National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

STUDIES ON A CONTINUOUS MARKOV MODEL FOR PROBABILISTIC FAULT ANALYSIS IN VLSI SEQUENTIAL CIRCUITS USING COMPUTER SIMULATION

BY
KAM LUNG WONG

A THESIS SUBMITTED TO THE
SCHOOL OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF APPLIED SCIENCES

OTTAWA-CARLETON INSTITUTE FOR ELECTRICAL ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING
FACULTY OF ENGINEERING

UNIVERSITY OF OTTAWA
November 1990

© Kam Lung Wong, Ottawa, Canada, 1990

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-67998-0

Canada

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

TABLE OF CONTENTS

TABLE OF CONTENTS iii
LIST OF FIGURES v
LIST OF TABLES vi
ACKNOWLEDGEMENTS vii
ABSTRACT viii
1.0 INTRODUCTION 1
1.1 The contribution 2
1.2 Scope 3
2.0 DETERMINISTIC TESTING 4
2.1 Test generation for combinational circuit 4
2.1.1 Fault Table method 4
2.1.2 Boolean Difference method 4
2.1.3 Path Sensitization (PS) method 5
2.1.4 Equivalent Normal Form (ENF) method 6
2.1.5 D-algorithm 6
2.2 Test generation for sequential circuit 9
2.2.1 Iterative array method 10
2.2.2 State table verification method 12
3.0 RANDOM TESTING 16
3.1 Test generation for combinational circuit 17
3.1.1 NAND tree network 17
3.1.2 Propagation probability 21
3.1.3 Detection surface 23
3.1.4 Detection probability 25
3.2 Test generation for sequential circuit 27
3.2.1 Weighted random test generation (WRTG) ... 27
3.2.2 Graph reduction 27
3.2.3 Error latency 28
4.0 CONTINUOUS MARKOV MODEL FOR RANDOM TESTING OF SEQUENTIAL CIRCUITS 31
4.1 Model definitions 32
4.2 Solving for model parameters 35
4.3 Fault testing strategy 41

5.0 EXPERIMENTATION.....47
5.1 Algorithm for calculating testing time.....47
5.2 Example 1: 8-bit SIPO shift register.....48
5.3 Example 2: 4-bit up/down counter.....54
6.0 DISCUSSIONS.....58
6.1 Observations.....58
6.2 Conclusions.....62
BIBLIOGRAPHY.....63
APPENDIX A1: FORTRAN program to simulate DM74LS164.....70
APPENDIX A2: FORTRAN program to simulate DM74LS191.....75
APPENDIX A3: FORTRAN program to calculate testing time.....80
APPENDIX B1: Solving of steady state probabilities.....87
APPENDIX B2: Solving of transition probabilities.....96
Papers by the Author on which Portions of the Present Thesis is Based.....105

LIST OF FIGURES

2.1 Sequential logic circuit representation.....9
2.2 Iterative array model of a sequential circuit.....11
3.1 A pseudorandom test pattern generator by a linear feedback shift register.....17
3.2 A combinational circuit example.....21
3.3 P(z=1) vs input probability x.....22
3.4 A timing diagram for fault and error latency.....29
4.1 The three states Markov model.....31
5.1 Logic representation of DM74LS164.....48
5.2 Testing time vs fault type for case 1.....51
5.3 Testing time vs fault type for case 2.....52
5.4 Testing time vs fault type for case 3.....53
5.5 Logic representation of DM74LS191.....55
6.1 Typical curve of P(0) vs $(B - \frac{A^2}{4})$.....58
6.2 P(0) vs different stuck-at-0 faults for case 3 of example 1.....61

LIST OF TABLES

5.1 Function table of DM74LS164 49
5.2 Results with last output monitored 51
5.3 Results with last three outputs monitored 52
5.4 Results with all outputs monitored 53
5.5 Results with Q_D monitored 56
5.6 Results with Q_C and Q_D monitored 57
5.7 Results with all flip-flop outputs monitored 57

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. **Sunil R. Das** for giving me the opportunity of working under his academic supervision. I also want to thank him for his patience and guidance during the preparation of this thesis.

Furthermore, I would like to thank all the professors and graduate students in the department of Electrical ineering of the University of Ottawa for providing the academic environment, the atmosphere and the help for the work of this thesis.

Finally, I want to thank my wife, **Yee-Man**, for her understanding and loving support during the preparation of this thesis. Just before the submission of this thesis, my first son, **Kin-Yung**, was born. His birth has given great joy to our family and also a big push for the final completion of this thesis.

ABSTRACT

A continuous parameter Markov model is proposed in this thesis to detect permanent stuck-at faults in sequential circuits by random testing. Given a sequential circuits with certain stuck-at faults specified, the fault-free and the faulty state tables of the circuit can be readily derived. By simulation of these two state tables on a computer, the parameters of the desired Markov model can be obtained. For a specified confidence level, it is easy to derive the model parameters and to estimate the required testing time. A complete mathematical analysis of the model is given that provides some useful insights into the nature of faults in relation to random testing and the associated confidence level.

1.0 INTRODUCTION

The increasing complexity of today's digital devices has rendered the problem of fault detection, fault analysis, and test generation not only an important and indispensable part of the manufacturing and maintenance process but at the same time extremely difficult [1-46]. The test generation of both combinational and sequential logic circuits can be broadly classified as either deterministic or probabilistic.

In deterministic approach, one needs exact knowledge of circuit behavior and test sequences are predetermined. For relatively complex digital systems, the associated computation time and costs can be prohibitive. On the other hand, random testing, which comes under the category of probabilistic test generation, can significantly reduce the effort involved.

Conventionally, in random testing, the effect of a failure in the logic network is propagated to the network outputs by applying random patterns to the primary inputs of the network. By using a simulator, the outputs of the fault-free and the faulty circuits are compared. If the outputs are different, the applied input pattern is retained as a test sequence. The experiment is continued until every fault in the circuit has been detected by at least one test pattern. The accuracy of the random testing depends on the length of the test patterns, and thus it is important to investigate the relationship between the random test length and the associated confidence level.

1.1 The contribution

In this thesis, the problem of detecting permanent stuck-at faults in sequential logic circuits by random testing is analyzed by a continuous parameter Markov model. The sequential circuits considered here are deterministic and synchronous, and can be represented by state diagrams with transition assigned outputs, or by equivalent state tables (Mealy model circuits). The circuits are also assumed to be strongly connected such that there exists a path, not necessarily of length 1, from any state to any other state.

Given a sequential circuit with certain stuck-at faults specified, the original state table and its faulty version can be readily derived from an analysis of the circuit under fault-free and faulty conditions, respectively. By simulation of these two tables on a computer, the parameters of the desired Markov model can be obtained. For a specified confidence level, it can estimate the length of test patterns required for fault detection.

A complete mathematical analysis of the model is given which provides some useful insights into the nature of faults in relation to random testing and the associated confidence level. The approach is valid for any sequential circuit with faults that result in a deterministic state table. Incidentally, all stuck-at faults in sequential circuits, and certain internal stuck-at faults in flip-flops which are commonly used as memory devices, have this property.

1.2 Scope

The material of this thesis is organized as follows :

Chapter 1 provides a brief introduction of the subject discussed in this thesis.

Chapters 2 and 3 give an overview of existing deterministic and probabilistic test generation approaches for digital devices, respectively.

A Markov model is then developed in Chapter 4 to describe the behavior of a faulty sequential logic circuit. A mathematical analysis is also given to calculate the testing time.

An algorithm for applying the Markov model is given in Chapter 5. It also contains two sequential logic circuits as examples to demonstrate the application of the approach.

Finally, discussions and conclusions are given in Chapter 6.

2.0 DETERMINISTIC TESTING

Deterministic testing techniques for combinational and sequential circuits have been described fully in [33,34]. The techniques are classified as deterministic since the faults are permanent, the test experiment is predetermined and all the I/O sequences are defined. In this chapter, some well known deterministic techniques are discussed briefly.

2.1 Test generation for combinational circuit

2.1.1 Fault Table method

The Fault Table method is probably the most classic approach to the solution of the problem. The method is to construct the truth tables under conditions of all single faults being studied. By comparing the faulty outputs with fault-free output, one can find the test set for the detection of different faults. Two faults are said to be indistinguishable if the circuit responses with these faults are identical. Sometimes one test set can detect several possible faults.

The draw back of the Fault Table method is that the size of the fault table will become unmanageable when the circuit is too big. A circuit with n inputs may have up to 2^n rows in the corresponding fault table.

2.1.2 Boolean Difference method

The Boolean Difference method is an algebraic method to analyze the faulty digital circuits. It uses the Boolean expressions to characterize the fault-free and the faulty circuit. After deriving two Boolean expressions, one for

fault-free and the other for the faulty circuit, they are then Exclusive-ORed. The result is 1 if a fault exists.

2.1.3 Path Sensitization (PS) method

From the Fault Table and Boolean Difference methods, we can find that some faults are equivalent or indistinguishable. Also, a test that detects a fault X can usually detect another fault Y.

The basic principle involved in path sensitization is to choose paths from the origin of the failure to the circuit output. The following steps are employed in the process of path sensitization :

a) Choose a path from the faulty line to one of the primary outputs.
b) Assign the faulty line a value 0 (1) if the fault is a stuck-at-1 (stuck-at-0) fault.
c) Along the chosen path, except for the lines of the path, assign a value 0 to each input to the OR and NOR gates in the path and a value 1 to each input to the AND and NAND gates in the path.
d) Trace back from the gates along the sensitized path toward the circuit inputs. If a consistent input combination (a test) exists, the procedure is terminated. If a contradiction is encountered, choose another path which starts at the faulty line, and repeat the above procedure.

The one dimensional path sensitization works for tree-type circuits, i.e. logic gates with fanout equal to one. For nontree circuits, multiple path sensitization may be necessary to find the test set.

2.1.4 Equivalent Normal Form (ENF) method

It is a systematic but heuristic procedure derived from the concept of path sensitization. The Equivalent Normal Form (ENF) of a combinational circuit is obtained by expressing the output of each gate as a sum-of-products expression of its inputs and preserving the identity of each gate by the subscript. Thus, each ENF corresponds to a hypothetical AND-OR equivalent circuit and the subscript captures all the gates that the signal path sensitized. Each subscripted input variable in an ENF is called a literal and an appearance of a literal in a term is also called a literal.

Thus, to test a particular literal for stuck-at faults, the literal must be assigned logical value opposite to the stuck-at value, i.e. assigning 1 for stuck-at-0 and 0 for stuck-at-1. The remaining literals in the term are assigned value 1 so that the stuck-at faults will be manifested at the AND gate. Also, at least one literal in each remaining term must be 0, in order that all remaining inputs to the OR gate be 0. Non-conflicting conditions for inputs are checked to determine the validity of the test patterns.

2.1.5 D-algorithm

It is the first algorithmic approach for test generation formulated in 1960s. It guarantees finding a test if there exists one. Several structures and concepts are involved and explained below.

Singular Cover : The compact version of the truth table. Xs or blanks are used to denote don't care conditions (i.e. the position could be 1 or 0).

Propagation D-cubes : D-cubes represent the input-output behavior of the fault-free and the faulty circuits. The symbol D is used to indicate the effect of the fault towards the circuit characteristics. The symbol D may assume only one value, either 0 or 1, throughout the circuit and D' is the inversion of D.

Primitive D-cube of a fault : It is used to specify the existence of a given fault. It consists of an input pattern which brings the influence of a fault to the output of the gate.

The primitive D-cubes of any fault in a gate can then be obtained from the singular covers of the faulty gates in the following manner :

1. From the singular covers of the fault-free and the faulty gates, let α_0 and α_1 be the sets of cubes in the singular covers of the fault-free gate whose output corresponds to 0 and 1, respectively. Also, let β_0 and β_1 be the corresponding sets in the singular covers of the faulty gate.
2. Intersect members of α_1 with members of β_0, and members of α_0 with members of β_1.

The primitive D-cubes of faults obtained from $\alpha_1 \cap \beta_0$ correspond to those inputs that produce a 1 output from the fault-free gate and a 0 output from the faulty gate. On the contrary, the primitive D-cubes of faults obtained from $\alpha_0 \cap \beta_1$ correspond to those inputs that produce a 0 output from the fault-free gate and a 1 output from the faulty gate.

