) \\ & 0 & 1 \end{array}} \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 4a & 1-4a & 0 \\ 5a & 0 & 1-5a \\ 5a & 0 & 1-5a \\ 4a & 1-4a & 0 \end{array} \right] \end{array} \qquad (4.3)
$$

Since there is only one input variable, and it is assumed that $p(X=0)=p_0$, $p(X=1)=p_1$ and $p_0+p_1=1$, the two submatrices are found to be

$$
[P(0)] = S(k) \begin{array}{c} \\ \left\{ \begin{array}{l} 0 \\ 1 \end{array} \right. \end{array} \begin{array}{c} \overbrace{\begin{array}{ccc} e(k+1) & & S^D(k+1) \\ & 0 & 1 \end{array}} \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 4a & 1-4a & 0 \\ 5a & 0 & 1-5a \end{array} \right] \end{array}
$$

$$
[P(1)] = S(k) \begin{array}{c} \\ \left\{ \begin{array}{l} 0 \\ 1 \end{array} \right. \end{array} \begin{array}{c} \overbrace{\begin{array}{ccc} e(k+1) & & S^D(k+1) \\ & 0 & 1 \end{array}} \\ \left[\begin{array}{ccc} 1 & 0 & 0 \\ 5a & 0 & 1-5a \\ 4a & 1-4a & 0 \end{array} \right] \end{array}
$$

Substitution of the above two matrices into equation (4.2) yields

$$
[P_{TB}] = S(k)
\begin{array}{c}
\\ e(k) \\ 0 \\ 1
\end{array}
\begin{array}{c}
\overbrace{\qquad\qquad}^{S^D(k+1)} \\
e(k+1) \qquad\quad 0 \qquad\qquad 1
\end{array}
\begin{bmatrix}
1 & 0 & 0 \\
4ap_0+5ap_1 & (1-4a)p_0 & (1-5a)p_1 \\
5ap_0+4ap_1 & (1-4a)p_1 & (1-5a)p_0
\end{bmatrix}
$$

and the matrix of correct state transitions and output states is

$$
[P_T'] = \begin{array}{c} \boxed{S(k) \;\backslash\; S^D(k+1)} \\ 0 \\ 1 \end{array}
\begin{array}{c} \\ 0 \qquad\qquad 1 \end{array}
\begin{bmatrix}
(1-4a)p_0 & (1-5a)p_1 \\
(1-4a)p_1 & (1-5a)p_0
\end{bmatrix}
$$

The flow-graph of the z-transform of $[P_T']$ is shown in Fig. 4.8.

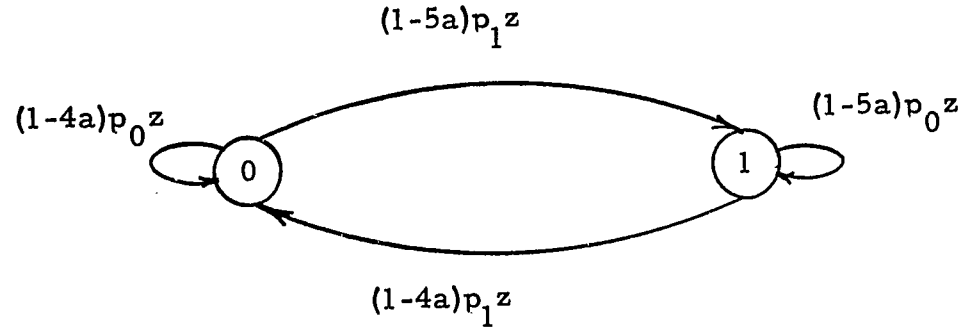

Fig. 4.8. The flow-graph of the z-transorm of $[P_T']$.

We now apply the signal flow-graph technique to find T_m's and σ^2's (see equations (3.3) and (3.5)).

$$\Delta = 1-(2-9a)p_0z - (1-9a+20a^2)(p_1^2-p_0^2)z^2$$

Assuming that the binary counter starts at "0" state, we have

$$q_0(z) = (1-(1-5a)p_0 z)/\Delta$$

$$q_1(z) = (1-5a)p_1 z/\Delta$$

Therefore

$$T_m = \frac{2p_1 - (2p_1 - 1)5a}{9ap_1 - 40a^2 p_1 + 20a^2}$$

and

$$\sigma^2 = \frac{4}{81a^2} + \frac{20p_1 - 58p_1^2}{81ap_1^2} \qquad \text{for } p_1 > 0$$

$$= \frac{1-4a}{16a^2} \qquad \text{for } p_1 = 0$$

Assuming that the binary counter starts at "1" state, we have

$$T_m = \frac{2p_1 - (2p_1 - 1)4a}{9ap_1 - 40a^2 p_1 + 20a^2}$$

and

$$\sigma^2 = \frac{4}{81a^2} + \frac{18p_1 - 54p_1^2}{81ap_1^2} \qquad \text{for } p_1 > 0$$

$$= \frac{1-5a}{25a^2} \qquad \text{for } p_1 = 0$$

The mean-time-to-first-error as a function of p_1 is shown in Table 4.1.

Table 4.1. T_m's vs p_1 for the binary counter (assume that $a \ll 1$).

p_1	T_m	
	starting state is 0	starting state is 1
00	2.25/9a	1.8/9a
.05	2/9a - 10	2/9a - 8
.10	2/9a - 4.45	2/9a - 3.56
.15	2/9a - 2.56	2/9a -2.04
.20	2/9a - 1.66	2/9a - 1.33
.25	2/9a - 1.11	2/9a - 0.89
.30	2/9a - 0.722	2/9a - 0.577
.35	2/9a - 0.477	2/9a - 0.382
.40	2/9a - 0.278	2/9a - 0.222
.45	2/9a - 0.128	2/9a - 0.0978
.50	2/9a	2/9a
.55	2/9a + 0.089	2/9a + 0.071
.60	2/9a + 0.189	2/9a + 0.151
.65	2/9a + 0.256	2/9a + 0.204
.70	2/9a + 0.316	2/9a + 0.253
.75	2/9a + 0.372	2/9a + 0.298
.80	2/9a + 0.417	2/9a + 0.334
.85	2/9a + 0.458	2/9a + 0.366
.90	2/9a + 0.495	2/9a + 0.396
.95	2/9a + 0.527	2/9a + 0.422
1.00	2/9a + 0.555	2/9a + 0.445

CHAPTER V

A MARKOV MODEL FOR RELIABILITY OF PROBABILISTIC SEQUENTIAL CIRCUITS WITH MARKOV INPUTS

In this chapter we shall develop a Markov model for probabilistic sequential circuits when the input can be expressed as a q-th ($q \geq 1$) order Markov chain[11]. We shall begin with first order Markov chain input, then extend our consideration to higher order Markov chain inputs[28]. For first order Markov chain input, the probability of the next input n-tuple u' depends on the present n-tuple u. Let the first order Markov chain input be governed by the transition matrix

$$[P_I] = \begin{array}{c} \begin{array}{c} \diagdown X(k+1) \\ X(k) \diagup \end{array} \\ \begin{array}{c} 0\ldots0 \\ 0\ldots1 \\ \vdots \\ 1\ldots1 \end{array} \end{array} \begin{array}{c} \begin{array}{cccc} 0\ldots0 & 0\ldots1 & \cdots & 1\ldots1 \end{array} \\ \begin{bmatrix} p_{(0,0)} & p_{(0,1)} & \cdots & p_{(0,2^n-1)} \\ p_{(1,0)} & p_{(1,1)} & \cdots & p_{(1,2^n-1)} \\ \vdots & \vdots & & \vdots \\ p_{(2^n-1,0)} & p_{(2^n-1,1)} & \cdots & p_{(2^n-1,2^n-1)} \end{bmatrix} \end{array} \tag{5.1}$$

where X(k+1) represents the input n-tuple at time k+1, k=0, 1, 2, . . . , and $p_{(u,u')}$ is the conditional probability that the next input n-tuple will be u', when the present input n-tuple is u, $u, u' \in \{0, 1, \ldots . 2^n-1\}$. The next input n-tuple is a function of present input n-tuple. It is governed by the following equation (5.2)

$$\overline{X(k+1)} = \overline{X(k)} . [P_I] \tag{5.2}$$

where $\overline{X(k)} = (X_0(k) \quad X_1(k) \quad \ldots \quad X_{2^n-1}(k))$ and

$\overline{X(k+1)}=(X_0(k+1) \quad X_1(k+1) \quad \ldots \quad X_{2^n-1}(k+1))$ are row vectors, $X_i(k)$ and $X_i(k+1)$ are the present and the next input n-tuples, $i=0,1,\ldots,2^n-1$, respectively.

5.1. Markov Model for Probabilistic Sequential Circuits with Markov Inputs

Consider the probabilistic sequential circuit as shown in Fig. 2.2. The modified P-matrix, $[P_{ec}]$, has a dimension of $(2^{n+r}+1) \times (2^r+1)$. For the probabilistic sequential circuit, the probability of next state v' at time k+1 depends on the present input-present state (n+r+1)-tuple at time k, where $v' \in \{e(k+1), 0, 1, \ldots, 2^r-1\}$. From the transition matrix $[P_I]$ of the first order Markov chain input, we know that the probability of the next input n-tuple u' at time k+1 depends only on the present input n-tuple at time k. Since the probability of next input n-tuple u' does not depend on the present state (r+1)-tuple of the probabilistic sequential circuit, we can modify the $[P_I]$ to have the same row headings as those of $[P_{ec}]$ in order to combine them. This can be achieved by enlarging the $[P_I]$ as follows:

$$
[P_I]_e = (X(k), S(k))
\begin{array}{c}
\\
e(k) \\
\left.\begin{array}{l}
0\ldots0,0\ldots0 \\
\\
0\ldots0,1\ldots1 \\
0\ldots1,0\ldots0 \\
\\
1\ldots1,0\ldots0 \\
\\
1\ldots1,1\ldots1
\end{array}\right.
\end{array}
\begin{array}{c}
\overbrace{\qquad\qquad\qquad\qquad\qquad\qquad}^{X(k+1)} \\
e(k+1)\quad 0\ldots0 \qquad\quad \ldots \qquad\quad 1\ldots1 \\
\left[\begin{array}{lll}
1 & 0 & \ldots\ 0 \\
0 & p_{(0,0)} & \ldots\ p_{(0,2^n-1)} \\
\vdots & \vdots & \vdots \\
0 & p_{(0,0)} & \ldots\ p_{(0,2^n-1)} \\
0 & p_{(1,0)} & \ldots\ p_{(1,2^n-1)} \\
\vdots & \vdots & \vdots \\
0 & p_{(2^n-1,0)} & \ldots\ p_{(2^n-1,2^n-1)} \\
\vdots & \vdots & \vdots \\
0 & p_{(2^n-1,0)} & \ldots\ p_{(2^n-1,2^n-1)}
\end{array}\right]
\end{array}
$$

From $[P_{ec}]$ and $[P_I]_e$ we know that both the probability of the next state (r+1)-tuple v' of the probabilistic sequential circuit and the probability of the next input (n+1)-tuple u' to this circuit depend on the present input-present state (n+r+1)-tuple i; $i \in \{e(k), 0, 1, \ldots, 2^{n+r}-1\}$. Furthermore the next input (n+1)-tuple u' and the next state (r+1)-tuple v' are mutually independent. Circuit-wise, the probabilistic sequential circuit is in parallel with the hypothetiacl circuit which represents the enlarged $[P_I]$. Therefore the total transition matrix is

$$[P_{TM}] = [P_I]_e * [P_{ec}]$$

$$
\begin{array}{c}
\\
e(k+1) \\
e(k) \\
=(X), S(k)\left\{\begin{array}{l} 0\ldots0, 0\ldots0 \\ \vdots \\ 1\ldots1, 1\ldots1 \end{array}\right.
\end{array}
\begin{array}{c}
\overbrace{X(k+1), S^{D}(k+1)} \\
0\ldots0 \quad \ldots \quad 1\ldots1 \\
\left[\begin{array}{ccc} 1 & 0 \ldots 0 & \\ \left[\begin{array}{c} \\ R_T \\ \\ \end{array}\right] & & \left[\begin{array}{c} \\ P_T' \\ \\ \end{array}\right] \end{array}\right]
\end{array}
\qquad (5.3)
$$

The dimension of $[P_{TM}]$ is $(2^{n+r}+1) \times (2^{n+r}+1)$. From equation (5.3), we know that $[P_{TM}]$ is a stochastic matrix. The probability that the next state (r+1)-tuple of the probabilistic sequential circuit is v' and the next input (n+1)-tuple u' is

$$p_j = p_{u'} \cdot p_{v'}$$

where

$$j = u' \times 2^r + v' \; ; \quad j = e(k+1) \text{ whenever } u' = e(k+1) \text{ and or } v' = e(k+1).$$

$$u' \in \{ e(k+1), 0, 1, \ldots, 2^n - 1 \}.$$

$$p_{u'} = \sum_{i=e(k)}^{2^{n+r}-1} p_i(k) \cdot p_{(i,u')}$$

$$p_{v'} = \sum_{i=e(k)}^{2^{n+r}-1} p_i(k) \cdot p_{i,v'}$$

$$i = u \times 2^r + v \; ; u \in \{e(k), 0, 1, \ldots, 2^r - 1\},$$

$$i = e(k) \text{ whenever } u = e(k) \text{ and/or } v = e(k).$$

$p_i(k)$ is the probability that the probabilistic sequential circuit is in state (r+1)-tuple v and the input (n+1)-tuple u at time k,

$p_{i,v'}$ is the (i, v') entry of $[P_{ec}]$,

$p_{(i,u')}$ is the (i, u') entry of $[P_I]_e$.