D-intersection : It is a process of propagating a D-cube from input gate to the output. To transmit the D-cube from gate G_i to the next level gate G_j with a chosen signal path, one must try to match or intersect the D

specification with one of the propagation D-cubes for G_j.

With the abovementioned structures and concepts, the D-algorithm can be defined as follows :

1. Choose a primitive D-cube of the fault under consideration.
2. Sensitize all possible paths from the faulty gate to a primary output of the circuit. This is done by successive intersection of the primitive D-cubes of the fault with the propagation D-cubes of successor gates. The procedure is called the "D-drive".
3. Repeat "D-drive" until a primary output ends up with a D or D'.
4. Check nonconflicting conditions to determine the validity of test pattern.

2.2 Test generation for sequential circuit

The general model of a sequential circuit or a sequential machine can be represented by a structure consisting of a combinational logic circuit with the memory unit feeding back to inputs as shown in Fig. 2.1.

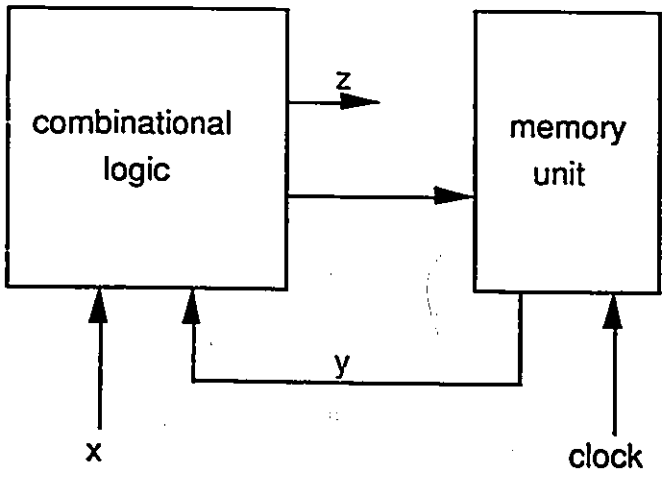

Fig 2.1: Sequential logic circuit representation

A sequential logic circuit can be classified as an asynchronous sequential circuit or a synchronous sequential circuit. For a synchronous sequential circuit, the operation is controlled or synchronized by an external clock and thus, the state or condition of the sequential circuit will only be changed at the specific time instant and become stable during the rest of the time. For an asynchronous sequential circuit, no external clock is usually required. Thus, the state will change whenever inputs are changed. Unless specified otherwise, a synchronous sequential circuit will be assumed in this thesis.

If the memory unit, as shown in Fig. 2.1, is composed of flip-flops, then the output of the sequential circuit, z(t), is determined by the present state of the flip-flops, y(t), and the system inputs, x(t). Thus, the output of a sequential circuit would have the general form

$$z(t) = F(x(t), y(t))$$

Due to the unique structure of a sequential circuit, we could have two distinct approaches to test generation.

1. *Iterative array method* : By converting a given synchronous sequential circuit into a one dimensional array of identical combinational circuits, and then cascading all together to give the same operation mode for the original circuit.
2. *State table verification method* : By verifying whether or not a given circuit is operating correctly in accordance with its state table.

2.2.1 Iterative array method

A general model of a sequential circuit is shown in Fig. 2.1. If there are n copies of the same sequential circuit with the feedback routes disconnected and redirected such that all copies are connected one after the other, i.e. the state of the first copy is connected to the second copy and thus becomes the system inputs of the second copy of the circuit, and, the state of the second copy is connected to the third, and so on, then the resulting structure will be as shown in Fig.2.2.

In Fig. 2.2, x(i), y(i) and z(i), associated with the ith copy of the circuit, correspond to the system inputs, the next state and the outputs of the sequential circuit at the ith instant of time.

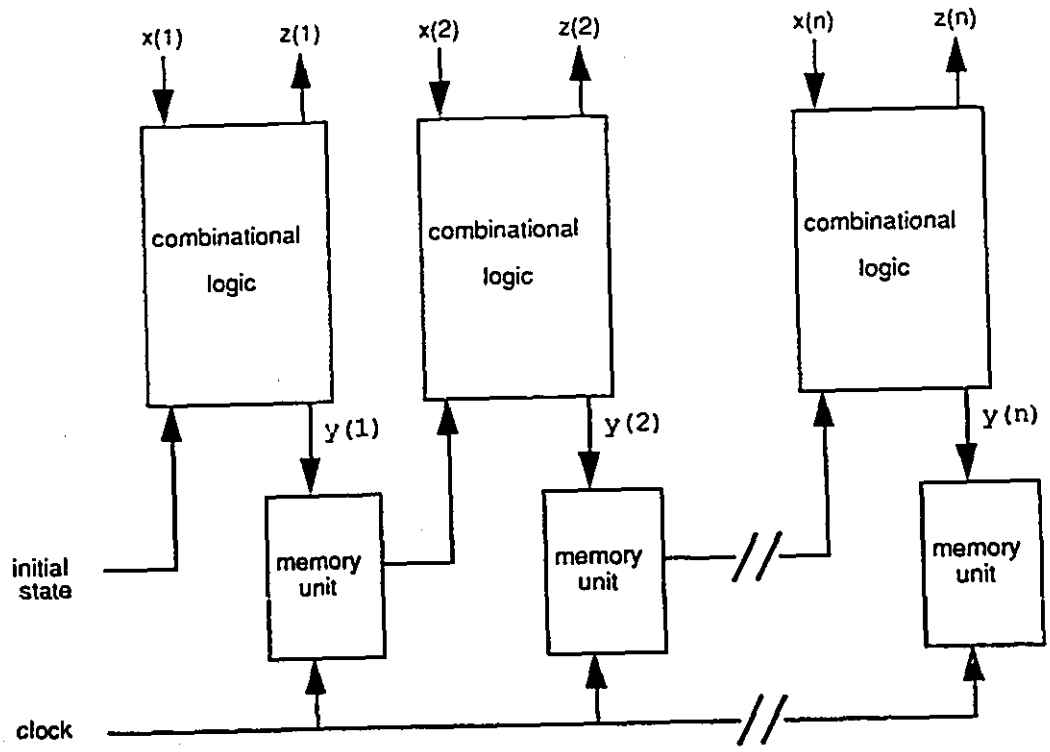

Fig 2.2: Iterative array model of a sequential circuit

The transformation process replaces the problem of finding a test sequence for a sequential circuit by the problem of finding tests for an iterative combinational circuit. Also, since each fault in the sequential circuit leads to n identical faults in the iterative network, the problem of fault detection in a sequential circuit is converted to the problem of multiple fault detection in a combinational network. In this case, most of the testing techniques for the combinational circuit such as path sensitization, Boolean difference method, D-algorithm, etc., can then be applied.

However, there are two problems by using this approach. They are :

1. The increased number of faults to be considered and the extra complexity of the replicated logic becomes unmanageable for complex circuits.
2. The length of the input sequence required, n, is usually different with different initial states. This requires derivation of test sequences for all possible initial states and then constructing a sequence which will detect the fault in all cases. This restricts the flexibility of the iterative model.

2.2.2 State table verification method

In this approach, a sequential circuit is studied analytically provided the state table (state transition and output table) is known. The different state identification sequences, viz homing sequence, transfer sequence, synchronizing sequence and distinguishing sequence, are studied closely in order to examine the performance of a sequential circuit. The abovementioned state identification sequences are defined below.

An input sequence is said to be a *homing* sequence if the final state of the sequential circuit can be determined uniquely from the circuit's response regardless of the initial state.

In the beginning of the test, the sequential circuit can be in any of its n states. It is "uncertainty" which present state the sequential circuit is in. Thus, the *uncertainty* of the sequential circuit is a set of possible present states of the sequential circuit.

After each input sequence, the sequential circuit would have different uncertainties. A collection of these uncertainties is referred to as an *uncertainty vector*. The individual uncertainties contained in this vector are called *components* of the uncertainty vector.

An uncertainty vector whose components contain a single state each is said to be a *trivial uncertainty vector*. An uncertainty vector whose components contain either single states or identical repeated states is said to be a *homogeneous uncertainty vector*.

With a given sequential circuit and initial uncertainty, a *successor tree* can be drawn to display graphically the x_i-successor uncertainties for every possible input sequence x_i.

The *homing* sequence is obtained from the homing tree, a successor tree in which a node becomes terminal if one of the following conditions occur :

1. The node is associated with an uncertainty vector whose nonhomogeneous components are associated with a node at a preceding level.
2. The node is associated with a trivial or a homogeneous uncertainty vector.

An input sequence is said to be a *distinguishing* sequence if the unique output sequence is produced in accordance to different initial states.

The *distinguishing* sequence is obtained from the distinguishing tree which is obtained the same way as the homing tree except that a node also becomes terminal if

3. The node is associated with an uncertainty vector containing a homogeneous nontrivial component.

An input sequence is said to be a *synchronizing* sequence if by applying the sequence, we take a sequential circuit to a specific final state regardless of the output or the initial state of the sequential circuit. It is described by a path in the synchronizing tree leading from the initial uncertainty to a singleton uncertainty, ie. the uncertainty that contains only one state.

The *synchronizing* sequence is obtained from the synchronizing tree, a successor tree in which a node becomes terminal if :

1. The node is associated with an uncertainty vector that is also associated with a node at a preceding level.
2. A node in the nth level is associated with an uncertainty containing just a single element.

A *transfer* sequence is the shortest input sequence that can bring the sequential circuit from one state to another state. It is evident that a *transfer* sequence will always exist in a strongly connected sequential circuit, i.e. we can reach each state of the given sequential circuit from every other states.

Now, we can design the checking experiment for a given sequential circuit. Basically, the checking experiment is a deterministic experiment used to verify the operation of the sequential circuit in accordance with the state table. If the circuit's response is different from the expected response, the sequential circuit is declared faulty but no specific fault can be identified from the experimental results. A sequential circuit is diagnosable, if and only

if, there is at least one distinguishing sequence available for the sequential circuit.

A checking experiment can be divided into three phases. They are :

1. Initialization phase : During the initialization phase, the sequential circuit under test is brought to a specified state from which the second phase will begin. A strongly connected sequential circuit can be brought to a fixed state by :

   a. applying a homing sequence to the sequential circuit, identifying the final state, and then applying a transfer sequence to bring it to a fixed state, if necessary;
   b. using the clear/preset capability of the sequential circuit;
   c. applying the synchronization sequence to the sequential circuit, if available.

2. State identification phase : During this phase, a distinguishing sequence is repeatedly applied to the sequential circuit under test to determine whether it has n different responses to the distinguishing sequence, indicating n distinct states.

3. Transition verification phase : During this phase, the sequential circuit under test is made to go through all possible state transitions by using the distinguishing sequence.

In practice, the state identification and transition verification are combined whenever possible to shorten the length of the experiment. Therefore, the checking experiment is not necessarily unique.

3.0 RANDOM TESTING

The deterministic test generation approaches discussed in Chapter 2 are usually unable to cope with complex circuits such as those at the LSI or VLSI level. For sequential logic circuits with m flop-flops, a total of 2^m states may be present. The computation time, required by using a deterministic test generation approach that detects all single or multiple stuck-at faults, would be prohibitively expensive for large circuits. The random testing could be a more effective way to deal with complex digital circuits of today.

In this chapter, random testing methods are discussed. There are basically two types of random testing procedures. The first procedure is called the random test generation method. The test patterns are generated randomly and then applied to the primary inputs of the digital circuit under test. The outputs of the faulty and the fault-free networks for each random input pattern are compared by using a simulator. If the output values are different, the random input pattern is retained as a test.

The second type of random testing procedure is performed by comparing the output of the circuit under test with the output of a known fault-free unit or "gold" unit, while both units are fed by the same pseudorandom sequences. The two outputs are EX-ORed and if the result is 1, a fault is detected.

The pseudorandom input patterns can be generated by a linear feedback shift register (LFSR) as described in [32]. Consider Fig. 3.1 as an example.

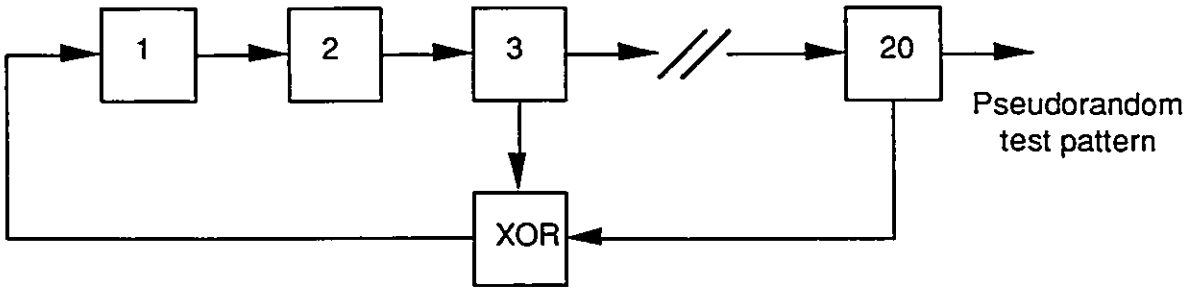

Fig 3.1: A pseudorandom test pattern generator by a linear feedback shift register

Such registers generate long, repetitive pseudorandom sequences of 0s and 1s, short segments of which would have properties similar to random patterns.

3.1 Test generation for combinational circuit

3.1.1 NAND tree network

In [5,10,11], the NAND fan-out-free tree network was studied. All NAND gates are with n inputs. Let p_k^0 be the probability of a logical 0 occurring on a line in the kth level of the network, and all the n inputs of the gate feeding this line are from the (k-1)th level. The number n can be treated as the number of fan-ins at any level k. Then,

$$p_k^0 = (p_{k-1}^1)^n = (1 - p_{k-1}^0)^n, \text{ and} \qquad (3.1)$$

$$p_k^1 = 1 - p_k^0 = 1 - (p_{k-1}^1)^n \qquad (3.2)$$

If we assume $p_0^1 = q$, then from Equation 3.2, we have

$p_1^1 = 1 - (p_0^1)^n = 1 - q^n;$

$p_2^1 = 1 - (p_1^1)^n = 1 - (1 - q^n)^n;$ and so on.

Example 3.1

For a two-input NAND tree, ie. n = 2, assume q = 0.5, then we have

$p_1^1 = 1 - (0.5)^2 = 0.75;$

$p_2^1 = 1 - (0.75)^2 = 0.4375;$ and so on.

Let P(L) be the detection probability, the probability of sensitizing a path of length L. By sensitizing a path, we mean propagating the effect of the fault from the point of its occurrence to a primary output through L levels of logic. For a NAND tree network, to sensitize a path from one level to the next level, all inputs to the NAND gates should be 1 except the input that is under study. Then P(L) can be equated as follows :

$$P(L) = P(0) * \prod_{k=0}^{L-1} (p_k^1)^{n-1} \qquad (3.3)$$

where P(0) is the probability of a fault existed at the point of its occurrence; and
$(p_k^1)^{n-1}$ is the probability of having (n-1) 1's in level k; and
for the input faults, P(0) = 1.

Example 3.2

For a NAND tree, assume $P(0) = 1$, the number of fan-ins $(n) = 2$, and the p_k^1s are taken from Example 3.1; then from Equation 3.3, we have

$$P(1) = P(0) * p_0^1 = 0.5;$$

$$P(2) = P(0) * \prod_{k=0}^{1} p_k^1 = 0.5 * 0.75 = 0.375;$$

$$P(3) = P(0) * \prod_{k=0}^{2} p_k^1 = 0.5 * 0.75 * 0.4375 = 0.164;$$

and so on.

Let q_{opt} be the optimal input probability that detection probability can be maximized. For large L, q_{opt} can be obtained from the equation

$$q_{opt} = 1 - q_{opt}^n \qquad (3.4)$$

By substituting this relation into Equations 3.2 and 3.3, we have

$p_0^1 = p_1^1 = p_2^1 = \ldots = p_{L-1}^1 = q_{opt}$, and

$P(L) = P(0) * q_{opt}^L$

For n = 2, we can solve q_{opt} by numerical method; then we have

$q_{opt} \cong 0.618$

Let P(L,M) be the probability of sensitizing a path of L levels by at least one out of M independent patterns; it can be considered as a measure of confidence level to detect the fault. Then

$$P(L,M) = 1 - \{1 - P(L)\}^M$$

Taking natural logarithm, we can derive

$$M = \ln\{1 - P(L,M)\} / \ln\{1 - P(L)\} \qquad (3.5)$$

It can be noted that for $q = q_{opt}$, the test pattern length, M, is the minimum for a given P(L,M).

Example 3.3

For P(L,M) = 99%, n = 2 and L = 3, we can calculate the test pattern length by Eq. 3.5 as

$$M(99\%, q = 0.5) \cong \ln(0.01) / \ln(1 - 0.164) \cong 26$$

$$M(99\%, q = 0.618) \cong \ln(0.01) / \ln(1 - 0.618^3) \cong 18$$

It was pointed out in [12] that maximum rather than average fanins, n, may be a more conservative measure which avoids the incorrect conclusion when the auxiliary outputs are not on the longest path of the circuit.

Thus M(99%) should be treated as the order of the number of random patterns rather than an upper bound. Since the estimate is statistical, its accuracy is better for a larger circuit [11]. The maximum fanin is more conservative in general but may be too pessimistic for a larger circuit.

3.1.2 Propagation probability

In [6,7], general combinational networks are analyzed. The different probabilities are assigned to the input signals to derive propagation probability. The test lengths are then determined based on propagation probability. Consider Fig. 3.2 as an example.

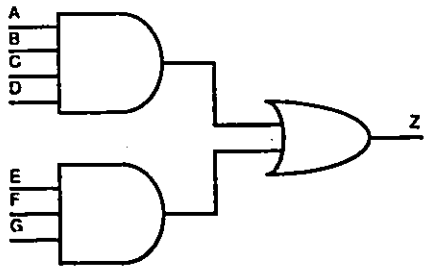

Fig 3.2: A combinational circuit example

In order to detect the failure of z stuck-at 0, we need to provide a z = 1 output. Thus the probability of detecting z stuck-at 0 is simply P(z = 1) when we apply a probability distribution of 1's and 0's to the input.

If all inputs (A to G) have the same probability x, of being 1, then we have

$$P(z = 1) = x^3 + x^4 - x^7$$

which is illustrated in Fig. 3.3.

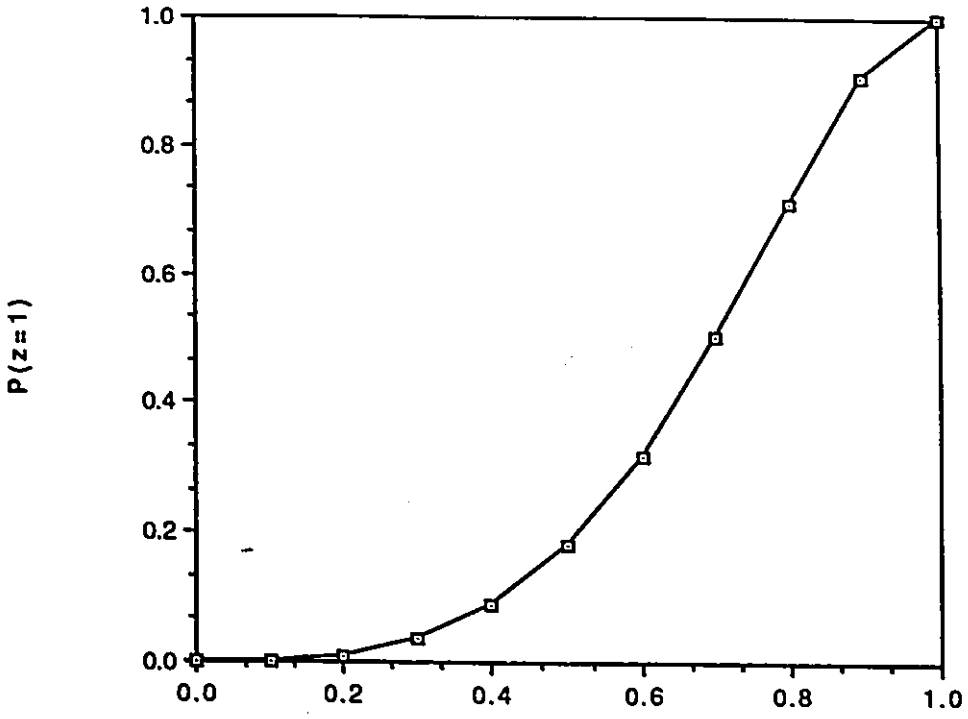

Fig 3.3: P(z=1) vs input probability x

It can be seen that the probability of detecting a fault is highly dependent upon the value of x. If we assume that the primary inputs have the same probability of 0's and 1's, i.e. x = 0.5, it yields P(Z = 1) = 0.1797. Thus if random patterns are applied to the inputs, we have a low probability of detecting the fault.

In fact, for combinational circuits, if the fault is detectable, the longer a circuit under test successfully executes a chain of tests, the greater is the probability that the circuit is fault-free. However, for an n input combinational circuit, only if the number of random test pattern is less than 2^n, then the random testing is recommended. The required number of random patterns could be

found by logic analysis. But for large circuits, the analysis would be prohibitive.

3.1.3 Detection surface

In [4], another method was proposed to find the upper bound for the test length. The method is developed by using the concept of *detection quality* and *detection surface*.

Let $(1-Q_D)$ be the *detection quality*, i.e. the probability for a result to be *right*. The result of a test is *right* if its value is "faulty" when the circuit is faulty. Q_D is then called *detection uncertainty*.

Also, let σ be the *detection surface* of a network, i.e. the number of Karnaugh map surfaces that are being covered by input vectors for the most difficult fault to detect.

Assume that one input vector has the probability $1-\sigma/2^n$ of not detecting the most difficult fault, where n is the number of primary inputs of the network. Then, for a random input sequence of length L, the detection uncertainty would be

$$Q_D = (1 - \sigma/2^n)^L$$

To take logarithm (base 10), we have

$$\sigma = 2^n * (1 - 10^{\log(Q_D)/L})$$

Let $Q_D = 10^{-d}$ and L = test length = 10^k. If $d << L$, then

$$1 - 10^{\log(Q_D)/L} \cong 2.3*d / 10^k$$

Thus, for $d << L$, we have

$$\log(\sigma) = n*\log(2) - k + \log(2.3*d) \qquad (3.6)$$

From Equation 3.6, it can be implied that, for a given Q_D, the test length L is a function of n and σ. Now, two approaches are given below to determine the detection surface for a combinational circuit.

1) For an irredundant function, it can be expressed in a sum-of-products form, i.e. AND-OR network. Then, the detection surface would have the following lower bound.

$$\sigma \geq \min_q \{2^{n-n_q-1}\} \qquad (3.7)$$

where n is the total number of primary inputs; and
n_q is the number of input variables appearing in the qth product term of the function.

2) For a multiple output combinational circuit, the detection surface would have the following lower bound.

$$\sigma \geq \min_k \{2^{n-n_k-1}\} \qquad (3.8)$$

where n is the total number of primary inputs; and
n_k is the number of input variables related to the kth output of the circuit.

Then, to find the overall detection surface σ of a combinational circuit, a multiple output circuit is first converted to k irredundant circuits. The minimum detection surfaces of those irredundant circuits are then determined. Finally, the minimum detection surface of these minimum σs is chosen as the detection surface of the circuit. Two remarks are given in [4].

1. It is reasonable to use the random testing for large circuits if they have a large σ. The test length L is a function of σ / 2^n.
2. If $\sigma > \ln(1/Q_D)$, then $L < 2^n$, i.e. random testing can be used.

3.1.4 Detection probability

In [15], an estimation procedure for detection probabilities was proposed. Using the procedure, a list of estimated detection probabilities for all faults in a combinational circuit can be determined. With these detection probabilities, the random pattern test length for multiple faults is developed in [17].