Since the probabilistic sequential circuit has first order Markov chain input, $p_{(i,u')}$ and $p_{i,v'}$ are fully defined by u, v, u' and v'. From equation (5.3), the input-state transitions of the probabilistic sequential circuit for the class of inputs considered, is described by a Markov chain with transition matrix $[P_{TM}]$.

From the above analysis, we have the following theorem:

Theorem 5.1. The behavior of the probabilistic sequential circuit with first order Markov chain input can be expressed as a Markov chain, and its state transition probability is governed by $[P_{TM}]$. The present states of the Markov chain are e(k) and the direct product of X(k) and S(k). The next states are e(k+1) and the direct product of X(k+1) and $S^D(k+1)$, where $X(k), X(k+1) \in \{0, 1, \ldots, 2^n-1\}$; $S(k), S^D(k+1) \in \{0, 1, \ldots, 2^r-1\}$.

Now let us consider the case when the Markov chain input is of q-th (q>1) order, i.e. the probability of next input n-tuple depends on the present input n-tuple and (q-1) past input n-tuples. Let the q-th order Markov chain input be governed by the transition matrix $[P_I]$.

$$[P_I] = \begin{array}{c|cccc} X(k+1), X(k), \ldots, X(k-q+2) \;/\; X(k), X(k-1), \ldots, X(k-q+1) & 0\ldots0, \ldots, 0\ldots0 & \cdots & 1\ldots1, \ldots, 1\ldots1 \\ \hline 0\ldots0, \ldots, 0\ldots0 & p_{(0,0)} & \cdots & p_{(0,2^{nq}-1)} \\ \vdots & \vdots & & \vdots \\ \underbrace{1\ldots1, \ldots, 1\ldots1}_{nq} & p_{(2^{nq}-1,0)} & \cdots & p_{(2^{nq}-1,2^{nq}-1)} \end{array}$$

We then have to enlarge the q-th order Markov chain input to have row headings e(k) and the direct product of X(k), X(k-1), ..., X(k-q+1) and S(k). The resulting q-th order Markov chain input is $[P_I]_e$ and

the modified P-matrix of the probabilistic sequential circuit, $[P_{ec}]$, which has row headings e(k) and the direct product of X(k) and S(k), must be enlarged. The resulting P-matrix, denoted by $[P_{ec}]_e$, must have the same row headings as $[P_I]_e$. applying the same technique as given before, we have

$$[P_{TM}] = [P_I]_e * [P_{ec}]_e \tag{5.4}$$

5.2. Algorithm for Finding the Reliability of Probabilistic Sequential Circuits with Markov Inputs

The flow chart representation of an algorithm for finding the reliability of probabilistic sequential circuits with q-th ($q \geq 1$) order Markov chain inputs isshown in Fig. 5.1. It is a convenient summary of our previous more detailed discussion.

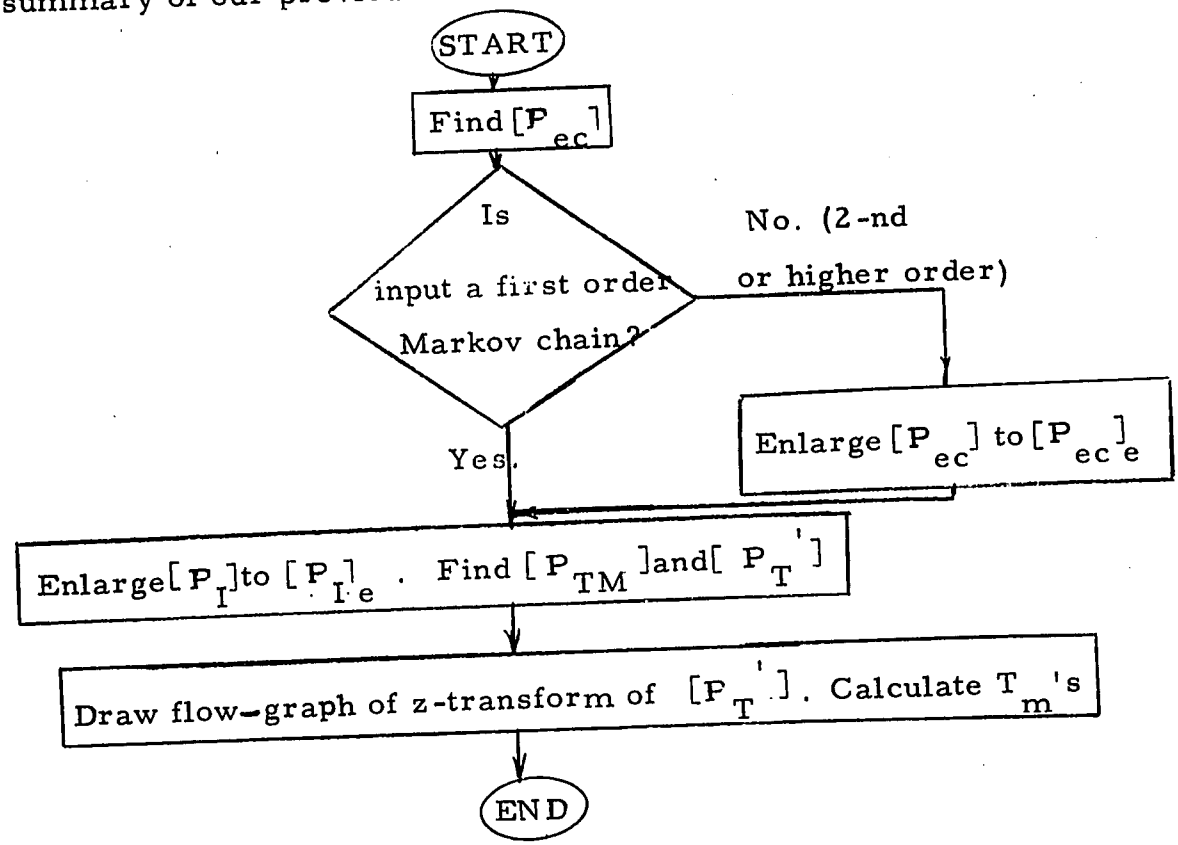

Fig. 5.1. The flow chart of algorithm for finding the reliability of probabilistic sequential circuits with Markov input.

5.3. Example.

In order to illustrate the method presented in this chapter, an example of binary counter is considered. Its ciruit diagram was shown in Fig. 4.7. The transition matrix of the first order Markov chain input is given as

$$[P_I] = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{bmatrix} p_{(0,0)} & p_{(0,1)} \\ p_{(1,0)} & p_{(1,1)} \end{bmatrix} \end{array}$$

(X(k+1) / X(k))

The modified P-matrix of this binary counter was found (equation 4.3) to be

$$[P_{ec}] = (X(k), S(k)) \begin{array}{c} \\ 0\,0 \\ 0\,1 \\ 1\,0 \\ 1\,1 \end{array} \begin{array}{c} \begin{array}{ccc} e(k+1) & & S^D(k+1) \\ & & \overbrace{\qquad\qquad} \\ e(k) & 1 & 0 \qquad\qquad 1 \end{array} \\ \begin{bmatrix} 4a & 1-4a & 0 \\ 5a & 0 & 1-5a \\ 5a & 0 & 1-5a \\ 4a & 1-4a & 0 \end{bmatrix} \end{array}$$

The enlarged transition matrix of the first order Markov chain input as explained before(Section 5.1) is

$$[P_{I_e}] = (X(k), S(k))$$

e(k) \ X(k+1)	0	1
e(k)=1	0	0
0 0	$p_{(0,0)}$	$p_{(0,1)}$
0 1	$p_{(0,0)}$	$p_{(0,1)}$
1 0	$p_{(1,0)}$	$p_{(1,1)}$
1 1	$p_{(1,0)}$	$p_{(1,1)}$

Substitution of the above two matrices into equation (5.3) yields

$$[P_{TM}] = (X(k)\ S(k))$$

e(k) \ X(k+1), S^D(k+1)	0 0	0 1	1 0	1 1
e(k)=1	0	0	0	0
0 0	4a	$(1-4a)p_{(0,0)}$	0	$(1-4a)p_{(0,1)}$
0 1	5a	0	$(1-5a)p_{(0,0)}$	0	$(1-5a)p_{(0,1)}$
1 0	5a	0	$(1-5a)p_{(1,0)}$	0	$(1-5a)p_{(1,1)}$
1 1	4a	$(1-4a)p_{(1,0)}$	0	$(1-4a)p_{(1,1)}$

and therefore

$$[P_T] =$$

X(k+1), S^D(k+1) / X(k), S(k)	0 0	0 1	1 0	1 1
0 0	$(1-4a)p_{(0,0)}$	0	$(1-4a)p_{(0,1)}$	0
0 1	0	$(1-5a)p_{(0,0)}$	0	$(1-5a)p_{(0,1)}$
1 0	0	$(1-5a)p_{(1,0)}$	0	$(1-5a)p_{(1,1)}$
1 1	$(1-4a)p_{(1,0)}$	0	$(1-4a)p_{(1,1)}$	0

The flow-graph of the z-transform of $[P_T']$ is shown in Fig. 5.2.

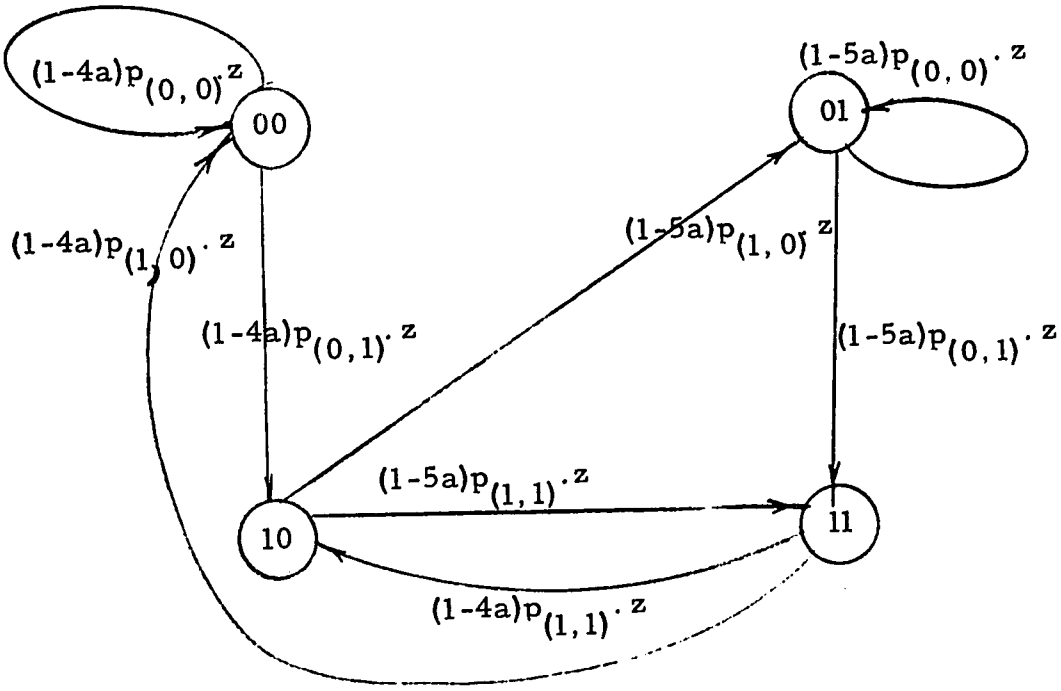

Fig. 5.2. The flow-graph of the z-transform of $[P_T']$.

We now apply the standard signal flow-graph technique to find T_m's (see equation (3.3)), we have

$$\Delta = 1 - p_{(0,0)}(1-4a)z - p_{(0,0)}(1-5a)z - p_{(1,1)}^2(1-5a)(1-4a)z^2 -$$

$$p_{(1,0)}p_{(0,1)}p_{(1,1)}(1-5a)^2(1-4a)z^3 - p_{(1,0)}p_{(0,1)}p_{(1,1)}(1-4a)^2(1-5a)z^3 -$$

$$p_{(1,0)}^2p_{(0,1)}^2(1-4a)^2(1-5a)^2z^4 + p_{(0,0)}^2(1-4a)(1-5a)z^2 +$$

$$p_{(0,0)}p_{(1,1)}^2(1-4a)^2(1-5a)z^3 + p_{(0,0)}p_{(1,1)}^2(1-5a)^2(1-4a)z^3 -$$

$$p_{(0,0)}p_{(0,1)}p_{(1,0)}p_{(1,1)}(1-4a)^2(1-5a)^2z^4 + p_{(0,0)}p_{(0,1)}p_{(1,0)}p_{(1.1)}$$

$$(1-5a)^2(-4a)^2z^4 - p_{(0,0)}^2p_{(1,1)}^2(1-5a)^2(1-4a)^2z^4$$

Assuming that the binary counter starts at ''00'' state. we have

$$q_0(z)=q_{00}(z) =(1-p_{(0,0)}(1-5a)z - p^2_{(1,1)}(1-5a)(1-4a)z^2 -$$

$$p_{(0,1)}p_{(1,0)}p_{(1,1)}(1-5a)^2(1-4a)z^3 +$$

$$p_{(0,0)}p^2_{(1,1)}(1-4a)^2(1-5a)z^3)/\Delta$$

$$q_1(z)=q_{01}(z)=(p_{(0,1)}p_{(1,0)}(1-4a)(1-5a)z^2)/\Delta$$

$$q_2(z)=q_{10}(z)=p_{(0,1)}(1-4a)z(1-p_{(0,0)}(1-5a)z)/\Delta$$

$$q_3(z)=q_{11}(z)=(p_{(0,1)}p_{(1,1)}(1-4a)(1-5a)z^2(1-p_{(0,0)}(1-5a)z) +$$

$$p^2_{(0,1)}p_{(1,0)}(1-4a)(1-5a)^2z^3)/\Delta$$

Substitution of the appropriate terms in equation (3.3) gives

$$T_m = \frac{2}{9a} + \frac{-27 +p_{(0,0)}(55-28p_{(1,1)})-23p^2_{(0,0)}+28p_{(1,1)}-5p^2_{(1,1)}}{9(2-3p_{(0,0)}+p_{(0,0)}p_{(1,1)}+p^2_{(0,0)}-p_{(1,1)})}$$

for $p_{(0,0)} \neq 1$

$$= 1/4a \qquad \text{for } p_{(0,0)}=p_{(1,1)}=1 \text{ or } p_{(0,0)}=p_{(1,0)}=1.$$

The mean-time-to-first-error for the binary counter starts at ''01'', ''10'' or ''11'' can be found in a similar fashion.