Consider a combinational circuit with faults f_1 and f_2. Let p_1 be the probability that a random pattern will detect f_1 but not f_2; p_2 be the probability that the pattern will detect f_2 but not f_1; and p_3 be the overlap probability that detects both faults.

Let e_n be the nth step escape probability of a fault set, the probability that at least one member of the fault set will not be detected by the nth random input vector. In order to compute the minimum length of the random test pattern N that detects the fault set with escape probability, the escape probability threshold e_t can be set arbitrarily to meet the following inequality :

$$e_n \leq e_t.$$

If the worst fault is the fault having the lowest detection probability of any fault in the circuit, then the test length to detect the worst fault N_{wf} is defined as follows.

$$N_{wf} = \left[\frac{\ln(e_t)}{\ln(1-p)} \right]_{ceil}$$

where $[x]_{ceil}$ denotes the ceiling of x, i.e. next integer above x; and
$p = \min (p_1+p_3, p_2+p_3)$

It has been found that the test length for the disjoint case ($p_3 = 0$) is greater than or equal to the conjoint test length ($p_3 > 0$). Thus, the upper bound to the minimum test length can be computed by assuming that all faults have disjoint test sets.

If we assume that a circuit contains k hard faults, ie. the faults having estimated detection probabilities less than or equal to twice that of the worse fault, then $N^{U}_{k_D}$, the upper bound of the minimum test length to detect all faults, is approximated as

$$N^{U}_{k_D} = \left[\frac{\ln(e_t/k)}{\ln(1-p)} \right]_{ceil} \qquad (3.9)$$

where p is the detection probability of the worst fault.

Applying $N^{U}_{k_D}$ random test patterns to a circuit guarantees the detection of all faults with a confidence of at least $1 - e_t$.

From Eq. 3.9, it can be deduced that a test length of 11/p can detect as many as 50 hard faults, each having a detection probability p, with a confidence level of 99.9%.

3.2 Test generation for sequential circuit

3.2.1 Weighted random test generation (WRTG)

In [9], the weighted random test generation method is discussed. An LSFR is used to produce 8 decoder inputs which are connected to an 8 to 256 lines decoder. The outputs of the decoder are then grouped to give different weights for each group that are connected to the primary inputs of the digital system. Also, dynamic adaptive weighted patterns can be generated which are assigned weights according to the rate of change of activity.

To further speed up the test generation procedure, a pattern reduction technique is also proposed. The ability of each new input pattern to detect faults in a fault list is determined. The successful tests are stored while the faults they detect are deleted from the fault list. The procedure is halted when all possible faults are detected.

The WRTG method has been applied to a variety of LSI development techniques. It is reported that the WRTG method can achieve higher testability coverage than the deterministic techniques even in the presence of embedded counters and shift registers type configuration. However, it requires higher number of test patterns and longer CPU time,

3.2.2 Graph reduction

In [19], an approximation method is developed to determine the test length, L, required to detect a given fault in a sequential logic circuit.

Let c be the confidence level of detecting a fault and PF_{lfs} be the probability of being in the least frequent state; then, the test length, L, is given as follows :

$$L > \log(1-c) \; / \; \log(PF_{lfs}) \tag{3.10}$$

In order to find PF_{lfs}, a graph reduction algorithm is developed to calculate the convergent probabilities of each internal state. An original graph is first reduced to several subgraphs and the propagation probabilities are then calculated. The minimum propagation probability, P_{min}, is retained and yields the PF_{lfs} as

$$PF_{lfs} = 1 - P_{min}$$

Then Eq. 3.10 becomes

$$L = \left[\frac{\log(1-c)}{\log(1-P_{min})} \right]_{ceil} \tag{3.11}$$

3.2.3 Error latency

It was pointed out in [2] that the abovementioned approximation method is insufficient when the fault and error latency exist. The error latency model is then developed to cover this aspect. In [25], the fault latency is defined as the time interval between the occurrence of a fault and the generation of an error by that fault; the error latency represents the interval between the generation of an error and the detection of that error by some detection mechanism. The concept of fault and error latency is illustrated in Fig. 3.4.

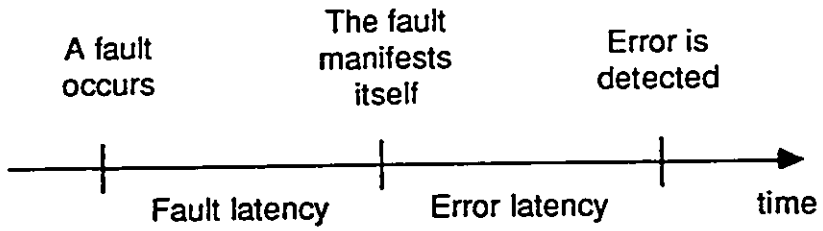

Fig 3.4: A timing diagram for fault and error latency

The error latency EL_k of a fault F_k is the number of input vectors applied to a digital circuit while F_k is active, until the first incorrect output vector due to F_k is observed.

When random inputs are applied to a deterministic sequential circuit, the probability of being in a given state at time n depends not only on the state of the circuit at time n-1 but on the probability of the input vectors at time n-1. The operation of the circuit may be described by a finite first-order Markov chain. Thus, we have

$$S_n = P^n * S_0 \qquad (3.12)$$

where S_0 is the initial state probability vector, and
P^n is the n step transition probability matrix.

For a given Error Latency Model (ELM), there is always an absorbing state, S_+, indicating the detection of the error. By Eq. 3.12, we can calculate the latency interval that the probability of being in S_+ is equal to the confidence level c. The latency interval , denoted by n(c), is defined as the minimum number of applied inputs necessary to achieve a confidence level c of observing an error due to

a given fault. Since the computation can be exhaustive for large circuits, the latency interval is approximated as

$$n(c) = h * (\log(1-c) / \log(1-q)) \qquad (3.13)$$

where h is the maximum length of the shortest path from each state to the absorbing state S_+, and
q is the minimum probability of the shortest path from each state to S_+ and c is the confidence level.

It was later pointed out in [21] that the approximation equation in [19] can be rewritten to yield comparable results as given in ELM. If a given fault affects transition(s) among the less probable states and the detection is not sure after a fault transition, i.e. $V_j * P_i << 1$, then the approximation equation can be rewritten as

$$n(c) = \log(1-c) / (P_i * \log(1-V_j)) \qquad (3.14)$$

where V_j is the probability of a fault affecting a transition j and P_i is the detection probability after the transition j is achieved.

Since P_i is fault dependent and the possible faults for a large circuit are numerous, both the approximation method and ELM are computationally prohibitive.

It can be concluded that there is practically no systematic and efficient method of dealing with sequential logic circuits using random or probabilistic testing.

4.0 CONTINUOUS MARKOV MODEL FOR RANDOM TESTING OF SEQUENTIAL CIRCUITS

In order to describe the behavior of a sequential circuit in the presence of a fault while subjected to random inputs, we propose a continuous parameter Markov model with three states designated as : state 0, state 1, state 2, as shown in Fig. 4.1.

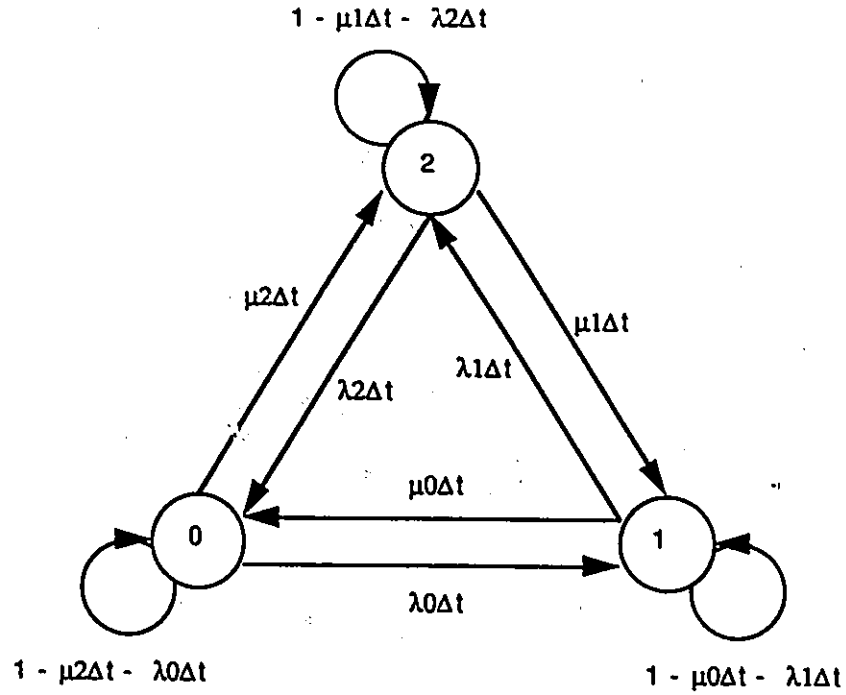

Fig 4.1: The three states Markov model

The state transition probabilities depend linearly on the infinitesimal time step Δt; λ_is and μ_js are the constants of probabilities. Clearly, these probabilities should increase as the time step Δt increases. Once a random pattern is applied, the circuit may stay in any one of the above three states. The different states are defined as :

state 0 : the fault exists, but causes no error output and error state transition;

state 1 : the fault causes an error output in the output terminals;

state 2 : the fault causes only error state transition, but no error output results in the output terminals.

To represent the interaction among the three state 0, 1, and 2 for the infinitesimal time step Δt in the circuit, we can define now the different state transition probabilities.

4.1 Model definitions

Let $P((S=x, T=t+\Delta t) | (S=y, T=t))$ denote the probability of the circuit of staying in state y at time t, but going to state x at time $t+\Delta t$. Then we have:

$\lambda_0 \Delta t$: $P((S=1, T=t+\Delta t) | (S=0, T=t)) = P((\text{fault causes an error output at time } t+\Delta t) \mid (\text{fault causes no error output and error state transition at time } t))$;

$\lambda_1 \Delta t$: $P((S=2, T=t+\Delta t) | (S=1, T=t)) = P((\text{fault causes only error state transition at time } t+\Delta t) \mid (\text{fault causes an error output at time } t))$;

$\lambda_2 \Delta t$: $P((S=0, T=t+\Delta t) | (S=2, T=t)) = P((\text{fault causes no error output and error state transition at time } t+\Delta t) \mid (\text{fault causes only error state transition at time } t))$;

$\mu_0 \Delta t$: $P((S=0, T=t+\Delta t) | (S=1, T=t)) = P((\text{fault causes no error output and error state transition at time } t+\Delta t) \mid (\text{fault causes an error output at time } t))$;

$\mu_1\Delta t$: P((S=1, T=t+Δt)|(S=2, T=t)) = P((fault causes an error output at time t+Δt) | (fault causes only error state transition at time t));

$\mu_2\Delta t$: P((S=2, T=t+Δt)|(S=0, T=t)) = P((fault causes only error state transition at time t+Δt) | (fault causes no error output and error state transition at time t));

$1-\lambda_0\Delta t-\mu_2\Delta t$: P((S=0, T=t+Δt)|(S=0, T=t)) = P((fault causes no error output and error state transition at time t+Δt) | (fault causes no error output and error state transition at time t));

$1-\lambda_1\Delta t-\mu_0\Delta t$: P((S=1, T=t+Δt)|(S=1, T=t)) = P((fault causes an error output at time t+Δt) | (fault causes an error output at time t));

$1-\lambda_2\Delta t-\mu_1\Delta t$: P((S=2, T=t+Δt)|(S=2, T=t)) = P((fault causes only error state transition at time t+Δt) | (fault causes only error state transition at time t));

Obviously, the parameters of the model depend on the circuit, and faults under consideration. Once we have the fault-free state table of the circuit and its faulty version corresponding to the faults specified, we may proceed to find all the parameters of the model. A rather simple way is to apply a large number of input patterns to the circuit under test with the occurrence of input signal values 0,1 in a constant probability. Under this condition, the behavior of the circuit may be characterized by a *Markov* process, and the probability of the circuit staying in a specified state assumes a constant value after long input patterns are applied. This property makes our estimation of parameters approach stable values.