CHAPTER VI

SYNTHESIS OF RELIABLE SEQUENTIAL CIRCUITS

In this chapter a method of obtaining a reliable sequential circuit by using error-correcting codes and redundancy technique is proposed. According to this method the probability of error in the final output of the designed sequential circuit can be made as small as desired by increasing the reliability of the final gate (OR or AND gate) in the output decoder, provided that the number of simultaneous errors, t, does not exceed the correction ability of the error-correcting code employed. Moreover, the errors in the r'-tuple at time k do not cause any error in the state r'-tuple and output m'-tuple at time k+1, where r' is the number of state variables and m' the number of output variables needed to realize the sequential circuit concerned.

In the following sections we shallconsider the case in which the output of any component can be connected to one output and/or state variable.

6.1. Application of Error-Correcting Codes to the Secondary State Assignments and Output States

In a communication system, it is assumed that the transmitter andthe receiver are perfect, and that the error is caused by the noise in the communication channel between the transmitter and the receiver. In order to protect the signal from errors caused by the noise during transmission, the signal may be encoded into a suitable error-correcting code, such that up to a maximum of a fixed number of errors in the received signal can be corrected by the decoder. If the probability of the number of simultaneous errors is greater than the correction ability of the employed error correcting code, then the output from the decoder may be in error (i.e. the actual transmitted signal may not be recovered). This technique utilizes the well-known signal redundancy scheme [29]

The circuit redundancy technique for combinatorial circuits has been developed by several authors[1,2,3,4,6,7,16]. Sheng and Chen [5] simplified the method developed by Winograd and Cowan [4]. The proposed method extends Sheng and Chen's method to the case of sequential circuits. The proposed method applies error-correcting codes to both the output states and the internal states (which were called "states" in the previous chapter) of the sequential circuit. To improve the reliability of sequential circuit, both the internal states and the output states are encoded in the manner to be described in the following:

Let us assume that the sequential circuit has g internal states, S_i, i=o,1,...,g-1. The erroneous states of S_i are represented by $\{S_i', S_i'' \ldots, S_i^a$, $a = \sum_{t'=1}^{t} \binom{r'}{t'}$, where t is the desired number of simultaneous errors to be corrected and r' is the number of state variables needed to represent the internal states$\}$. Should the internal state be S_i because of t, or less, state variables in error, S_i is changed to one of the erroneous states of S_i. The Hamming distance (H.D.) between any one of the erroneous states of S_i and S_j, $i \neq j, i, j \in \{0,1,\ldots,g-1\}$, is greater than t.

The procedure consists of the following:

Step 1. Encode the internal states of the sequential circuit so that the Hamming distance between each pair of internal states (S_i, S_j, $i \neq j$) is equal to or greater than 2t+1.

Step 2. Assign the next internal states of the erroneous internal states of S_i as the ones corresponding to S_i for all $i \in \{0, 1, \ldots, g-1\}$.

Step 3. Assign the output states corresponding to the erroneous states of S_i of the as the ones corresponding to S_i for all $i \in \{0,1,\ldots,g-1\}$.

Step 4. Add output checking function(s) if necessary, such that the H.D. between any two encoded output states (m'-tuples) Z_i and Z_j is equal to or greater than 2t+1, where $i, j \in \{0,1,\ldots,2^{m'}-1\}$.

The purpose of steps 1,2 and 3 is to prevent the errors in the state variables which occur at time k from causing any further errors in the sequential circuit at time k+h , h $\geq$1.

Let us suppose now that the next internal state is S_i. If t, or less, errors occur at time k, the next internal state becomes, say, S_i'. In steps 2 and 3, we assign the same next internal states and output m-tuples, say S_j and Z_j respectively, to both S_i and S_i'. Now if the internal state at time k+1 is S_i' instead of S_i, then the next internal state of S_i', i.e. the internal state at time k+2, is S_j, and the output m-tuple is Z_j. Hence the next internal state r'-tuple and the output m-tuple do not change, although the internal state at time k+1 is changed from S_i to S_i' due to t, or less, simultaneous errors in the sequential circuit at time k. The correctness of the next internal state and output m'-tuple depends on the correct operations of the components employed at time k+1. The erroneous internal state at time k does not affect the next internal state r'-tuple and output m-tuple at time k+1, i.e. the internal state error is corrected at the next instant of time by the combinatorial circuit and does not propagate. The combinatorial circuit also computes the next internal state and present output m-tuple at the same time. Therefore we can say that t, or less, simultaneous errors appearing in the internal state can be corrected by the combinatorial circuit exactly one step of time after their appearance, i.e. the errors (t or less) in the internal state do not propagate).

Step 4 is used to enlarge the output space, such that each pair of output states has a H. D. of 2t+1 or greater.

6.2. A Method of Constructing Reliable Output Decoders

In this section a method of constructing a reliable output decoder is proposed. Since the output m'-tuples from the sequential circuit have the property that their H. D. $\geq$2t+1, then t, or less, simultaneous errors which appear in the output m'-tuples can be corrected by the output decoder. The output decoder can decode any m'-tuple into

correct m-tuple provided the number of errors is no greater than t ($m' \geq m$). The reliable output decoder is divided into two layers. The first layer consists of coding circuits which are used to decode the m'-tuples into the correct output m-tuples ($m' \geq m$). The redundancy technique is applied to the first layer of the reliable output decoder. The number of copies of the i-th decoding circuit which is used to extract the i-th output variable is 2t+1; $i \in \{1, 2, \ldots, m\}$. The output of the j-th copy of the i-th decoding circuit is denoted by $y_{i,j}$; $j \in \{1, 2, \ldots, 2t+1\}$. The second layer of the reliable output decoder consists of m majority gates. The input variables to the i-th majority gate are $y_{i,j}$'s; $j \in \{1, 2, \ldots, 2t+1\}$. and the output is y_i; $i \in \{1, 2, \ldots, m\}$. The above scheme is shown in Fig. 6.1 , in which f_i is the i-th decoding circuit, $i \in \{1, 2, \ldots, m\}$, and M's are majority gates.

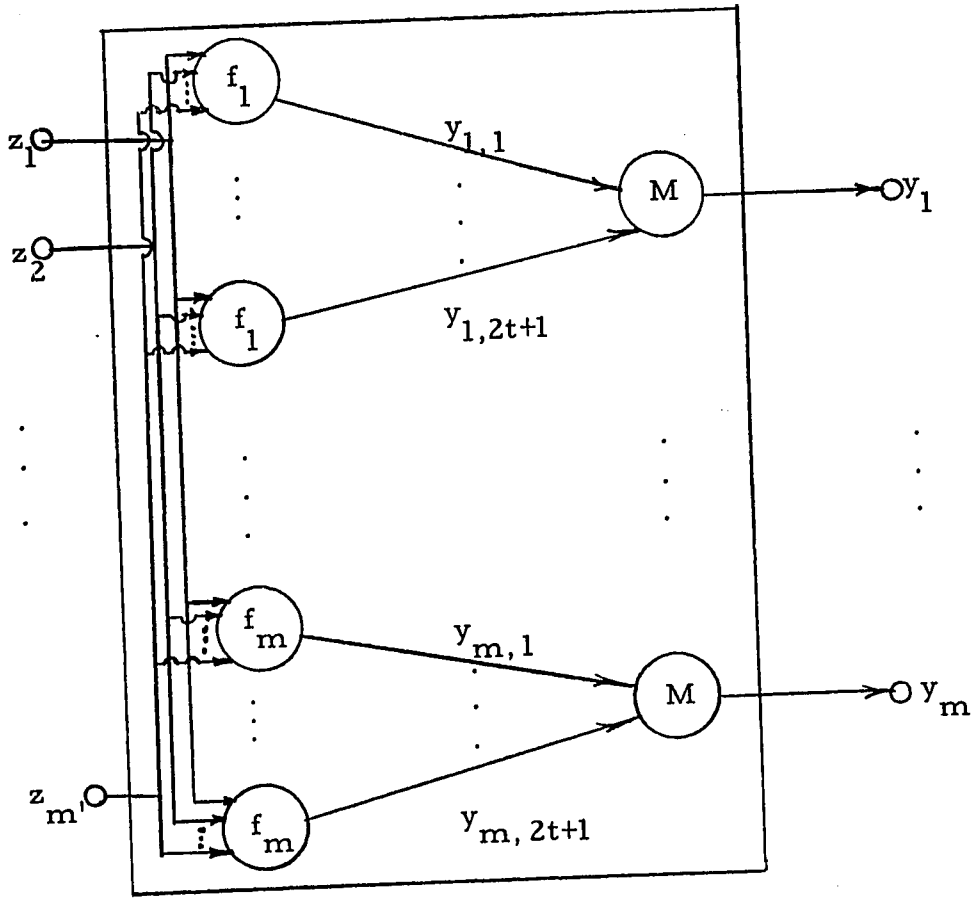

Fig. 6.1. Reliable output decoder

We now derive the P-matrix for the majority gate with three inputs, (i.e. for t=1). The switching equation for the i-th majority gate is

$$y_i = y_{i,1} \cdot y_{i,2} + y_{i,2} \cdot y_{i,3} + y_{i,1} \cdot y_{i,3} \tag{6.1}$$

$$= (y_{i,1} + y_{i,2})(y_{i,2} + y_{i,3})(y_{i,1} + y_{i,3}) \tag{6.2}$$

The block diagram of the i-th majority gate is

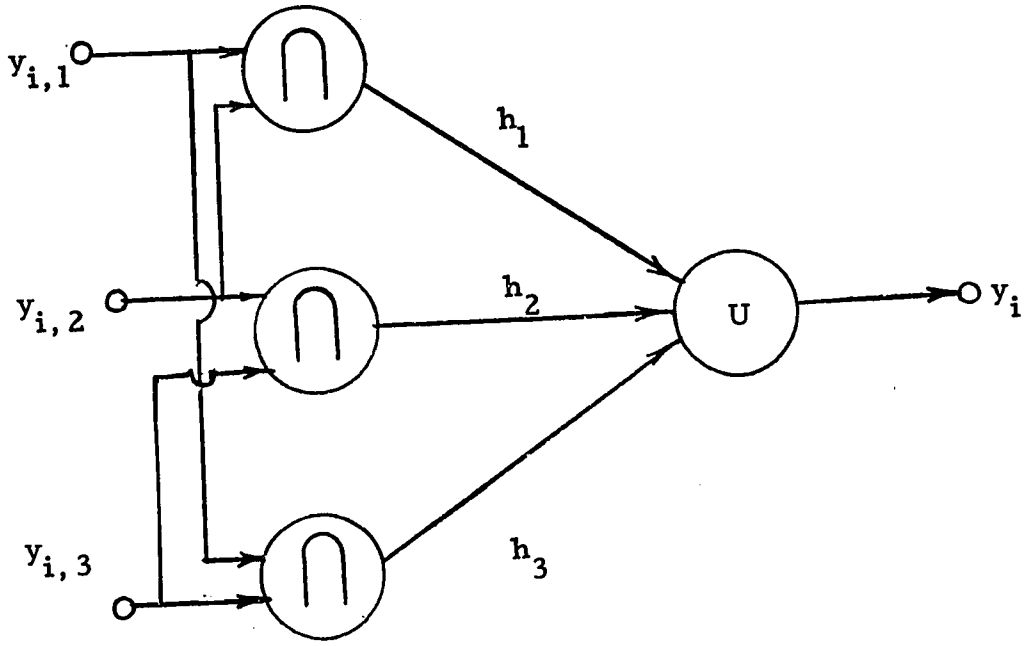

Fig. 6.2. Realization of equation (6.1).

(Type 1 majority gate).

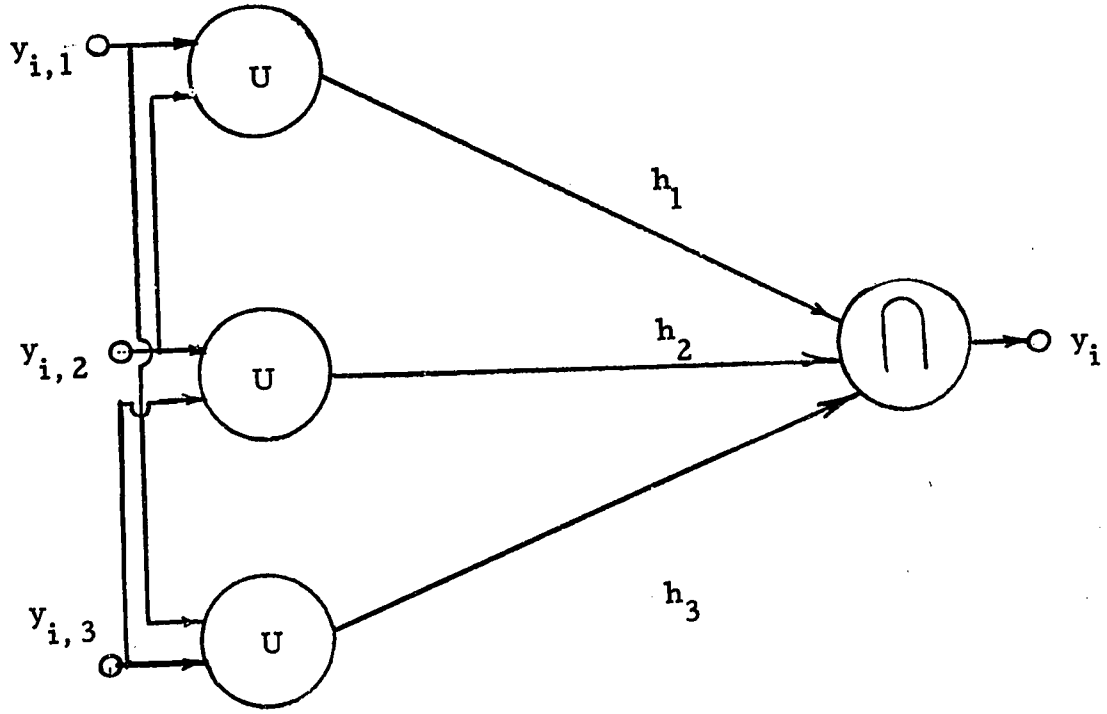

Fig. 6.3. Realization of equation (6.2).