In order to simulate the transition behavior of the circuit with respect to the model, we note that we have nine different counts provided to store the information regarding the circuit after a large number of the input patterns are applied. Let C_{ab} = N(circuit going from state a to state b);

where a = 0,1,2; b = 0,1,2;
N denotes the total occurrence number or count of the event under reference.

Specifically, for example, with a = 2, b = 1, we have

C_{21} = N(circuit going from state 2 to state 1) = N(fault causing only error state transition in input pattern r, while causing an error output in input pattern r+1).

All counts are cleared to zero initially. By using a random number generator we can generate an input value on the interval [0,1], and then apply it to the circuit with a certain constant probability. However, instead of physical inputs to the circuit, we may also simulate the stimulation with respect to the fault-free state table of the circuit and its corresponding faulty table. On comparing the status of these two state tables, next on each application of a random pattern, we will have any of the following three different cases in the compared result.

Case 1 : Neither output nor state transition is different, and the circuit is in state 0.
Case 2 : Output is different, and the circuit is in state 1.
Case 3 : Only state transition is different, but output is not different, and the circuit is in state 2.

The circuit is evidently in state 0 initially, since at the beginning of test no differences ever occurred. After the first random pattern is applied to the circuit, we compare the outcomes with respect to the two different tables, and then decide upon which one of the aforementioned three cases occurs; the state of the circuit in respect of the current input pattern is thus determined.

Assume that on application of the first input pattern we have determined the circuit to be in state i, $0 \le i \le 2$; then $C_{0i} = C_{0i} + 1$. Next we apply the second random pattern to the circuit, and determine which state it is in. If it is now in state j, $0 \le j \le 2$; then $C_{ij} = C_{ij} + 1$. This process is repeated until a large number of input patterns are applied to the circuit. Suppose a total of n input patterns are applied. Hence we have :

$$n = \sum_{i=0}^{2} \sum_{j=0}^{2} C_{ij}$$

For large n, the model parameters can be estimated as follows :

$\mu_0 = C_{10}/t$, $\mu_1 = C_{21}/t$, $\mu_2 = C_{02}/t$; and
$\lambda_0 = C_{01}/t$, $\lambda_1 = C_{12}/t$, $\lambda_2 = C_{20}/t$;

where t is the total simulation time.

4.2 Solving for model parameters

Once the model is established, and all the parameters of the model are calculated by computer simulation, we can write the undernoted set of differential equations involving the

state probabilities, on the assumption that at time T = 0, the fault causes no error output and error state transition in the circuit. The set of differential equations are:

$$\frac{\partial P_0(t)}{\partial t} = P_1(t) * \mu_c + P_2(t) * \lambda_2 - P_0(t) * (\mu_2 + \lambda_0) \qquad (4.1)$$

$$\frac{\partial P_1(t)}{\partial t} = P_0(t) * \lambda_c + P_2(t) * \mu_1 - P_1(t) * (\mu_0 + \lambda_1) \qquad (4.2)$$

$$\frac{\partial P_2(t)}{\partial t} = P_0(t) * \mu_2 + P_1(t) * \lambda_1 - P_2(t) * (\lambda_2 + \mu_1) \qquad (4.3)$$

with $P_0(0) = 1$, $P_1(0) = P_2(0) = 0$ and $P_i(t)$ being the probability at time t for the circuit of staying in state i, $0 \leq i \leq 2$.

After solving this homogeneous system of differential equations (refer to Appendix B1), we have:

$$P_0(t) = \left(\frac{D_0}{B}\right) * u(t) + e^{-(A/2)*t} * \left(\frac{B - D_0}{B} * \cos\left(\sqrt{B - \frac{A^2}{4}} * t\right) + \frac{2*B*C_0 - A*B - A*D_0}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B - \frac{A^2}{4}} * t\right) \right) \qquad (4.4)$$

$$P_1(t) = \left(\frac{D_1}{B}\right) * u(t) + e^{-(A/2)*t} * \left(\frac{-D_1}{B} * \cos\left(\sqrt{B - \frac{A^2}{4}} * t\right) + \frac{2*B*C_1 - A*D_1}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B - \frac{A^2}{4}} * t\right) \right) \qquad (4.5)$$

$$P_2(t) = \left(\frac{D_2}{B}\right) * u(t) + e^{-(A/2)*t} * \left(\frac{-D_2}{B} * \cos\left(\sqrt{B - \frac{A^2}{4}} * t\right) + \frac{2*B*C_2 - A*D_2}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B - \frac{A^2}{4}} * t\right) \right) \qquad (4.6)$$

where $A = \mu_0 + \mu_1 + \mu_2 + \lambda_0 + \lambda_1 + \lambda_2$,

$B = \mu_0*\mu_1 + \mu_1*\mu_2 + \mu_0*\mu_2 + \lambda_0*\lambda_1 + \lambda_1*\lambda_2 + \lambda_0*\lambda_2 + \lambda_0*\mu_1 + \lambda_2*\mu_0 + \mu_2*\lambda_1$,

$C_0 = \mu_0 + \mu_1 + \lambda_1 + \lambda_2$,

$D_0 = \mu_0*\lambda_2 + \mu_0*\mu_1 + \lambda_1*\lambda_2$,

$C_1 = \lambda_0$,

$D_1 = \lambda_0*\lambda_2 + \lambda_0*\mu_1 + \mu_1*\mu_2$,

$C_2 = \mu_2$,

$D_2 = \mu_0*\mu_2 + \mu_2*\lambda_1 + \lambda_0*\lambda_1$,

and,

$$u(t) \begin{cases} = 0 & \text{for } t < 0 \\ = 1 & \text{for } t \geq 0 \end{cases}$$

Obviously, we have

$$\lim_{t\to\infty} P_0(t) + \lim_{t\to\infty} P_1(t) + \lim_{t\to\infty} P_2(t) = \frac{D_0 + D_1 + D_2}{B} = 1$$

With respect to the term $(B - \frac{A^2}{4})$ in the above solutions, we may have three different cases.

Case 1: $(B - \frac{A^2}{4}) = 0$

In this case, Equations 4.4 to 4.6 become

$$P_0(t) = (\frac{D_0}{B})*u(t) + e^{-(A/2)*t} * (\frac{B-D_0}{B} + \frac{2*B*C_0 - A*B - A*D_0}{2*B} * t)$$

$$P_1(t) = (\frac{D_1}{B})*u(t) + e^{-(A/2)*t} * (\frac{-D_1}{B} + \frac{2*B*C_1 - A*D_1}{2*B} * t)$$

$$P_2(t) = (\frac{D_2}{B})*u(t) + e^{-(A/2)*t} * (\frac{-D_2}{B} + \frac{2*B*C_2 - A*D_2}{2*B} * t)$$

Thus, for $(B - \frac{A^2}{4}) = 0$ or $A = 2 * \sqrt{B}$, the probabilities of the three states converge to their steady state values in a non-oscillatory manner.

Case 2: $(B - \frac{A^2}{4}) > 0$

In this case, damping exists and the probabilities of three states decay exponentially in an oscillatory manner.

Case 3: $(B - \frac{A^2}{4}) < 0$

In this case, Equations 4.4 to 4.6 can be rewritten as

$$P_0(t) = (\frac{D_0}{B}) * u(t) + K_1 * e^{-S_1 * t} + K_2 * e^{-S_2 * t}$$

$$P_1(t) = (\frac{D_1}{B}) * u(t) + K_1' * e^{-S_1 * t} + K_2' * e^{-S_2 * t}$$

$$P_2(t) = (\frac{D_2}{B}) * u(t) + K_1'' * e^{-S_1 * t} + K_2'' * e^{-S_2 * t}$$

where $S_1 = \frac{A}{2} - \sqrt{\frac{A^2}{4} - B}$, $S_2 = \frac{A}{2} + \sqrt{\frac{A^2}{4} - B}$,

$$K_1 = \frac{C_0 - S_1 - \frac{D_0}{S_1}}{2 * \sqrt{\frac{A^2}{4} - B}}, \quad K_2 = \frac{C_0 - S_2 - \frac{D_0}{S_2}}{-2 * \sqrt{\frac{A^2}{4} - B}},$$

$$K_1' = \frac{C_1 - \frac{D_1}{S_1}}{2 * \sqrt{\frac{A^2}{4} - B}}, \quad K_2' = \frac{C_1 - \frac{D_1}{S_2}}{-2 * \sqrt{\frac{A^2}{4} - B}},$$

$$K_1'' = \frac{C_2 - \frac{D_2}{S_1}}{2 * \sqrt{\frac{A^2}{4} - B}}, \quad K_2'' = \frac{C_2 - \frac{D_2}{S_2}}{-2 * \sqrt{\frac{A^2}{4} - B}}.$$

Here the probabilities of the three states converge to their steady state values in the same nonoscillatory manner as in case 1, but at a much slower rate.

From the above analysis, we can see that in order for a fault in a circuit to be detected quickly with a constant probability, it is necessary that the parameters or rates of the model should satisfy the condition specified by $(B - \frac{A^2}{4}) \geq 0$.

In case this condition is not met, we have to modify our input signal probability assignment for the circuit to change the parameters of the model in a way that will increase the convergence speed of the state probabilities so that the fault may be detected as fast as possible.

To further analyze our model, we may develop transition probabilities among the states. Let $P_{ij}(t)$ denotes the probability of going from state i at time t_0 to state j at time t_0+t. We can hence write the following sets of differential equations in terms of the state transition probabilities.

$$\frac{\partial P_{00}(t)}{\partial t} = P_{01}(t) * \mu_0 + P_{02}(t) * \lambda_2 - P_{00}(t) * (\mu_2+\lambda_0) \qquad (4.7)$$

$$\frac{\partial P_{01}(t)}{\partial t} = P_{00}(t) * \lambda_0 + P_{02}(t) * \mu_1 - P_{01}(t) * (\mu_0+\lambda_1) \qquad (4.8)$$

$$\frac{\partial P_{02}(t)}{\partial t} = P_{00}(t) * \mu_2 + P_{01}(t) * \lambda_1 - P_{02}(t) * (\lambda_2+\mu_1) \qquad (4.9)$$

with $P_{00}(0) = 1$, $P_{01}(0) = P_{02}(0) = 0$ and

$$P_{00}(t) + P_{01}(t) + P_{02}(t) = 1.$$

$$\frac{\partial P_{10}(t)}{\partial t} = P_{11}(t) * \mu_0 + P_{12}(t) * \lambda_2 - P_{10}(t) * (\mu_2+\lambda_0) \qquad (4.10)$$

$$\frac{\partial P_{11}(t)}{\partial t} = P_{10}(t) * \lambda_0 + P_{12}(t) * \mu_1 - P_{11}(t) * (\mu_0 + \lambda_1) \qquad (4.11)$$

$$\frac{\partial P_{12}(t)}{\partial t} = P_{10}(t) * \mu_2 + P_{11}(t) * \lambda_1 - P_{12}(t) * (\lambda_2 + \mu_1) \qquad (4.12)$$

with $P_{11}(0) = 1$, $P_{10}(0) = P_{12}(0) = 0$ and
$P_{10}(t) + P_{11}(t) + P_{12}(t) = 1$.

$$\frac{\partial P_{20}(t)}{\partial t} = P_{21}(t) * \mu_0 + P_{22}(t) * \lambda_2 - P_{20}(t) * (\mu_2 + \lambda_0) \qquad (4.13)$$

$$\frac{\partial P_{21}(t)}{\partial t} = P_{20}(t) * \lambda_0 + P_{22}(t) * \mu_1 - P_{21}(t) * (\mu_0 + \lambda_1) \qquad (4.14)$$

$$\frac{\partial P_{22}(t)}{\partial t} = P_{20}(t) * \mu_2 + P_{21}(t) * \lambda_1 - P_{22}(t) * (\lambda_2 + \mu_1) \qquad (4.15)$$

with $P_{22}(0) = 1$, $P_{20}(0) = P_{21}(0) = 0$ and
$P_{20}(t) + P_{21}(t) + P_{22}(t) = 1$.

The above sets of equations are isomorphic with Equations 4.1 to 4.3 defined above. These systems of differential equations can be solved as before. The solutions corresponding to the different sets are given as follows, with

$P_{00}(t) = P_0(t)$, $P_{01}(t) = P_1(t)$, and $P_{02}(t) = P_2(t)$, with all parameters being the same as those in $P_0(t)$, $P_1(t)$, and $P_2(t)$.