(Type 2 majority gate).

The P-matrices for the majority gate of Fig. 6.2 and Fig. 6.3, denoted by $[P_{M1}]$ and $[P_{M2}]$ respectively, are

$$[P_{M1}]=\begin{Bmatrix} \begin{matrix} h_1 \\ y_{i,1},y_{i,2},y_{i,3} \end{matrix} & 0 & 1 \\ 0\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 0\ 1 & 1-c_0 & c_0 \\ 0\ 1\ 0 & 1-c_1 & c_1 \\ 0\ 1\ 1 & 1-c_1 & c_1 \\ 1\ 0\ 0 & 1-c_2 & c_2 \\ 1\ 0\ 1 & 1-c_2 & c_2 \\ 1\ 1\ 0 & c_3 & 1-c_3 \\ 1\ 1\ 1 & c_3 & 1-c_3 \end{Bmatrix} * \begin{Bmatrix} \begin{matrix} h_2 \\ y_{i,1},y_{i,2},y_{i,3} \end{matrix} & 0 & 1 \\ 0\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 0\ 1 & 1-c_1 & c_1 \\ 0\ 1\ 0 & 1-c_2 & c_2 \\ 0\ 1\ 1 & c_3 & 1-c_3 \\ 1\ 0\ 0 & 1-c_0 & c_0 \\ 1\ 0\ 1 & 1-c_1 & c_1 \\ 1\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1\ 1 & c_3 & 1-c_3 \end{Bmatrix} * \begin{Bmatrix} \begin{matrix} h_3 \\ y_{i,1},y_{i,2},y_{i,3} \end{matrix} & 0 & 1 \\ 0\ 0\ 0 & 1-c_0 & c_0 \\ 0\ 0\ 1 & 1-c_1 & c_1 \\ 0\ 1\ 0 & 1-c_0 & c_0 \\ 0\ 1\ 1 & 1-c_1 & c_1 \\ 1\ 0\ 0 & 1-c_2 & c_2 \\ 1\ 0\ 1 & c_3 & 1-c_3 \\ 1\ 1\ 0 & 1-c_2 & c_2 \\ 1\ 1\ 1 & c_3 & 1-c_3 \end{Bmatrix} \cdot \begin{Bmatrix} \begin{matrix} y_i \\ h_1,h_2,h_3 \end{matrix} & 0 & 1 \\ 0\ 0\ 0 & 1-d_0 & d_0 \\ 0\ 0\ 1 & d_1 & 1-d_1 \\ 0\ 1\ 0 & d_2 & 1-d_2 \\ 0\ 1\ 1 & d_3 & 1-d_3 \\ 1\ 0\ 0 & d_4 & 1-d_4 \\ 1\ 0\ 1 & d_5 & 1-d_5 \\ 1\ 1\ 0 & d_6 & 1-d_6 \\ 1\ 1\ 1 & d_7 & 1-d_7 \end{Bmatrix}$$

y_i / $y_{i,1}, y_{i,2}, y_{i,3}$	0	1
0 0 0	$1-3c_0-d_0$	$3c_0+d_0$
0 0 1	$1-c_0-2c_1-d_0$	$c_0+2c_1+d_0$
0 1 0	$1-c_0-c_1-c_2-d_0$	$c_0+c_1+c_2+d_0$
0 1 1	c_3+d_2	$1-c_3-d_2$
1 0 0	$1-c_0-2c_2-d_0$	$c_0+2c_2+d_0$
1 0 1	c_3+d_1	$1-c_3-d_1$
1 1 0	c_3+d_4	$1-c_3-d_4$
1 1 1	d_7	$1-d_7$

(6.3)

and

h_1 / $y_{i,1}, y_{i,2}, y_{i,3}$	0	1
0 0 0	$1-d_0$	d_0
0 0 1	$1-d_0$	d_0
0 1 0	d_1	$1-d_1$
0 1 1	d_1	$1-d_1$
1 0 0	d_2	$1-d_2$
1 0 1	d_2	$1-d_2$
1 1 0	d_3	$1-d_3$
1 1 1	d_3	$1-d_3$

*

h_2 / $y_{i,1}, y_{i,2}, y_{i,3}$	0	1
0 0 0	$1-d_0$	d_0
0 0 1	d_1	$1-d_1$
0 1 0	d_2	$1-d_2$
0 1 1	d_3	$1-d_3$
1 0 0	$1-d_0$	d_0
1 0 1	d_1	$1-d_1$
1 1 0	d_2	$1-d_2$
1 1 1	d_3	$1-d_3$

$[P_{M2}] =$ *

h_3 / $y_{i,1}, y_{i,2}, y_{i,3}$	0	1
0 0 0	$1-d_0$	d_0
0 0 1	d_1	$1-d_1$
0 1 0	$1-d_0$	d_0
0 1 1	d_1	$1-d_1$
1 0 0	d_2	$1-d_2$
1 0 1	d_3	$1-d_3$
1 1 0	d_2	$1-d_2$
1 1 1	d_3	$1-d_3$

$\cdot$

y_i / h_1, h_2, h_3	0	1
0 0 0	$1-c_0$	c_0
0 0 1	$1-c_1$	c_1
0 1 0	$1-c_2$	c_2
0 1 1	$1-c_3$	c_3
1 0 0	$1-c_4$	c_4
1 0 1	$1-c_5$	c_5
1 1 0	$1-c_6$	c_6
1 1 1	c_7	$1-c_7$

$=$

y_i / $y_{i,1}, y_{i,2}, y_{i,3}$	0	1
0 0 0	$1-c_0$	c_0
0 0 1	$1-d_0-c_3$	d_0+c_3
0 1 0	$1-d_0-c_6$	d_0+c_6
0 1 1	$2d_1+d_3+c_7$	$1-2d_1-d_3-c_7$
1 0 0	$1-d_0-c_5$	d_0+c_5
1 0 1	$d_1+d_2+d_3+c_7$	$1-d_1-d_2-d_3-c_7$
1 1 0	$2d_1+d_3+c_7$	$1-2d_1-d_3-c_7$
1 1 1	$3d_3+c_7$	$1-3d_3-c_7$

(6.4)

let us assume that the component gates are realized with diodes. Let μ be the probability that a diode is conducting when it should be not conducting, and let μ' be the probability that the diode is not conducting when it should be conducting. Assume that $\mu, \mu' \ll 1$. Hence the higher order product terms can be neglected. For the circuit realization of type 1 majority gate (Fig. 6.2), we have $d_1=d_2=d_4=\mu'$, $d_3=d_5=d_6=(\mu')^2=0$, $d_0=3\mu$, $d_7=(\mu')^3=0$, $c_0=\mu^2=0$, $c_1=c_2=\mu$ and $c_3=2\mu'$. For the circuit realization of type 2 majority gate (Fig. 6.3), we have $c_0=(\mu)^3=0$, $c_7=3\mu'$, $c_1=c_2=c_4=\mu^2=0$, $c_3=c_5=c_6=\mu$, $d_0=2\mu$, $d_1=d_2=\mu'$ and $d_3=(\mu')^2=0$. Thus the P-matrices for the type 1 and type 2 majority gates become:

$$[P_{M1}] = \begin{array}{c|ccc|cc} y_i \backslash y_{i,1}, y_{i,2}, y_{i,3} & & & & 0 & 1 \\ \hline 0 & 0 & 0 & & 1-3\mu & 3\mu \\ 0 & 0 & 1 & & 1-5\mu & 5\mu \\ 0 & 1 & 0 & & 1-5\mu & 5\mu \\ 0 & 1 & 1 & & 3\mu' & 1-3\mu' \\ 1 & 0 & 0 & & 1-5\mu & 5\mu \\ 1 & 0 & 1 & & 3\mu' & 1-3\mu' \\ 1 & 1 & 0 & & 3\mu' & 1-3\mu' \\ 1 & 1 & 1 & & 0 & 1 \end{array} \qquad (6.5)$$

and

$$[P_{M2}]= \begin{array}{c|cc} y_{i,1}, y_{i,2}, y_{i,3} \backslash y_i & 0 & 1 \\ \hline 0\ 0\ 0 & 1 & 0 \\ 0\ 0\ 1 & 1-3\mu & 3\mu \\ 0\ 1\ 0 & 1-3\mu & 3\mu \\ 0\ 1\ 1 & 5\mu' & 1-5\mu' \\ 1\ 0\ 0 & 1-3\mu & 3\mu \\ 1\ 0\ 1 & 5\mu' & 1-5\mu' \\ 1\ 1\ 0 & 5\mu' & 1-5\mu' \\ 1\ 1\ 1 & 3\mu' & 1-3\mu' \end{array} \qquad (6.6)$$

Assume that $\mu=\mu'$. Then from equations (6.5) and (6.6), we note that the reliability of the same sequential circuit with type 1 majority gate may be different from that of type 2 majority gate. Assume that there is no error in the $y_{i,j}$'s, and that the probability of producing $y_i=1$ is p, then the probability of producing correct output y_i is equal to $p + (1-p)(1-3\mu) = 1-3\mu(1-p)$ when using type 1 majority gate. However the probability of producing correct y_i is equal to $p(1-3\mu) + (1-p) = 1-3\mu$ when using type 2 majority gate. When $p \geq 1/2$, $1-3\mu(1-p)$ is greater than or equal to $1-3\mu p$. Thus the reliability of this sequential circuit with type 1 majority gate is higher than or equal to that of using type 2 majority gate. Similarly, if $p \leq 1/2$, the reliability of the sequential circuit with type 2 majority gate is not less than that of using type 1 majority gate. If $p=1/2$, the reliability of the sequential circuit when using type 1 majority gate is equal to that of using type 2 majority gate. In order to obtain a more

reliable majority gate, the diode in the OR(AND) gates in the last components of the type 1(2) majority gates are replaced by 2 diodes in series(parallel) for single-error correction as seen from equatios (6.5) and (6.6) (since $d_0=3\mu^2=0$ for type 1 majority gate, $c_7=3(\mu')^2=0$ for type 2 majority gate). For t-error correction, the number of diodes in series (parallel) is equal to t+1, since the probability that there are t simultaneous errors is assumed to be different from zero, while the probability of t+1 simultaneous errors is equal to zero. Also the number of copies of i-th decoding circuits is (2t+1).

If we construct the sequential circuit and output decoder according to the above scheme, then t, or less, simultaneous errors in the sequential circuit are corrected by the combinatorial circuit (one unit of time later) and the circuits in the first layer of the reliable output decoder (in the same unit of time). The t, or less, simultaneous errors in the decoding circuits are corrected by the circuits in the second layer of the reliable output decoder. The reliability of the decoder depends on the last components of the majority gates. In spite of the complexity of the combinatorial circuit and the decoding circuits, the reliability of the sequential circuit depends on the reliability of the last components of the majority gates and the probability p with $y_i=1$.

When m=1 and t=1, the first layer of the reliable output decoder can be eliminated. In this case, three output variables are needed, i.e. m'=3. Two correct m'-tuples are used, so that we can always choose m'-tuples 111 and 000 to represent the 1 and 0 output states respectively. The decoding circuits are simply three inputs majority gates, similar to that in the second layer of the reliable output decoder. Thus we can combine the first and second layers of the reliable output decoder into a single majority gate.

6.3. An Algorithm for Constructing Reliable Sequential Circuits.

Now we are going to summarize the results, presented in the previous sections, for constructing a reliable sequential circuit which has t-error correcting ability. Assume that the sequential circuit has g states and m output variables. The algorithm consists of the following:

Step 1. Encode the internal states so that the H.D. between each pair of states is 2t+1 or greater.

Step 2. Modify the flow table such that the next internal and output states of the erroneous states of S_i are the same as those of S_i, $i \in \{0,1,\ldots,g-1\}$.

Step 3. Derive the switching equations of the unit delay elements.

The purpose of these three steps is to construct reliable state transitions.

Step 4. Encode the output states so that the H. D. between each two output m'-tuples will be $\geq 2t+1$, $m' \geq m$. Find the switching equations for z_i; $i \in \{1,2,\ldots,m'\}$.

Step 5. If m>1, then apply the method presented in Section 6.2. Derive the switching equations of $y_{i,j}$'s; $i \in \{1,2,\ldots,m\}$; $j=1,2,\ldots,2t+1$. 2t+1 copies of each $f_1, f_2, \ldots, f_m$ are employed.

The last two steps are used to construct the first layer of the reliable output decoder.

Step 6. Construct m majority gates each of which has 2t+1 input variables and a single output variable. The input to the i-th majority gate are $y_{i,j}$'s; $j \in \{1,2,\ldots,2t+1\}$, and output variable is y_i.

Step 7. Derive the switching equations for y_i's in terms of $y_{i,j}$'s ; $i \in \{1,2,\ldots,m\}$.

These two steps are used to construct the second layer of the reliable output decoder.

Step 8. The reliability of the output decoder can be increased by using the method presented in Section 6.2.

The last step is used to improve the reliability of the output

decoder as well as of the overall sequential circuit.

Step 5'. If m=1, then let 111...1 be the output state 1 and 000...0 be the output state 0. The number of output functions is (2t+1), i.e. m'=2t+1. Go to step 7.

If the sequential circuit is designed according to the foregoing steps, we can obtain a sequential circuit which can correct t, or less, simultaneous errors which are caused by the erroneous operations of components, in the internal states and/or output states.

6.4. Examples.

Example 6.1. Design a binary counter such that any single error in the operation of its components cannot cause erroneous output.

The flow table of the binary counter is shown in Fig. 6.4.

$x_1(k)$ / $S(k)$	0	1
S_0	S_0 / 0	S_1 / 0
S_1	S_1 / 0	S_0 / 1

Fig. 6.4. Flow table of binary counter.

where each entry means: $S_i(k+1)$ / $y_1(k)$=next state / output state.

Step 1. Since there are only two internal states in the binary counter and t=1, then three state variables can be used to represent S_0 and S_1. Let us consider the following secondary state assignment: $s_1=s_2=s_3=0$ is assigned to represent S_0 and $s_1=s_2=s_3=1$ is assigned to represent S_1, where s_i's, i = 1, 2, 3 , are the inputs and outputs of the unit delay elements D_i's. Thus (S_0'=001, S_0''=010 and S_0'''=100) and (S_1'=110, S_1''=101 and S_1'''=011) are the erroneous states of S_0 and S_1 respectively.

Step 2. The modified flow table is shown in Fig. 6.5.

$s_1(k), s_2(k), s_3(k)$	$x_1(k)$ = 0	$x_1(k)$ = 1
0 0 0	000 / 0	111 / 0
0 0 1	000 / 0	111 / 0
0 1 0	000 / 0	111 / 0
1 0 0	000 / 0	111 / 0
1 1 1	111 / 0	000 / 1
0 1 1	111 / 0	000 / 1
1 0 1	111 / 0	000 / 1
1 1 0	111 / 0	000 / 1

Fig. 6.5. Modified flow table of binary counter with single-error correcting ability in the state transitions.

Step 3. The switching equations of the outputs of the unit delay elements are [30]

$$s_1(k+1) = x_1(k) \oplus (s_1(k)s_2(k) + s_1(k)s_3(k) + s_2(k)s_3(k)) \qquad (6.7)$$

$$s_2(k+1) = x_1(k) \oplus (s_1(k)s_2(k) + s_1(k)s_3(k) + s_2(k)s_3(k)) \qquad (6.8)$$

$$s_3(k+1) = x_1(k) \oplus (s_1(k)s_2(k) + s_1(k)s_3(k) + s_2(k)s_3(k)) \qquad (6.9)$$

where the sbmbol " $\oplus$ " denotes the exclusive-or operation.

Up to this point a binary counter has been designed for the requiremen of correcting all single-errors in its internal state transitions.

Step 4. We are now going to encode the output states so that the H. D. between two output m'-tuples will be $\geq 2t+1=3$. In order to have single-error correction ability, t=1, two more output functions z_2 and z_3 must be added such that the H. D. between these two output states is three. Let $z_3=z_2=z_1$. Then the flow table of Fig. 6.5 becomes

$s_1(k),s_2(k),s_3(k)$	0	1
0 0 0	000 / 000 / 0	111 / 000 / 0
0 0 1	000 / 000 / 0	111 / 000 / 0
0 1 0	000 / 000 / 0	111 / 000 / 0
1 0 0	000 / 000 / 0	111 / 000 / 0
1 1 1	111 / 000 / 0	000 / 111 / 1
0 1 1	111 / 000 / 0	000 / 111 / 1
1 0 1	111 / 000 / 0	000 / 111 / 1
1 1 0	111 / 000 / 0	000 / 111 / 1

Fig. 6.6. Modified flow table of binary counter with single-error correcting ability in both state transitions and output states.

The switching equations for $z_1(k)$, $z_2(k)$ and $z_3(k)$ are

$$z_1(k) = x_1(k)(s_1(k)s_2(k) + s_1(k)s_3(k) + s_3(k)s_2(k)) \qquad (6.10)$$

$$z_2(k) = x_1(k)(s_1(k)s_2(k) + s_1(k)s_3(k) + s_2(k)s_3(k)) \quad (6.11)$$

$$z_3(k) = x_1(k)(s_1(k)s_2(k) + s_1(k)s_3(k) + s_2(k)s_3(k)) \quad (6.12)$$

Step 5'. Since there are only two output states and $y_1(k)=z_1(k)=z_2(k)=z_3(k)$, the first and second layers can be combined into a single majority gate.

Step 7. The switching equation of the output y_1 of the reliable output decoder is

$$y_1(k) = z_1(k)z_2(k) + z_1(k)z_3(k) + z_2(k)z_3(k)$$

$$=(z_1(k) + z_2(k))(z_1(k) + z_3(k))(z_2(k) + z_3(k))$$

By studying equations (6.7) to (6.12), we note that equations (6.10) to (6.12) are part of the equations (6.7) to (6.9) respectively. In this case the circuit which realizes $z_1(k)$, $z_2(k)$ and $z_3(k)$ is part of the circuit which realizes $s_1(k+1)$, $s_2(k+1)$ $s_3(k+1)$ respectively. When type 2 majority gate is used, the final circuit of the reliable binary counter is of the form in Fig. 6.7. In order to increase the reliability of the majority gate each diode in the last AND gate of the majority gate is replaced by two diodes in parallel.

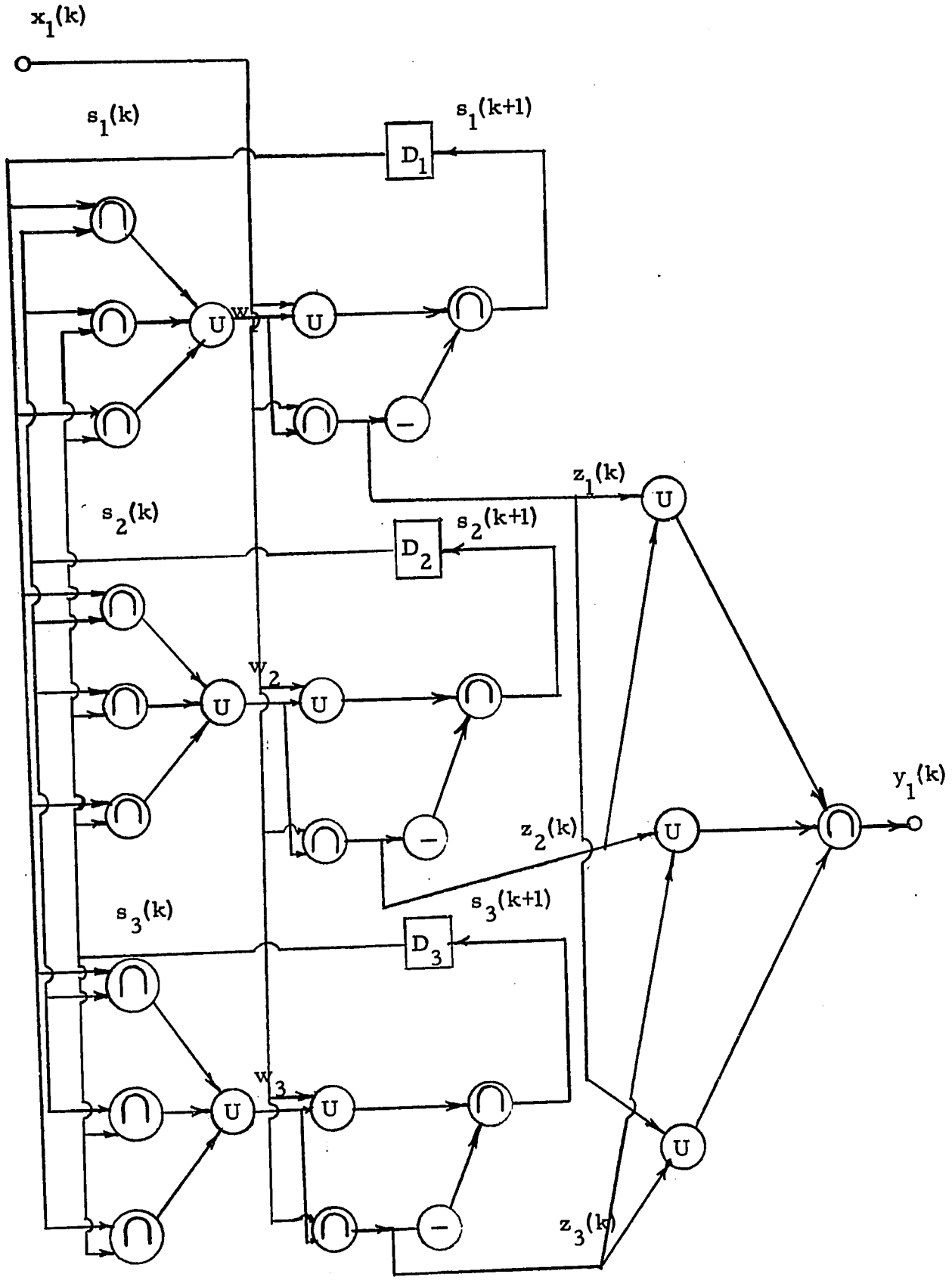

Fig. 6.7. Binary counter with single-error correcting capability.

Now the reliability of this binary counter(assuming that a single diode is used in each input terminal of the last AND gate) will be estimated and compared with the binary counter given in Fig. 4.7. The P-matrix between (X(k), S(k))and S(k+1) denoted by $[P_{XS-S}]$, can be found as (assuming that $c_i=d_i=e_i=a_i=b_i=a$ for $i \in\{0,1,\ldots,7\}$

$[P_{XS-S}]=$

$S^D(k+1)$ / X(k), S(k)		000	001	010	011	100	101	110	111
00	00	1-21a	7a	7a	0	7a	0	0	0
00	01	1-21a	7a	7a	0	7a	0	0	0
00	10	1-21a	7a	7a	0	7a	0	0	0
00	11	0	0	0	7a	0	7a	7a	1-21a
01	00	1-21a	7a	7a	0	7a	0	0	0
01	01	0	0	0	7a	0	7a	7a	1-21a
01	10	0	0	0	7a	0	7a	7a	1-21a
01	11	0	0	0	6a	0	6a	6a	1-18a
10	00	0	0	0	9a	0	9a	9a	1-27a
10	01	0	0	0	9a	0	9a	9a	1-27a
10	10	0	0	0	9a	0	9a	9a	1-27a
10	11	1-18a	6a	6a	0	6a	0	0	0
11	00	0	0	0	9a	0	9a	9a	1-27a
11	01	1-18a	6a	6a	0	6a	0	0	0
11	10	1-18a	6a	6a	0	6a	0	0	0
11	11	1-15a	5a	5a	0	5a	0	0	0

The P-matrix between (X(k), S(k)) and $y_1(k)$, denoted by $[P_{XS-y}]$, is

$$[P_{XS-y}] = \begin{array}{c|cc} X(k), S(k) \backslash y_1(k) & 0 & 1 \\ \hline 00\ 00 & 1 & 0 \\ 00\ 01 & 1 & 0 \\ 00\ 10 & 1 & 0 \\ 00\ 11 & 1 & 0 \\ 01\ 00 & 1 & 0 \\ 01\ 01 & 1 & 0 \\ 01\ 10 & 1 & 0 \\ 01\ 11 & 1 & 0 \\ 10\ 00 & 1 & 0 \\ 10\ 01 & 1 & 0 \\ 10\ 10 & 1 & 0 \\ 10\ 11 & 3\mu' & 1-3\mu' \\ 11\ 00 & 1 & 0 \\ 11\ 01 & 3\mu' & 1-3\mu' \\ 11\ 10 & 3\mu' & 1-3\mu' \\ 11\ 11 & 3\mu' & 1-3\mu' \end{array}$$

From the $[P_{XS-S}]$, we note that any erroneous operation of components in the reliable binary counter may cause at the most one state variable to be in error. This single-error in the internal state can be corrected in the next step of time because of the

single-error correcting code which is used in the encoded states.