$P_{10}(t) = P_2(t)$, with $C_2' = \mu_0$,
$D_2' = \lambda_1*\lambda_2 + \lambda_2*\mu_0 + \mu_0*\mu_1$;
$P_{11}(t) = P_0(t)$, with $C_0' = \lambda_0 + \lambda_2 + \mu_1 + \mu_2$,
$D_0' = \lambda_0*\lambda_2 + \lambda_0*\mu_1 + \mu_1*\mu_2$;
$P_{12}(t) = P_1(t)$, with $C_1' = \lambda_1$,
$D_1' = \lambda_0*\lambda_1 + \lambda_1*\mu_2 + \mu_0*\mu_2$.

$$P_{20}(t) = P_1(t), \text{ with } C_1'' = \lambda_2,$$
$$D_1'' = \lambda_1*\lambda_2 + \lambda_2*\mu_0 + \mu_0*\mu_1;$$
$$P_{21}(t) = P_2(t), \text{ with } C_2'' = \mu_1,$$
$$D_2'' = \lambda_0*\lambda_2 + \lambda_0*\mu_1 + \mu_1*\mu_2;$$
$$P_{22}(t) = P_0(t), \text{ with } C_0'' = \lambda_0 + \lambda_1 + \mu_0 + \mu_2,$$
$$D_0'' = \lambda_0*\lambda_1 + \lambda_1*\mu_2 + \mu_0*\mu_2.$$

As in the case of state probabilities, the condition for rapid convergence to steady state values for the state transition probabilities depends on $(B - \frac{A^2}{4})$. If $(B - \frac{A^2}{4}) \geq 0$, all of the state transition probabilities converge quickly to their steady values. The result about $P_{00}(t) = P_0(t)$, $P_{01}(t) = P_1(t)$, and $P_{02}(t) = P_2(t)$ appears to be reasonable outcome in view of the assumption that $P_0(0) = 1$.

4.3 Fault testing strategy

Faults in a digital circuit are detected by applying test patterns to the primary inputs of the circuit, and observing the output response with respect to these input patterns. In the strategy of random testing, a large number of input patterns are generated randomly. However, if a fault is not sensitive to these generated test patterns, a wrong conclusion may be drawn regarding the existence of the fault. One way to minimize the probability of such a wrong conclusion is to apply a large number of test patterns to the circuit until either the fault is detected, or the confidence about the circuit being error-free is larger than or equal to some prespecified value. In order to increase our confidence in the testing procedure, we must minimize the probability that a fault exists, but is not detected.

Assume that the test patterns are applied to the circuit under test (CUT) continuously from time t_0 to time t_0+s, and the testing is terminated prior to t_0+s, if the fault is detected. We determine the maximum testing time s(max) so that the probability of a wrong conclusion is smaller than or equal to some precalculated value α, that is,

$$P(\text{fault exists, but is not detected during the interval } [t_0, t_0+s] \mid \text{fault exists}) \leq \alpha.$$

This probability may be decomposed, and expressed as:

$$P(\text{fault exists, but is not detected during the interval } [t_0, t_0+s] \mid \text{fault exists}) = P(X \cap Y_1 \cap Z) + P(X \cap Y_2 \cap Z) \leq \alpha,$$

where X, Y_1, Y_2, Z are the events defined as

- X: fault exists in the circuit,
- Y_1: fault causes no error output and error state transition at time t_0,
- Y_2: fault causes only error state transition at time t_0,
- Z: fault causes no error output from time t_0 to time t_0+s.

We then have:

$$P(X \cap Y_1 \cap Z) + P(X \cap Y_2 \cap Z) = P(Z \mid X \cap Y_1) * P(Y_1 \mid X) * P(X) + P(Z \mid X \cap Y_2) * P(Y_2 \mid X) * P(X)$$

With reference to our model, we can now compute the aforesaid probabilities as follows:

$$P(Z \mid X \cap Y_1) = 1 - P_{01}(s)$$

$$= 1 - \left[\left(\frac{D_1}{B}\right) * u(s) + e^{-(A/2)*s} * \left(\frac{-D_1}{B} * \cos\left(\sqrt{B-\frac{A^2}{4}} * s\right) + \frac{2*B*C_1 - A*D_1}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B-\frac{A^2}{4}} * s\right)\right) \right]$$

$$= \left(\frac{B - D_1}{B}\right) * u(s) - e^{-(A/2)*s} * \left(\frac{-D_1}{B} * \cos\left(\sqrt{B-\frac{A^2}{4}} * s\right) + \frac{2*B*C_1 - A*D_1}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B-\frac{A^2}{4}} * s\right) \right)$$

$$P(Z|X \cap Y_2) = 1 - P_{21}(s)$$

$$= 1 - \left[\left(\frac{D_2''}{B}\right) * u(s) + e^{-(A/2)*s} * \left(\frac{-D_2''}{B} * \cos\left(\sqrt{B-\frac{A^2}{4}} * s\right) + \frac{2*B*C_2'' - A*D_2''}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B-\frac{A^2}{4}} * s\right)\right) \right]$$

$$= \left(\frac{B - D_2''}{B}\right) * u(s) - e^{-(A/2)*s} * \left(\frac{-D_2''}{B} * \cos\left(\sqrt{B-\frac{A^2}{4}} * s\right) + \frac{2*B*C_2'' - A*D_2''}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B-\frac{A^2}{4}} * s\right) \right)$$

To determine $P(Y_1|X)$ and $P(Y_2|X)$, assume that the fault existed for a long time prior to time t_0 so that the circuit is in a stable condition, and the steady state probabilities can be used. Hence,

$$P(Y_1|X) = \lim_{t \to \infty} P_0(t) = \frac{D_0}{B},$$

$$P(Y_2|X) = \lim_{t \to \infty} P_2(t) = \frac{D_2}{B}.$$

Finally, let the a priori probability $P(X) = P(\text{fault exists}) = P$. Then, we have:

$$P(X \cap Y_1 \cap Z) + P(X \cap Y_2 \cap Z)$$

$$= p * \frac{D_0}{B} * \left[\left(\frac{B - D_1}{B}\right) * u(s) - e^{-(A/2)*s} * \frac{-D_1}{B} * \cos\left(\sqrt{B - \frac{A^2}{4}} * s\right) \right.$$

$$\left. - e^{-(A/2)*s} * \frac{2*B*C_1 - A*D_1}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B - \frac{A^2}{4}} * s\right) \right]$$

$$+ P * \frac{D_2}{B} * \left[\left(\frac{B - D_2''}{B}\right) * u(s) - e^{-(A/2)*s} * \frac{-D_2''}{B} * \cos\left(\sqrt{B - \frac{A^2}{4}} * s\right) \right.$$

$$\left. - e^{-(A/2)*s} * \frac{2*B*C_2'' - A*D_2''}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B - \frac{A^2}{4}} * s\right) \right]$$

$$\leq \alpha. \qquad (4.16)$$

Now, if we put:

$$\frac{-D_i^*}{B} = g_i,$$

$$\frac{2*B*C_i^* - A*D_i^*}{2 * B * \sqrt{B - \frac{A^2}{4}}} = h_i, \text{ and}$$

$$\sqrt{B - \frac{A^2}{4}} * s = \beta.$$

Then,

$$\frac{-D_i^*}{B} * \cos\left(\sqrt{B - \frac{A^2}{4}} * s\right) + \frac{2*B*C_i^* - A*D_i^*}{2 * B * \sqrt{B - \frac{A^2}{4}}} * \sin\left(\sqrt{B - \frac{A^2}{4}} * s\right)$$

$$= g_i * \cos\beta + h_i * \sin\beta$$

$$= \sqrt{g_i^2 + h_i^2} * \left(\frac{g_i}{\sqrt{g_i^2 + h_i^2}} * \cos\beta + \frac{h_i}{\sqrt{g_i^2 + h_i^2}} * \sin\beta \right)$$

$$= \sqrt{g_i^2 + h_i^2} * (\sin\theta * \cos\beta + \cos\theta * \sin\beta)$$

$$= \sqrt{g_i^2+h_i^2} * (\sin(\theta+\beta))$$

where $\sin\theta = \dfrac{g_i}{\sqrt{g_i^2+h_i^2}}$,

$\cos\theta = \dfrac{h_i}{\sqrt{g_i^2+h_i^2}}$.

Since,

$$-\sqrt{g_i^2 + h_i^2} \le g_i*\cos\beta + h_i*\sin\beta \le \sqrt{g_i^2 + h_i^2}$$

Then,

$$P(X\cap Y_1\cap Z) + P(X\cap Y_2\cap Z)$$

$$= P * \frac{D_0}{B} * \left(\frac{B - D_1}{B} + e^{-(\lambda/2)*s} * \Omega_1\right)$$
$$+P * \frac{D_2}{B} * \left(\frac{B - D_2''}{B} + e^{-(\lambda/2)*s} * \Omega_2\right)$$
$$\le \alpha.$$

where $\Omega_i = \sqrt{g_i^2 + h_i^2}$, $i = 1,2$.

Hence,

$$\frac{P}{B} * e^{-(\lambda/2)*s} * (D_0*\Omega_1 + D_2*\Omega_2) \le \alpha - \frac{P}{B^2} * (D_0*(B-D_1) + D_2*(B-D_2''))$$

Consequently,

$$e^{(\lambda/2)*s} \ge \frac{(D_0*\Omega_1 + D_2*\Omega_2) * \frac{P}{B}}{\alpha - \frac{P}{B^2} * (D_0*(B-D_1) + D_2*(B-D_2''))}$$
$$= W \qquad (4.17)$$

Thus, the testing time required is

$$s = \frac{2}{A} * \ln(W) \qquad (4.18)$$

This equation gives us an upper bound of testing time after parameters of the model and α are determined. The necessary condition for this equation to hold is

$$\alpha > \frac{P}{B^2} * (D_0*(B-D_1) + D_2*(B-D_2")) \qquad (4.19)$$

If it is not satisfied, then no test is needed, because the quality of the circuit is good enough to pass the error probability α without any testing.

For a given circuit, with each line i stuck-at-k, k = 0,1, we may have the corresponding testing time s(i,k) calculated by the above equation. However, in order to test the circuit for an unknown single fault, we must take

$$s(max) = \max_{i=1}^{\text{total \# of lines}} \max_{k=0}^{1} s(i,k)$$

If we apply test input patterns with testing time at least s(max) to the circuit, we will have confidence level $(1-\alpha)$ that no detection error occurs. The procedure can be applied as well to test for any combination of stuck-at-0 or stuck-at-1 faults in the circuit.

5.0 EXPERIMENTATION

A FORTRAN program was written for IBM VM/SP environment to simulate sequential circuits and to derive parameters for the continuous parameter Markov model. The parameters for the Markov model were obtained by varying input signal probability values in a manner such that they meet the required convergence criterion, i.e. $(B - \frac{A^2}{4}) \geq 0$, as derived in the previous chapter. An algorithm for calculating the maximum testing time s(max) as used in the simulation is given below.

5.1 Algorithm for calculating testing time

Step 1: Set variables for the error probability, α;
total input patterns per unit time, L;
total number of simulation run required, R;
the a priori probability of stuck-at fault, P.

Step 2: For the line i of the circuit under consideration, assign the probability of the line having logic 0 to an initial value, possibly 0.

Step 3: Simulate the circuit and perform transition frequency count experiment.

Step 4: Generate the set of Markov parameters λ_is and μ_js and calculate $(B - \frac{A^2}{4})$.

Step 5: If $(B - \frac{A^2}{4}) < 0$, then repeat Steps 3 and 4 with a new probability value.

Step 6: If $(B - \frac{A^2}{4}) > 0$, then find the root by using linear approximation until $(B - \frac{A^2}{4}) < |\varepsilon|$, where ε is an arbitrary number, say, 0.001.

Step 7: Calculate the testing time with the set of Markov parameters λ_is and μ_js found.

Step 8: Repeat Steps 2 to 7 for more simulation runs.

To illustrate the algorithm developed and the simulation process, two examples are given below.

5.2 Example 1: 8-bit SIPO shift register

The first example that we have considered is a National Semiconductor IC, DM74LS164. It is an 8 bit serial in/parallel out shift register. This shift register features gated serial inputs and an asynchronous CLEAR. A LOW logic level at either serial input inhibits entry of the new data, and resets the first flip-flop to the LOW level at the next clock pulse, thus providing complete control over incoming data. The logic diagram and function table of DM74LS164 are given in Fig. 5.1 and Table 5.1, respectively.

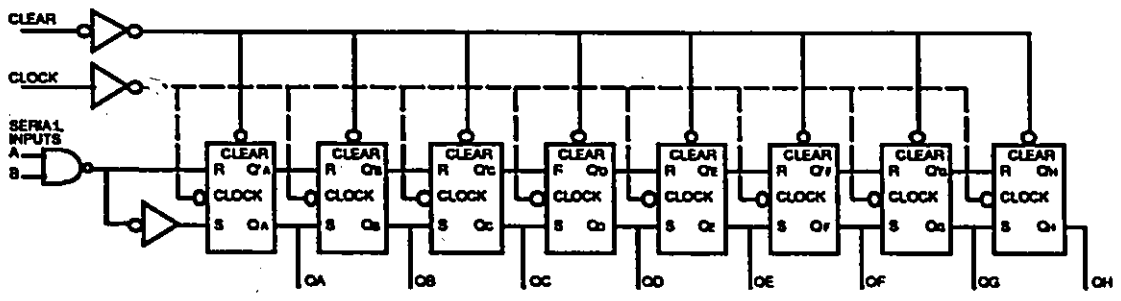

Fig 5.1: Logic representation of DM74LS164

Inputs				Outputs			
CLEAR	Clock	A	B	Q_A	Q_B	...	Q_H
L	X	X	X	L	L	...	L
H	L	X	X	Q_{Ao}	Q_{Bo}	...	Q_{Ho}
H	↑	H	H	H	Q_{An}	...	Q_{Gn}
H	↑	L	X	L	Q_{An}	...	Q_{Gn}
H	↑	X	L	L	Q_{An}	...	Q_{Gn}

Table 5.1: Function table of DM74LS164

where H = logic HIGH level, L = logic LOW level;
X = don't care condition;
↑ = transition from logic LOW to HIGH level;
Q_{Ao} to Q_{Ho} = the previous Q_A to Q_H logic levels;
Q_{An} to Q_{Gn} = the logic levels of preceding FF at clock transition.

To simplify the simulation, the following input conditions are assumed.

1) The CLEAR input was assigned a HIGH logic level for shift operation.
2) Serial inputs A and B were tied together to simulate single bit input.

Since SR flip-flops constituted the logic diagram, the case with R=S=1 was not allowed. Thus, stuck-at-0 faults at inverted outputs of Q_i, for i = A to G, were considered. For each fault category, the rate of generating the input patterns was assumed to be 1000/ms, and a 95% confidence interval for detecting the fault was used. A priori probability of fault was assumed to be 0.1. After 100000

input patterns, the maximum testing times are estimated for the following three cases. All SA0 faults have been considered.

1) Only Q_H was monitored.
2) Both Q_F, Q_G and Q_H were monitored.
3) All outputs were monitored.

Example 5.1

In the first case, for stuck-at-0 fault at the inverted output of flip-flop A, after applying 100000 input patterns with P(input = logic 0) = 0.085, we have

C_{00} = 44383, C_{01} = 0, C_{02} = 4228; and
C_{10} = 4227, C_{11} = 755, C_{12} = 3632; and
C_{20} = 0, C_{21} = 7859, C_{22} = 34916.

Then, the model parameters are calculated as follows :

$\mu_0 = C_{01}/100$ ms = 0.0/ms,
$\mu_1 = C_{12}/100$ ms = 36.3/ms,
$\mu_2 = C_{20}/100$ ms = 0.0/ms,
$\lambda_0 = C_{10}/100$ ms = 42.3/ms,
$\lambda_1 = C_{21}/100$ ms = 78.6/ms,
$\lambda_2 = C_{02}/100$ ms = 42.3/ms.

The estimated testing time will then be 0.046 ms.

Results of the three cases are presented in Tables 5.2 through 5.4, and in Figures 5.2 through 5.4, respectively. P(0) is the probability of having logic 0 at input.

SA0 fault	P(0)	Parameters, # of trans/ms	testing time, ms
INV(Q_A)	0.085	λ_0=0.0, λ_1=36.3, λ_2=0.0, μ_0=42.3, μ_1=78.6, μ_2=42.3.	0.046
INV(Q_B)	0.114	λ_0=0.0, λ_1=40.8, λ_2=0.0, μ_0=47.4, μ_1=88.2, μ_2=47.4.	0.042
INV(Q_C)	0.114	λ_0=0.0, λ_1=47.0, λ_2=0.0, μ_0=54.8, μ_1=101.8, μ_2=54.8.	0.036
INV(Q_D)	0.145	λ_0=0.0, λ_1=57.4, λ_2=0.0, μ_0=66.5, μ_1=123.9, μ_2=66.5.	0.031
INV(Q_E)	0.186	λ_0=0.0, λ_1=70.3, λ_2=0.0, μ_0=81.3, μ_1=151.6, μ_2=81.3.	0.027
INV(Q_F)	0.269	λ_0=0.0, λ_1=90.8, λ_2=0.0, μ_0=104.9, μ_1=195.7, μ_2=104.9.	0.023
INV(Q_G)	0.465	λ_0=0.0, λ_1=115.2, λ_2=0.0, μ_0=133.3, μ_1=248.5, μ_2=133.3.	0.017

Table 5.2: Results with last output monitored

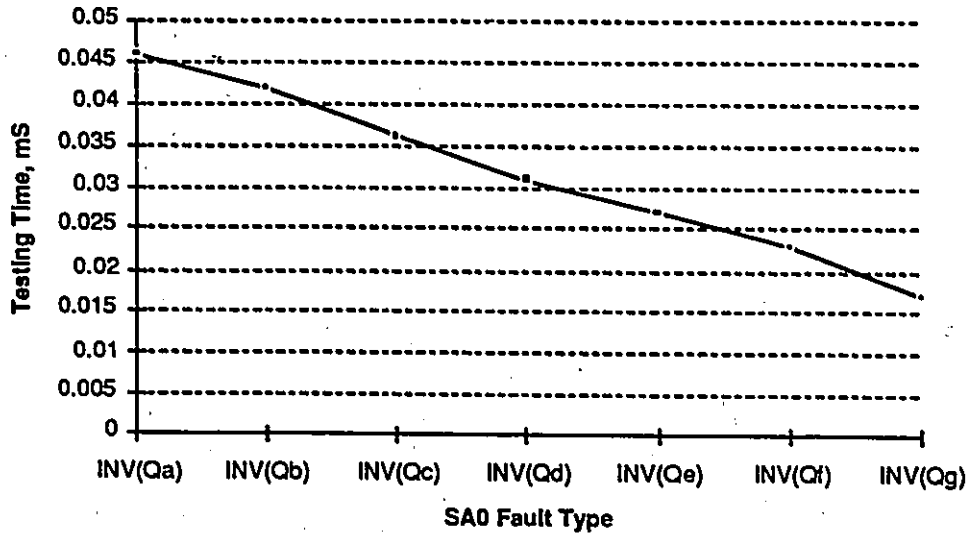

Fig 5.2: Testing time vs fault type for case 1

SA0 fault	P(0)	Parameters, # of trans/ms	Testing Time, ms
INV(Q_A)	0.117	λ_0=0.0, λ_1=37.5, λ_2=0.0, μ_0=43.5, μ_1=81.0, μ_2=43.5.	0.047
INV(Q_B)	0.146	λ_0=0.0, λ_1=41.9, λ_2=0.0, μ_0=48.7, μ_1=90.6, μ_2=48.7.	0.041
INV(Q_C)	0.188	λ_0=0.0, λ_1=46.4, λ_2=0.0, μ_0=54.1, μ_1=100.5, μ_2=54.1.	0.035
INV(Q_D)	0.268	λ_0=0.0, λ_1=48.8, λ_2=0.0, μ_0=56.9, μ_1=105.7, μ_2=56.9.	0.034
INV(Q_E)	0.464	λ_0=0.0, λ_1=33.1, λ_2=0.0, μ_0=38.4, μ_1=71.5, μ_2=38.4.	0.056
INV(Q_F)	0.463	λ_0=0.0, λ_1=61.7, λ_2=0.0, μ_0=71.5, μ_1=133.3, μ_2=71.5.	0.030
INV(Q_G)	0.462	λ_0=0.0, λ_1=114.9, λ_2=0.0, μ_0=132.8, μ_1=247.7, μ_2=132.8.	0.019
INV(Q_A)+ INV(Q_D)+ INV(Q_G)	0.117	λ_0=0.0, λ_1=37.2, λ_2=0.0, μ_0=43.2, μ_1=80.4, μ_2=43.2.	0.047
all INV(Q)	0.118	λ_0=0.0, λ_1=37.2, λ_2=0.0, μ_0=43.1, μ_1=80.4, μ_2=43.1.	0.049

Table 5.3: Results with last three outputs monitored

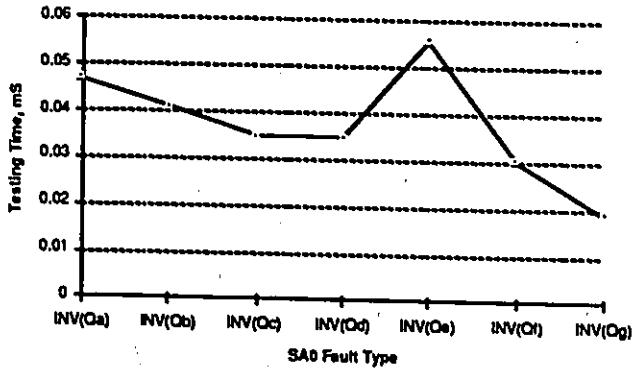

Fig 5.3: Testing time vs fault type for case 2

SA0 fault	P(0)	Parameters, # of trans/ms	Testing Time, ms
INV(Q_A)	0.487	λ_0=0.0, λ_1=2.1, λ_2=0.0, μ_0=2.6, μ_1=4.6, μ_2=2.6.	0.607
INV(Q_B)	0.475	λ_0=0.0, λ_1=4.5, λ_2=0.0, μ_0=5.5, μ_1=9.9, μ_2=5.5.	0.292
INV(Q_C)	0.458	λ_0=0.0, λ_1=9.6, λ_2=0.0, μ_0=11.4, μ_1=21.0, μ_2=11.4.	0.154
INV(Q_D)	0.463	λ_0=0.0, λ_1=17.9, λ_2=0.0, μ_0=21.2, μ_1=39.1, μ_2=21.2.	0.082
INV(Q_E)	0.461	λ_0=0.0, λ_1=33.1, λ_2=0.0, μ_0=38.5, μ_1=71.6, μ_2=38.5.	0.052
INV(Q_F)	0.463	λ_0=0.0, λ_1=61.7, λ_2=0.0, μ_0=71.2, μ_1=132.9, μ_2=71.2.	0.038
INV(Q_G)	0.466	λ_0=0.0, λ_1=115.1, λ_2=0.0, μ_0=133.0, μ_1=248.1, μ_2=133.0.	0.018

Table 5.4: Results with all outputs monitored

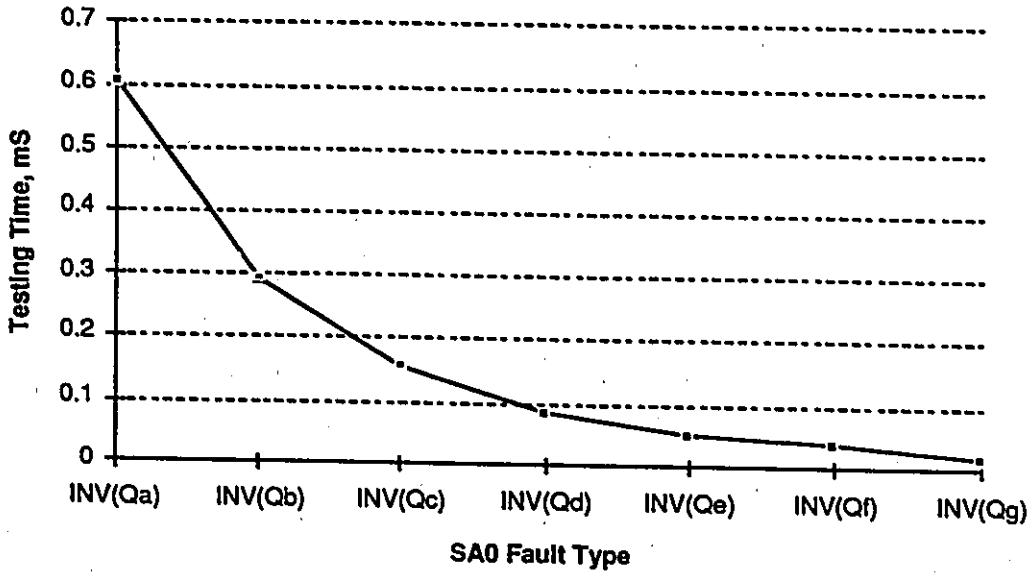

Fig 5.4: Testing time vs fault type for case 3

5.3 Example 2: 4-bit up/down counter

The second example that we have considered is another IC from National Semiconductor Corporation, DM74LS191. It is a synchronous 4-bit up/down counter. The synchronous operation is provided by having all flip-flops clocked simultaneously. The outputs of the four master-slave flip-flops are triggered on a LOW-to-HIGH level transition of the clock input, if the ENABLE input is LOW. A HIGH at the ENABLE input inhibits counting. The direction of the count is determined by the level of the UP/DOWN input. When LOW, the counter counts up and vice versa.