Now let us use the P-matrix method to analyze the above statement. If the present input state is zero, then the the P-matrix between S(k) and S^D(k+1) for X(k)=0 is

$[P_{S\text{-}S}^{(0)}]$ =

S(k) \ S^D(k+1)	000	001	010	011	100	101	110	111
000	1-21a	7a	7a	0	7a	0	0	0
001	1-21a	7a	7a	0	7a	0	0	0
010	1-21a	7a	7a	0	7a	0	0	0
011	0	0	0	7a	0	7a	7a	1-21a
100	1-21a	7a	7a	0	7a	0	0	0
101	0	0	0	7a	0	7a	7a	1-21a
110	0	0	0	7a	0	7a	7a	1-21a
111	0	0	0	6a	0	6a	6a	1-18a

The P-matrix between S(k) and w_1, w_2 or w_3 at time k+1 (relabel the a as a' at the time k+1 in order to distinguish the errors occur at two different time steps), denoted by $[P_{w_i}]$, i=1, 2, 3, respectively, is

$$[P_{w_i}]=[P_{S\text{-}S^{(0)}}].$$

$S^D(k+1)$ \ w_i	0	1
000	1-4a'	4a'
001	1-4a'	4a'
010	1-4a'	4a'
011	2a'	1-2a'
100	1-4a'	4a'
101	2a'	1-2a'
110	2a'	1-2a'
111	a'	1-a'

$$=$$

S(k) \ w_i	0	1
000	1-4a'	4a'
001	1-4a'	4a'
010	1-4a'	4a'
011	a'	1-a'
100	1-4a'	4a'
101	a'	1-a'
110	a'	1-a'
111	a'	1-a'

$i=1, 2, 3$

When the present input is one, the P-matrix between S(k) and w_i at time k+1 is

$[P_{w_i}]=$

$S(k)$ \ w_i	0	1
000	a'	1-a'
001	a'	1-a'
010	a'	1-a'
011	1-4a'	4a'
100	a'	1-a'
101	1-4a'	4a'
110	1-4a'	4a'
111	1-4a'	4a'

From the above matrices, we note that all the errors in the state variable occurring at time k will be corrected by the combinatorial circuit at time k+1, k=0,1,2,.... In other words the errors in the state variable do not propagate. By taking into account the single-error correcting ability of the combinatorial circuit for the internal state transitions, the total transition matrix for the binary counter with Bernoulli input with $p_0=p_1=1/2$ is:

$[P_{TB}]_2=$

$e(k)$ \ $e(k+1)$	000	001	010	011	100	101	110	111
1	0	0	0	0	0	0	0	0
000	1/2	0	0	0	0	0	0	1/2
001	1/2	0	0	0	0	0	0	1/2
010	1/2	0	0	0	0	0	0	1/2
011	$(1-3\mu')/2$	0	0	0	0	0	0	1/2
100	1/2	0	0	0	0	0	0	1/2
101	$(1-3\mu')/2$	0	0	0	0	0	0	1/2
110	$(1-3\mu')/2$	0	0	0	0	0	0	1/2
111	$(1-3\mu')/2$	0	0	0	0	0	0	1/2

($3\mu'/2$ in the first column)

If we use type 1 majority gate, instead of type 2 majority gate, in the reliable output decoder, then the total transition matrix becomes:

$$[P_{TB}] = S(k)$$

$e(k)$ \ $e(k+1)$	1	000	001	010	011	100	101	110	111
1	0	0	0	0	0	0	0	0	
000	3μ	$(1-3\mu)/2$	0	0	0	0	0	0	$(1-3\mu)/2$
001	3μ	$(1-3\mu)/2$	0	0	0	0	0	0	$(1-3\mu)/2$
010	3μ	$(1-3\mu)/2$	0	0	0	0	0	0	$(1-3\mu)/2$
011	$3\mu/2$	$1/2$	0	0	0	0	0	0	$(1-3\mu)/2$
100	3μ	$(1-3\mu)/2$	0	0	0	0	0	0	$(1-3\mu)/2$
101	$3\mu/2$	$1/2$	0	0	0	0	0	0	$(1-3\mu)/2$
110	$3\mu/2$	$1/2$	0	0	0	0	0	0	$(1-3\mu)/2$
111	$3\mu/2$	$1/2$	0	0	0	0	0	0	$(1-3\mu)/2$

The mean-time-to-first-error is

$$T_m(2) = S(0)$$

000	$4/3\mu'$
001	$4/3\mu'$
010	$4/3\mu'$
011	$(4-6\mu')/3\mu'$
100	$4/3\mu'$
101	$(4-6\mu')/3\mu'$
110	$(4-6\mu')/3\mu'$
111	$(4-6\mu')/3\mu'$

and

$$T_m(1) = S(0) \begin{cases} 000 & (4-6\mu)/9\mu \\ 001 & (4-6\mu)/9\mu \\ 010 & (4-6\mu)/9\mu \\ 011 & 4/9\mu \\ 100 & (4-6\mu)/9\mu \\ 101 & 4/9\mu \\ 110 & 4/9\mu \\ 111 & 4/9\mu \end{cases}$$

where $T_m(i)$ denotes the mean-time-to-first-error of the reliable binary counter when type i majority gate is used, $i \in \{1, 2\}$.

If we assume that the initial state is at S_0, then

$R_{\frac{T_m(2)}{T_m}}$ = reliability improvement of the binary counter of Fig. 6.7 with respect to the binary counter of Fig. 4.7

$$= \frac{4/3\mu'}{2/9a} = 6a/\mu'$$

$R_{\frac{T_m(1)}{T_m}}$ = reliability improvement of the binary counter of Fig. 6.7 but using type 1 majority gate instead of type 2 majority gate as the reliable output decoder with respect to the binary counter of Fig. 4.7

$$= \frac{(4-6\mu)/9\mu}{2/9a} = (2-3\mu)a/\mu$$

$R_{\frac{T_m(2)}{T_m(1)}}$ = reliability improvement of the reliable binary counter using type 2 majority gate (Fig. 6.7) with respect to the same binary counter but using type 1 majority gate as the reliable output decoder.

$$= \frac{4/3\mu'}{(4-6\mu)/9\mu} = \frac{6\mu/\mu'}{2-3\mu}$$

If $\mu \neq \mu' = a \ll 1$, then

$$R_{\frac{T_m(2)}{T_m}} = 6$$

$$R_{\frac{T_m(1)}{T_m}} \tilde{=} 2$$

$$R_{\frac{T_m(2)}{T_m(1)}} \tilde{=} 3$$

where the symbol $\tilde{=}$ denotes approximate equality.

From this analysis we note that the mean-time-to-first-error of the reliable binary counter using type 2 (type 1) majority gate is 6 (2) times the mean-time-to-first-error of the original binary counter (Fig. 4.7). Furthermore, the mean-time-to-first-error of the reliable counter using type 2 majority gate is 3 times the mean-time-to-first-error of the reliable binary counter using type 1 majority gate.

The reason why the reliability of the binary counter with type 2 majority gate is better than that with type 1 majority is given as follows:

When type 2 majority gate is used as the decoder and if the output variable is one when it should be zero, i.e. when all the three input diodes of the AND gate are conducting instead of non-conducting the probability of erroneous output is equal to μ^3 and is equal to

$3\mu'$ if the output is zero when it should be one. When type 1 majority gate is used instead of type 2 majority gate, if any one of the three input diodes of the OR gate is conducting instead of non-conducting, the probability of erroneous output is equal to 3μ. If the output is one when it should be zero, the probability of erroneous output is equal to $(\mu')^3$. For t=1, $\mu^3 = (\mu')^3 = 0$. Thus the ratio of probabilities of error (i.e. the failure rate in our case) is equal to $(3\mu'/4) / 9\mu/4 = 1/3$ and $R_{\frac{T_m(2)}{T_m(1)}} = 3$. (We assume $p_0 = p_1 = 1/2$, thus $p(y_1=1) = 1/4$ and $p(y_1=0) = 3/4$.)

Example 6.2. Design a sequential circuit with single input and two output variables. The first output variable is 1 whenever even number of input pulses are encountered; the second output variable is 1 whenever 4λ ($\lambda = 1, 2, \ldots$) input pulses are encountered. No single -error in the circuit operation can cause the output states to be in error. The probability that two components fail at the same instant of time is assumed zero.

The flow table of the sequential circuit is shown in Fig. 6.8.

$x_1(k)$ / $S(k)$	0	1
S_0	S_0 / 00	S_1 / 00
S_1	S_1 / 00	S_2 / 10
S_2	S_2 / 00	S_3 / 00
S_3	S_3 / 00	S_0 / 11

Fig. 6.8. Flow table of the designed sequential circuit.

Entries mean: next internal / first output, second output.

The modified flow table is shown in Fig. 6.9.

$s_1(k), s_2(k), s_3(k), s_4(k), s_5(k)$ \ $x_1(k)$	0	1
0 0 0 0 0	00000/00000/00	11100/00000/00
0 0 0 0 1	00000/00000/00	11100/00000/00
0 0 0 1 0	00000/00000/00	11100/00000/00
0 0 1 0 0	00000/00000/00	11100/00000/00
0 1 0 0 0	00000/00000/00	11100/00000/00
1 0 0 0 0	00000/00000/00	11100/00000/00
1 1 1 0 0	11100/00000/00	00111/10110/10
0 1 1 0 0	11100/00000/00	00111/10110/10
1 0 1 0 0	11100/00000/00	00111/10110/10
1 1 0 0 0	11100/00000/00	00111/10110/10
1 1 1 1 0	11100/00000/00	00111/10110/10
1 1 1 0 1	00111/00000/00	11011/00000/00
0 0 1 1 1	00111/00000/00	11011/00000/00
0 0 0 1 1	00111/00000/00	11011/00000/00
0 0 1 0 1	00111/00000/00	11011/00000/00
0 0 1 1 0	00111/00000/00	11011/00000/00
0 1 1 1 1	00111/00000/00	11011/00000/00
1 0 1 1 1	11011/00000/00	00000/11001/11
1 1 0 1 1	11011/00000/00	00000/11001/11
0 1 0 1 1	11011/00000/00	00000/11001/11
1 0 0 1 1	11011/00000/00	00000/11001/11
1 1 0 0 1	11011/00000/00	00000/11001/11
1 1 0 1 0	11011/00000/00	00000/11001/11
1 1 1 1 1		

Fig. 6.9. Modified flow table of Fig. 6.8.

Where S_0=00000, S_1=11100, S_2=00111 and S_3=11011.

The modified sequential circuit will have single-error correction ability in state transitions and encoded output states. The switching equations of the outputs of the unit delay elements and the output variables are

$$
\begin{aligned}
s_1(k+1) = {} & \overline{x_1(k)}[\,{}_2(k)s_3(k)s_1(k)\overline{s_4(k)} + s_1(k)s_2(k)s_3(k)\overline{s_5(k)} \\
& + s_1(k)s_2(k)\overline{s_4(k)}\,\overline{s_5(k)} + s_1(k)s_3(k)\overline{s_4(k)}\,\overline{s_5(k)} + \\
& s_2(k)s_3(k)\overline{s_4(k)}\,\overline{s_5(k)} + s_1(k)s_2(k)s_4(k)s_5(k) + \\
& s_1(k)s_2(k)\overline{s_3(k)}s_4(k) + s_1(k)s_2(k)\overline{s_3(k)}s_5(k) + \\
& s_1(k)\overline{s_3(k)}s_4(k)s_5(k) + s_2(k)\overline{s_3(k)}s_4(k)s_5(k)] + \\
& x_1(k)[\overline{s_1(k)}\,\overline{s_2(k)}\,\overline{s_3(k)}s_4(k) + \overline{s_1(k)}\,\overline{s_2(k)}s_3(k)\overline{s_5(k)} + \\
& \overline{s_1(k)}s_2(k)\overline{s_3(k)}\,\overline{s_4(k)}\,\overline{s_5(k)} + s_1(k)\overline{s_3(k)}\,\overline{s_4(k)}\,\overline{s_5(k)} + \\
& \overline{s_2(k)}\,\overline{s_3(k)}\,\overline{s_4(k)}\,\overline{s_5(k)} + \overline{s_1(k)}s_3(k)s_4(k)s_5(k) + \\
& \overline{s_2(k)}s_3(k)s_4(k)s_5(k) + \overline{s_1(k)}\,\overline{s_2(k)}s_3(k)s_4(k) + \\
& \overline{s_1(k)}\,\overline{s_2(k)}s_3(k)s_5(k) + \overline{s_1(k)}\,\overline{s_2(k)}s_4(k)s_5(k)\,] \qquad (6.13)
\end{aligned}
$$

$$
s_2(k+1) = s_1(k+1) \qquad (6.14)
$$

$$
\begin{aligned}
s_3(k+1) = {} & \overline{x_1(k)}[\overline{s_1(k)}s_3(k)s_4(k)s_5(k) + \overline{s_2(k)}s_3(k)s_4(k)s_5(k) + \\
& \overline{s_1(k)}\,\overline{s_2(k)}s_3(k)s_4(k) + \overline{s_1(k)}\,\overline{s_2(k)}s_3(k)s_5(k) + \\
& \overline{s_1(k)}\,\overline{s_2(k)}s_4(k)s_5(5)\,] + x_1(k)[\overline{s_1(k)}\,\overline{s_2(k)}\,\overline{s_3(k)}\,\overline{s_4(k)} + \\
& \overline{s_1(k)}\,\overline{s_2(k)}\,\overline{s_3(k)}\,\overline{s_5(k)} + \overline{s_1(k)}s_2(k)\overline{s_4(k)}\,\overline{s_5(k)} + \\
& \overline{s_1(k)}\,\overline{s_3(k)}\,\overline{s_4(k)}\,\overline{s_5(k)} + \overline{s_2(k)}\,\overline{s_3(k)}\,\overline{s_4(k)}\,\overline{s_5(k)}\,] + \\
& s_1(k)s_2(k)s_3(k)\overline{s_4(k)} + s_1(k)s_2(k)s_3(k)\overline{s_5(k)} + \\
& s_1(k)s_2(k)\overline{s_4(k)}\,\overline{s_5(k)} + s_1(k)s_3(k)\overline{s_4(k)}\,\overline{s_5(k)} + \\
& s_3(k)s_2(k)\overline{s_4(k)}\,\overline{s_5(k)} \qquad (6.15)
\end{aligned}
$$

$$s_4(k+1) = x_1(k)[s_1(k)s_2(k)\overline{s_4(k)}s_5(k) + s_1(k)s_2(k)\overline{s_3(k)}s_4(k) +$$
$$s_1(k)s_2(k)\overline{s_3(k)}s_5(k) + s_1(k)\overline{s_3(k)}s_4(k)s_5(k) +$$
$$s_2(k)\overline{s_3(k)}s_4(k)s_5(k) + \overline{s_1(k)}\,\overline{s_2(k)}s_3(k)s_4(k) +$$
$$\overline{s_1(k)}\,\overline{s_2(k)}s_3(k)s_5(k) + \overline{s_1(k)}\,\overline{s_2(k)}s_4(k)s_5(k) +$$
$$\overline{s_1(k)}s_3(k)s_4(k)s_5(k) + \overline{s_2(k)}s_3(k)s_4(k)s_5(k)] +$$
$$x_1(k)[\overline{s_2(k)}\,\overline{s_3(k)}\,\overline{s_4(k)}\,\overline{s_5(k)} + \overline{s_1(k)}\,\overline{s_3(k)}\,\overline{s_4(k)}\,\overline{s_5(k)} +$$
$$\overline{s_1(k)}\,\overline{s_2(k)}\,\overline{s_4(k)}\,\overline{s_5(k)} + \overline{s_1(k)}\,\overline{s_2(k)}\,\overline{s_3(k)}\,\overline{s_5(k)} +$$
$$\overline{s_1(k)}\,\overline{s_2(k)}\,\overline{s_3(k)}\,\overline{s_4(k)} + s_1(k)s_2(k)\overline{s_4(k)}\,\overline{s_5(k)} +$$
$$s_1(k)s_3(k)\overline{s_4(k)}\,\overline{s_5(k)} + s_2(k)\overline{s_4(k)}\,s_3(k)\overline{s_5(k)} +$$
$$s_1(k)s_2(k)s_3(k)\overline{s_5(k)} + s_1(k)s_2(k)s_3(k)\overline{s_4(k)}] \qquad (6.16)$$

$$s_5(k+1) = s_4(k+1) \qquad (6.17)$$

$$z_1(k) = x_1(k)[s_1(k)s_2(k)s_3(k)\overline{s_4(k)} + s_1(k)s_2(k)s_3(k)\overline{s_5(k)} +$$
$$s_1(k)s_2(k)\overline{s_4(k)}\,\overline{s_5(k)} + s_1(k)s_3(k)\overline{s_4(k)}\,\overline{s_5(k)} +$$
$$s_2(k)s_3(k)\overline{s_4(k)}\,\overline{s_5(k)} + s_1(k)s_2(k)s_4(k)s_5(k) +$$
$$s_1(k)s_2(k)\overline{s_3(k)}s_4(k) + s_1(k)s_2(k)\overline{s_3(k)}s_5(k) +$$
$$s_1(k)\overline{s_3(k)}s_4(k)s_5(k) + s_2(k)\overline{s_3(k)}s_4(k)s_5(k)] \qquad (6.18)$$

$$z_2(k) = x_1(k)[s_1(k)s_2(k)s_4(k)s_5(k) + s_1(k)s_2(k)\overline{s_3(k)}s_4(k) +$$
$$s_1(k)s_2(k)\overline{s_3(k)}s_5(k) + s_1(k)\overline{s_3(k)}s_4(k)s_5(k) +$$
$$s_2(k)\overline{s_3(k)}s_4(k)s_5(k)] \qquad (6.19)$$

$$z_3(k) = x_1(k)[s_1(k)s_2(k)s_3(k)\overline{s_4(k)} + s_1(k)s_2(k)s_3(k)\overline{s_5(k)} +$$
$$s_1(k)s_2(k)\overline{s_4(k)}\,\overline{s_5(k)} + s_1(k)s_3(k)\overline{s_4(k)}\,\overline{s_5(k)} +$$
$$s_2(k)s_3(k)\overline{s_4(k)}\,\overline{s_5(k)}] \qquad (6.20)$$

$$z_4(k) = z_3(k) \qquad (6.21)$$

$$z_5(k) = z_2(k) \qquad (6.22)$$

The switching equations for the outputs of the first layer of the reliable output decoder are

$$y_{1,1}(k) = y_{1,2}(k) = y_{1,3}(k) = z_1(k)z_3(k)z_4(k)\overline{z_5(k)} + z_1(k)\overline{z_2(k)}z_3(k)z_4(k) +$$
$$\overline{z_2(k)}z_3(k)z_4(k)z_5(k) + z_1(k)\overline{z_2(k)}z_4(k)\overline{z_5(k)} +$$
$$z_1(k)\overline{z_2(k)}z_3(k)\overline{z_5(k)} + z_1(k)z_2(k)\overline{z_3(k)}z_5(k) +$$
$$z_1(k)z_2(k)\overline{z_4(k)}z_5(k) + z_1(k)z_2(k)\overline{z_3(k)}\,\overline{z_4(k)} +$$
$$z_1(k)\overline{z_3(k)}\,\overline{z_4(k)}z_5(k) + z_2(k)\overline{z_3(k)}\,\overline{z_4(k)}z_5(k) \qquad (6.23)$$

$$y_{2,1}(k) = y_{2,2}(k) = y_{2,3}(k) = z_1(k)z_2(k)\overline{z_3(k)}z_5(k) +$$
$$z_1(k)z_2(k)\overline{z_4(k)}z_5(k) + z_1(k)z_2(k)\overline{z_3(k)}\,\overline{z_4(k)} +$$
$$z_1(k)\overline{z_3(k)}\,\overline{z_4(k)}z_5(k) + z_2(k)\overline{z_3(k)}\,\overline{z_4(k)}z_5(k) \qquad (6.24)$$

Type 2 majority gates are used in the second layer of the reliable output decoder. The switching equations of the output variables are

$$y_i(k) = [y_{i,1}(k) + y_{i,2}(k)][y_{i,1}(k) + y_{i,3}(k)][y_{i,2}(k) + y_{i,3}(k)]$$
$$i \in \{1, 2\} \qquad (6.25)$$

Thus the sequential circuit for the specification can be obtained, utilizing equations (6.13) to (6.25).

CHAPTER VII

REALIZATION WITH MAXIMUM FEEDBACK-FREE TAIL MACHINE

The purpose of this chapter is to derive a method of testing whether or not a given sequential machine represented by a flow table can be realized by a feedback-free machine. If a given sequential machine cannot be realized by a feedback-free machine, then we decompose the sequential machine into submachines in series. The first submachine (front machine) contains all the feedback paths, and the second, third, ... submachines (tail machine) are feedback-free. The front machine must be kept as small as possible, i.e. it must contain as few unit delay elements as possible. The states and outputs of the series combination of the submachines are equivalent to those of the original machine.

There are 3 reasons for decomposing every sequential machine with a large number of states into several submachines in series:

1. Each submachine will have a smaller number of states.

2. It is easier to handle a sequential machine with a small number of states than the one with a larger number of states, especially when error-correcting codes are employed to eliminate the prescribed number of simultaneous errors which appear in the state transitions.

3. If it is required that no errors in the sequential machine can be allowed to stay in it permanently, then the front machine can be more reliable while the tail machine can be less reliable. The state transitions in the front machine can be made more reliable by applying error-correcting codes in its secondary state assignment.

Before presenting the method, a few definitions must first be introduced.

7.1. Definitions

In this section the basic material [17] which is used in the rest of this chapter is presented.

Defintion 7.1. A partition π, over a set S of a sequential machine M_1, is

a collection of disjoint subsets of S, called blocks, such that their set union is S and the intersections of any two subsets are empty.

A block in the partition of the set S is denoted by $\overline{S}$. $\#(\pi)$ is the number of blocks in π and $L(\pi)$ denotes the number of states in the largest block in π. Two states which appear in the same block of a partition π are said to be identified by π. We denote the trivial partition, zero partition, where each block contains a single state by 0, and the trivial partition having a single block by I.

Definition 7.2. A partition π on S of M_1 is said to have the substitution property (S.P.) if each input maps blocks of π into blocks of π.

Definition 7.3. A partition pair (π , π') on M_1 is an ordered pair of partitions on S if each input maps blocks of π into blocks of π', where π, π' are partitions on M_1, and π is the front partition of π'.

Definition 7.4. If π is a partition on S of M_1, let

$$m(\pi) = \cup \{\pi' \,/\, (\pi , \pi') \text{ is a partition pair on } M_1 \}$$

and

$$M(\pi) = \bigcap \{\pi' \,/\, (\pi' , \pi) \text{ is a partition pair on } M_1 \}$$

The $m(\pi)$ is an operator which gives the minimum tail partition and $M(\pi)$ is the operator which gives the maximum front partition, Where minimum and maximum number of blocks in the tail and front partitions respectively.

A machine M_1 of g internal states is represented by an one step transition matrix whose row and column headings are internal states $(S_0, S_1, \ldots, S_{g-1})$. The (i, j) entry is the input state which causes the internal state of the machine M_1 to change from the present state S_i to next internal state j, and output state in one step. In the following we shall consider the state behavior of M_1 only.

Definition 7.5. Two rows , i_1 and i_2, in the i-th order transition matrix are in the same block of the i-th front partition if the (i_1, j) and (i_2, j), for all $j \in \{0, 1, \ldots, g-1\}$, entries are the same whenever they are specified.

7.2. Previous Results

The relationship between definite-event machine and feedback-free

machine has been studied by Arden[31] who stated that every definite-event machine is representable as a feedback-free machine and vice-versa. Hartmanis and Stearns [17] applied the partition method and found that a sequential machine which can be realized by a machine without feedback (feedback-free machine) is equivalent to $m^k(I)=0$ ($M^k(0)=I$) $k = g$. where g is the number of states of the sequential machine. The $M^k(0)$ is the front partition of $M^{k-1}(0)$, $k \in\{1, 2, \ldots, g\}$ $M^0(0)=0$. Hartmanis and Stearns have also shown that if $M^g(0)=\pi_2 \neq I$, then a given sequential machine can be realized with feedback front machine defined by π_2 (each block of π_2 is a state of the front machine) followed by a feedback-free tail machine, defined by $\lambda_{g-1}, \lambda_{g-2}, \ldots, \lambda_0$. $\lambda_{g-1}, \lambda_{g-2}$, ..., and λ_0 are obtained from $\pi_2 \cap \lambda_{g-1}=\pi_{g-1}$, $\pi_{g-1} \cap \lambda_{g-2}=\pi_{g-2}, \ldots,$ $\pi_1 \cap \lambda_0=0$. Kohavi [32] developed a technique where, first, an equivalent machine M_1' is obtained using a state splitting technique (implication graph) and then an S. P. partition is found for M_1' (if it exists). He used an implication graph to find S.P. partition, π_3, and then to decompose the machine M_1' into two submachines in series such that the front machine M_2 is defined by the S.P. partition and the tail machine M_3 is defined by λ_3 such that $\pi_3 \cap \lambda_3 = 0$. In general neither of the submachines (M_2 and M_3) is a feedback-free machine.

To the best of our knowledge thus far, no work has been done in finding the maximum feedback-free tail machine by using transition matrices.

7.3. A Method for Finding the Maximum Feedback-free Tail Machine - Successive Transition Matrices Method

In this section a method for finding the maximum feedback-free tail machine is developed. The transition matrix is the mathematical counterpart of the flow table, and it enables one to carry out mechanically a number of operations, which, in the flow table, can be carried out visually. The transition matrix is therefore advantageous whenever the operations cannot be carried out visually, or whenever the flow table is complex to the extent that the visual approach is difficult.

7.3.1. A Method of Generating the Front Partitions

The first, second, and i-th front partitions of a sequential machine M_1 of g states can be obtained by examining the first, second, . . . , and i-th order transition matrices respectively. The first, second, . . . , i-th order transition matrices can be generated by raising the power of $[M]$, where $[M]$ is the one step transition matrix (defined by Gill [33]) of M_1. If there is an input "a" which causes the transition from state S_i to state S_j in one step, then "a" appear in the (i, j) entry of $[M]$. The k-th order transition matrix, $[M]^k$, is the k-step transition between any two states of M_1.

In a k-th order transition matrix, if there are any two rows, say i-th and j-th rows, whose entries are equal, then M_1 will give us the same states after applying the same input sequence of length k if M_1 starts at either S_i or S_j. If the entries of all rows are the same, then M_1 will give us the same state after applying the same input sequence of length k, regardless of the starting state, i.e. M_1 is realizable as a feedback-free machine. If all rows in $[M]$ are different, then it can be shown, by utilizing matrix theory, that all rows are different in $[M]^k$, $k \geq 2$. If there are at least two rows, say row i and row j , are equal in $[M]$ then at least the entries of these two rows are equal in $[M]^2$, due to the entries in row i and row j are equal. Raising the power of $[M]$ until the entries of all rows are equal or no more new equal rows are created, then the process of generating successive transition matrices is complete.

Let $[M]^k$ be the highest order of transition matrix which has all rows equal or $[M]^{k+1}$ does not create any more new row. If all rows in the k-th order transition matrix are the same, then the sequential machine M_1 is feedback-free machine realizable. If some of the rows are different, then equivalent class of states (rows with the same entries belong to the same equivalent class of states) defines a state in the feedback front machine; the number of states in the front machine equals the number of equivalent classes of states in the h-th order transition matrix ($h \in \{1, 2, \ldots, (k-1)\}$) defines a block in the h-th front partition.

Since we have to distinguish the states in π_k to form π_{k-1}, a partition λ_{k-1} must be obtained such that

$$\pi_k \cap \lambda_{k-1} = \pi_{k-1}$$

Similarly we have

$$\pi_{k-1} \cap \lambda_{k-2} = \pi_{k-2}$$

$$\vdots$$

$$\pi_1 \cap \lambda_0 = 0$$

These partitions, $\lambda_{k-1}, \lambda_{k-2}, \ldots, \lambda_0$, define the states of the submachines which are feedback-free. If $\pi_k = I$, then M_1 is feedback-free machine realizable. If $\pi_k \neq I$, then the machine is realized by a feedback front machine followed by a feedback-free tail machine. The number of unit delay elements in the feedback front machine is equal to $(\log_2 \#(\pi_k)$ and in the h-th feedback-free submachine it is equal to $(\log_2 \#(\lambda_h)), h \in \{1,2, \ldots, (k-1)\}$.

7.3.2. An Algorithm for Finding the Front Partitions, the Partitions for the Submachines

The flow-chart representation of an algorithm for finding the front partitions and the partitions for the submachines is shown in Fig. 7.1. It is a convenient summary of our previous detailed discussion.

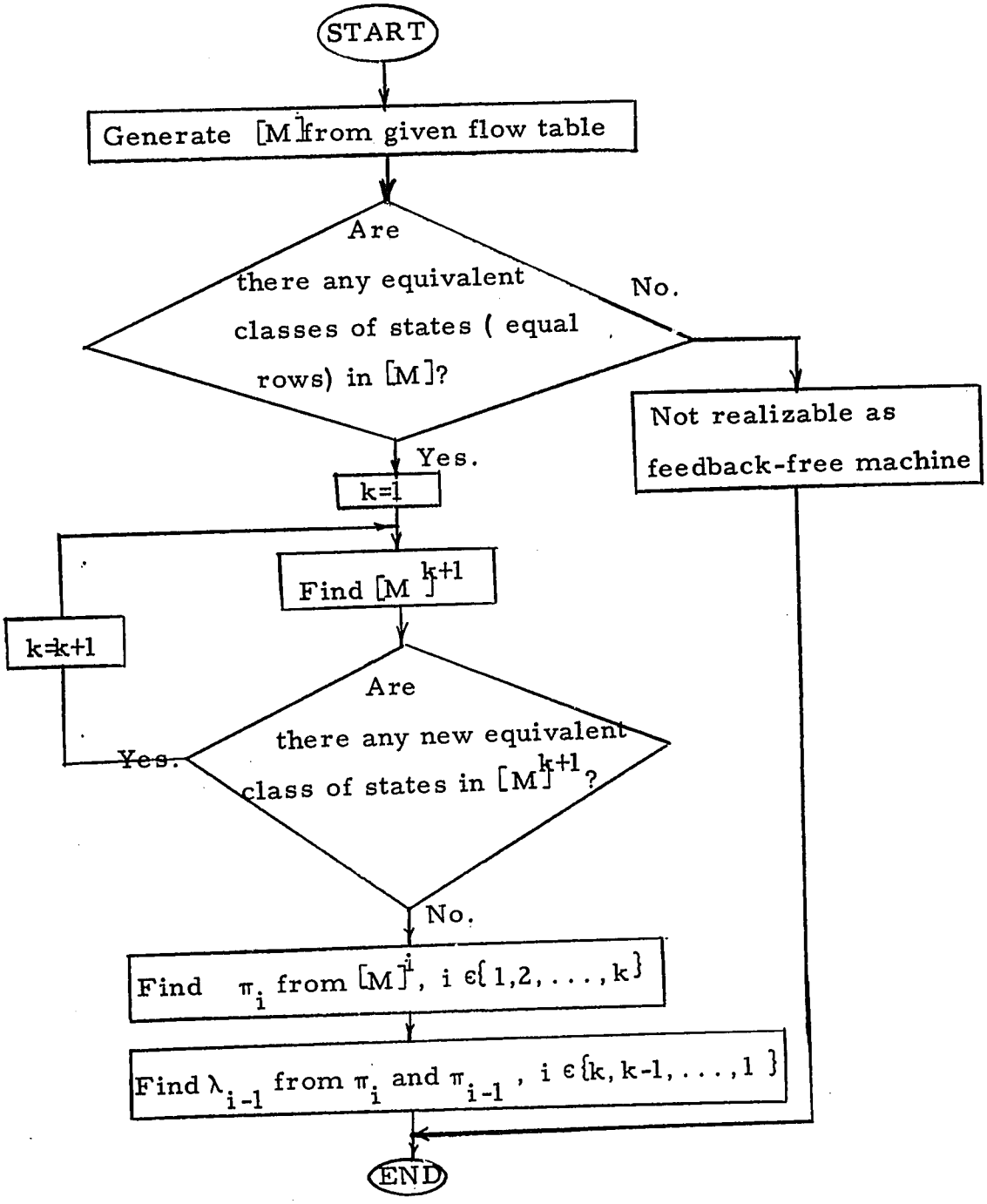

Fig. 7.1 The flow chart of algorithm for finding front partitions and partitions for maximum feedback-free machine realization.

7.4. Example

Given a sequential machine M as shown in Fig. 7.2.

X(k) / S(k)	a	b	c
S_0	S_3	S_2	S_1
S_1	S_0	S_2	S_1
S_2	S_1	S_2	S_0
S_3	S_3	S_2	S_1

Fig. 7.2. Machine M

From the flow table of M, we obtain the following transition matrix

$$[M] = \begin{array}{c} \\ S_0 \\ S_1 \\ S_2 \\ S_3 \end{array} \begin{array}{c} \begin{array}{cccc} S_0 & S_1 & S_2 & S_3 \end{array} \\ \begin{bmatrix} 0 & c & b & a \\ a & c & b & 0 \\ c & a & b & 0 \\ 0 & c & b & a \end{bmatrix} \end{array}$$

where the element 0 in the (i, j) entry denotes that there is no input which will cause the state of the machine to change from state S_i to state S_j.

$$[M]^2 = [M].[M] =$$

	S_0	S_1	S_2	S_3
S_0	ca+bc	cc+ba+ac	ca+bb+ab	aa
S_1	ca+bc	ac+cc+ba	ab+cb+bb	aa
S_2	aa+bc	cc+ac+ba	cb+ab+bb	ca
S_3	ca+bc	cc+ac+ba	cb+bb+ab	aa

$$[M]^3 = [M]^2.[M] =$$

	S_0	S_1	S_2	S_3
S_0	cca+baa +aca	cac+bcc+ (cc+ba+ac)c	(ca+bc+cc+ ba+ac+cb+ bb+ab+aa)b	(ca+bc+aa)a
S_1	''	''	''	''
S_2	''	''	''	''
S_3	''	''	''	''

Front partitions are

$$\pi_1 = \{\overline{S_0, S_3} ; \overline{S_1}; \overline{S_2}\}$$

$$\pi_2 = \{\overline{S_0, S_1, S_3} ; \overline{S_2}\}$$

and $$\pi_3 = \{\overline{S_0, S_1, S_2, S_3}\}$$

Partitions for feedback-free submachines are

$$\pi_3 \bigcap \lambda_2 = \pi_2 \qquad \lambda_2 = \{\overline{S_0, S_1, S_3}; \overline{S_2}\}$$

$$\pi_2 \bigcap \lambda_1 = \pi_1 \qquad \lambda_1 = \{\overline{S_0, S_3} ; \overline{S_1, S_2}\}$$

$$\pi_1 \bigcap \lambda_0 = 0 \qquad \lambda_0 = \{\overline{S_0, S_2} : \overline{S_1, S_3}\}$$

Since $\pi_3 = I$, thus this sequential machine is realizable as a feedback-free machine.

CHAPTER VIII

CONCLUSIONS AND SUGGESTIONS FOR FURTHER STUDY

A simplified method for finding Markov models for probabilistic sequential circuits with Bernoulli or Markov inputs has been presented. The z-transform and signal flow-graph techniques have been found to be very useful in establishing the reliability of this class of circuits. It has also been shown that a more reliable sequential circuit with prescribed reliability can be obtained by applying error-correcting codes and the redundancy technique to the secondary state assignments, the output states and the output decoder. Finally, the transition matrix method was found to be very useful in testing for maximum feedback-free machine realization of a given sequential circuit.

The following problems may be worthy of further investigation:

1. Application of information theory to the analysis and synthesis of probabilistic sequential circuits.

2. Development of synthesis methods for a reliable sequential circuit with minimum number of components utilizing probabilistic logical matrix.

REFERENCES

(1) J. von Neumann, " Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Component ", in " Automata Studies ", C.E. Shannon and J. McCarthy, (Ed.), Princeton University Press, N.J. No. 34, pp. 43-97, 1956.

(2) E.F. Moore and C.E. Shannon, " Reliable Circuits Using Less Reliable Relays ", Part I and II, Journal Franklin Institute, Vol 262, pp. 198-208 and 281-297, Sept. and Oct. 1956.

(3) O. Boshko, G.S. Glinski and J. Therrien, " Reliable Digital Networks from Unreliable Components ", Trans. EIC, Vol. 5, No. 3, pp. 184-201, 1961.

(4) S. Winograd and J.D. Cowan, " Reliable Computation in the Presence of Noise ", The MIT Press, Cambridge, Mass., 1963.

(5) C.L. Sheng and I.N. Chen, " On Redundancy Techniques for Reliable Computation- a Unified Viewpoint from Enlarging of Space ", T.R. No. 66-13, Dept. of E.E., The university of Ottawa, Ottawa, Canada, 1966.

(6) S. Amarel and J.A. Brzozowski, " Theoretical Considerations on Reliability Properties of Recursive Triangular Switching Networks", in " Redundancy Techniques for Computing Systems", Wilcox and Mann, (Ed.), Spartan Book, Washington, D.C., pp. 70-128, 1962

(7) J.G. Tryon, " Quadded Logic ", in "Redundancy Techniques for Computing Systems ", Wilcox and Mann, (Ed.), Spartan Book, Washington, D.C., pp. 205-228, 1962.

(8) K. Udagawa and Y. Inagaki, " Probabilistic Studies of Logical Circuits and the Synthesis of Reliable Circuit Using Probabilistic Logical Matrix", Journal of the Institute of Electrical Communication Engineers in Japan, Vol. 46, No.11, pp. 195-206, Nov. 1963.

(9) V.I. Levin, " Probabilistic Analysis of Combination Systems and Their Reliability ", Engineering Cybernetics, No. 6, pp. 78-84, Nov.-Dec. 1964.

(10) G.S. Glinski and Who-Kee Chung, " A Markov Model for Reliability of Probabilistic Sequential Circuit with Bernoulli Inputs", Proceedings of the Sixth Annual Allerton Conference on Circuit and System Theory, University of Illinois, Urbana, pp. 568-577, Oct. 2-4, 1968.

(11) G.S. Glinski and Who-Kee Chung, " A Markov Model for Reliability of Probabilistic Sequential Circuits with Markov Inputs", Proceedings of the Seventh Annual Allerton Conference on Circuit and System Theory, University of Illinois, Urbana, pp. 690-691, Oct. 8-10, 1969.

(12) G.S. Glinski, B.S. Sotskov and H.S. Weissmann, " Reliability", Proceedings of the Second Congress of the International Federation of Automatic Control, Basle, Switzerland, pp. 701-708, 1963.

(13) K. Udagawa and Y. Inagaki, " Probabilistic Studies of Sequential Circuits and Estimation of Their Reliability Using Markov Chain ", Journal of the Institute of Electrical Communication Engineers in Japan, Vol. 47, No. 4, pp. 276-286, April 1964.

(14) M.C. Pease, " Methods of Matrix Algebra ", Academic Press, N.Y., 1965.

(15) J.G. Kemeny and J.L. Snell, " Finite Markov Chains ", D. Van Nostrand Co., Inc., Princeton, N.J., 1960.

(16) D.B. Armstrong, " A General Method for Applying Error-correction to Synchronous Digital Systems ", Bell System Technical Journal, Vol. 40, No. 2, pp. 577-593, March 1961.

(17) J. Hartmanis and R.E. Stearns, " Algebraic Structure Theory of Sequential Machines ", Prentice-Hall, Englewood Cliffs, N.J. 1966.

(18) G.H. Mealy, " Methods for Synthesizing Sequential Circuit ", Bell System Technical Journal, Vol. 34, pp. 1045-1078, 1955.

(19) L. Tin Htun, " Reliability Prediction Techniques for Complex Systems ", IEEE Transaction on Reliability, Vol. R-15, No. 2, pp. 58-69, Aug. 1966.

(20) R.A. Howard, " Dynamic Programming and Markov Processes ", Technology Press of MIT and John Wiley and Sons, Inc., N.Y. 1960.

(21) I. Bazovsky, " Reliability Theory and Practice ", Prentice-Hall, Englewood Cliffs, N.J. 1961.

(22) A. Papoulis, " Probability, Random Variables, and Stochastic Processes ", McGraw-Hill Book Co., N.Y. 1965.

(23) S.J. Mason and H.J. Zimmermann, " Electronic Circuits, Signal, and Systems ", John Wiley and Sons, Inc., N.Y. 1960.

(24) G.S. Glinski, " Information Theory of Computing Channels ", Symposium on the Educational Implication of the Expanding Frontiers in the Communication Sciences, Kingston, Ont., Canada, June 1963.

(25) V.I. Levin, " Probability and Reliability Analysis of Finite Automata ", Engineering Cybernetics, No., pp. 53-61, Sept.-Oct. 1965.

(26) G.D. Bruce and K.S. Fu, " A Model for Finite-state Probabilistic Systems ", Proceedings of the First Annual Allerton Conference on Circuit and System Theory, University of Illinois, Urbana, pp. 632-651, 1963.

(27) W. Feller, " An Introduction to Probability Theory and Its Applications ", Vol. I, John Wiley and Sons, Inc., N.Y. 1957.

(28) L. Takács, " Stochastic Processes ", Methuen and Co. Ltd., London 1960.

(29) W.W. Peterson, " Error-correcting Codes ", MIT Press, Cambridge, Mass., and John Wiley and Sons, Inc., N.Y. 1961.

(30) E.J. McCluskey, " Introduction to the Theory of Switching Circuits ", McGraw-Hill, N.Y. 1965.

(31) D.N. Arden, " Delayed-logic and Finite-state Machines ", Proceedings of the Second Annual Symposium on Switching Circuit Theory and Logical Design, Detroit, Mich., pp. 133-151, 1961.

(32) Z. Kohavi, " Secondary State Assignment for Sequential Machines ", Transaction on Electronic Computers, Vol. EC-13, No. 3, pp. 193-203, June 1964.

(33) A. Gill, " Introduction to the Theory of Finite-state Machines ", Mcgraw-Hill Book Co., N.Y. 1962.

VITA

Name: Who-Kee Chung.

Born: 8 Jan. 1938, Hong Kong.

Educated:

Primary : Hong Kong.

Secondary: Clementi Middle School, Hong Kong.

University:

(a) National Taiwan University, Taipei, Taiwan, China. 1961, B. Sc. in E.E.

(b) University of New Brunswick, Fredericton, New Brunswick, 1964, M. Sc. in E.E.

END OF REEL