The counter is also fully programmable; that is, outputs may be preset by placing a LOW on the LOAD input and entering desired data at DATA INPUT A to D. The output will change independent of the clock input. Also, two outputs, RIPPLE CLOCK and MAX/MIN, are provided for cascading purposes. The logic diagram of DM74LS191 in given in Fig. 5.5.

To simplify the simulation, the following input conditions are assumed.

1) The ENABLE input was assigned a LOW logic level for count operation.
2) The LOAD input was assigned a HIGH logic level for continuous counting.

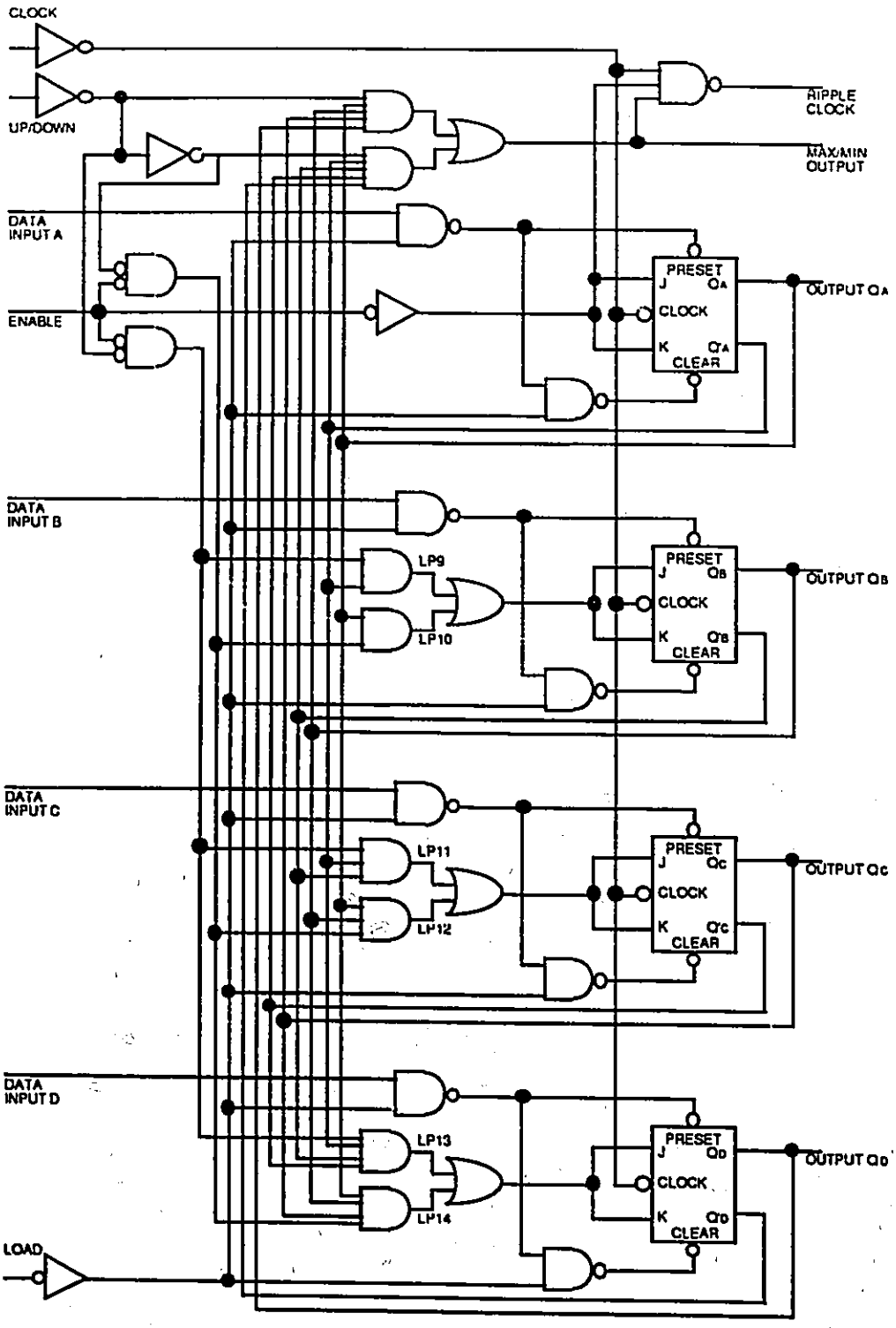

Fig 5.5: Logic representation of DM74LS191

We have assumed stuck-at-0 faults at location LP9 to LP14 as in Fig.5.5. For each fault category, the rate of generating the input patterns was assumed to be 1000/mS, and a 95% confidence interval for detecting the fault was used. A priori probability of fault was assumed to be 0.1. After 100000 input patterns, the maximum testing times are estimated for the following three cases.

1) Only Q_D was monitored.
2) Both Q_C and Q_D were monitored.
3) All flip-flop outputs were monitored.

Results of three cases are presented in Tables 5.5 through 5.7, respectively. P(0) is the probability of having logic 0 at input.

SA0 fault	P(0)	Parameters, # of trans/ms	Testing Time, ms
LP9	--	$(B - \frac{A^2}{4}) < 0.$	--
LP10	1.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP11	0.464	λ_0=0.0, λ_1=28.7, λ_2=0.0, μ_0=34.1, μ_1=62.8, μ_2=34.1.	0.050
LP12	0.537	λ_0=0.0, λ_1=28.0, λ_2=0.0, μ_0=34.8, μ_1=62.8, μ_2=34.8.	0.044
LP13	0.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP14	1.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022

Table 5.5: Results with Q_D monitored

SA0 fault	P(0)	Parameters, # of trans/ms	Testing Time, ms
LP9	0.456	λ_0=0.0, λ_1=29.4, λ_2=0.0, μ_0=34.5, μ_1=63.9, μ_2=34.5.	0.053
LP10	0.532	λ_0=0.0, λ_1=28.3, λ_2=0.0, μ_0=33.0, μ_1=61.3, μ_2=33.0.	0.059
LP11	0.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP12	1.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP13	0.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP14	1.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022

Table 5.6: Results with Q_C and Q_D monitored

SA0 fault	P(0)	Parameters, # of trans/ms	Testing Time, ms
LP9	0.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP10	1.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP11	0.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP12	1.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP13	0.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022
LP14	1.000	λ_0=0.0, λ_1=0.0, λ_2=0.0, μ_0=62.5, μ_1=62.5, μ_2=62.5.	0.022

Table 5.7: Results with all flip-flop outputs monitored

6.0 DISCUSSIONS

Two sequential logic circuits, DM74164 and DM74191, have been simulated and results are presented in Chapter 5. Some of the observations are given below.

6.1 Observations

1) Besides the two end points, there is usually one root for $(B - \frac{A^2}{4})$, i.e. the value equal to 0, as shown in Fig. 6.1. Thus, to speed up the simulation, we can stop the search once the root has been found. Also, for the example in Fig. 6.1, the value of $(B - \frac{A^2}{4})$ ranges from 630 to -7300. The arbitrary number ε mentioned in Step 6 of the proposed algorithm can be set at 3% of max value of $(B - \frac{A^2}{4})$ to minimize the iteration involved.

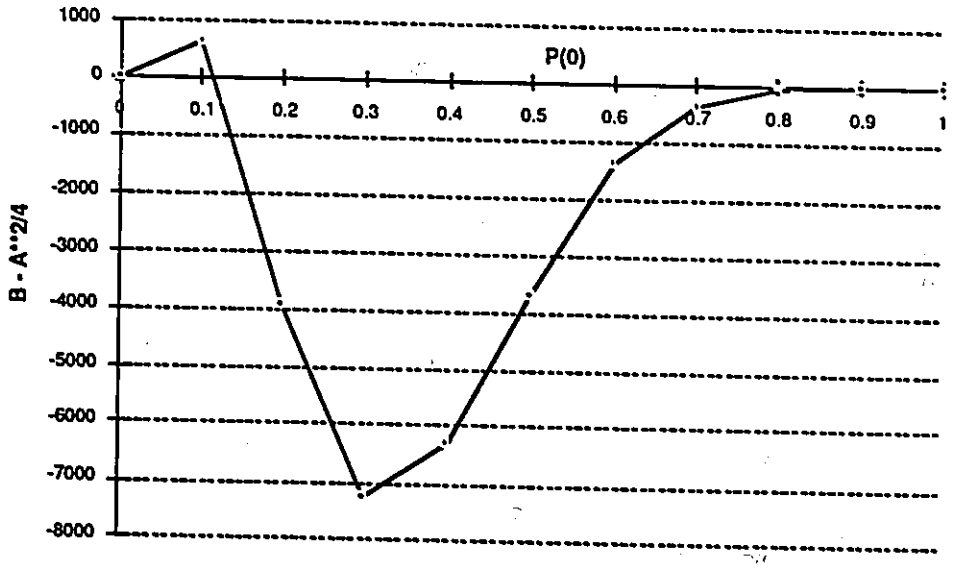

Fig 6.1: Typical curve of P(0) vs. $(B - \frac{A^2}{4})$

2) $\lim_{t\to\infty} P_0(t) = \lim_{t\to\infty} P_1(t) = \lim_{t\to\infty} P_2(t)$

where, $\lim_{t\to\infty} P_0(t) = \frac{D_0}{B}$

$\lim_{t\to\infty} P_1(t) = \frac{D_1}{B}$

$\lim_{t\to\infty} P_2(t) = \frac{D_2}{B}$

Also, $D_0 = D_1 = D_2$

This leads to the fact that for an N state continuous time Markov model,

$$\lim_{t\to\infty} P_i(t) = \frac{1}{N} \qquad \text{where } i = 1 \text{ to } N. \tag{6.1}$$

This implies that for an N state Markov model, the chance of reaching each state, in spite of initial state probabilities, will be the same. This can also be extended to transition probabilities. Thus we have

$$\lim_{t\to\infty} P_{ij}(t) = \frac{1}{N} \tag{6.2}$$

where, $i = 1$ to N, $j = 1$ to N;

And, $D_0' = D_1' = D_2'$,

$D_0'' = D_1'' = D_2''$.

3) $(\mu_i - \lambda_i)$ = constant for $i = 0$ to 2. This implies that there are net rates of transitions from state 0 to state 2, state 2 to state 1 and state 1 to state 0. This agrees with the Eq. 6.1 above in the way that at any time instant, if there is a transition, the probability of reaching any possible state is 1/3.

Also, since our model is strongly connected, each state can be reached from all other states, so that between

certain time interval, the transition probability of one state to all states is also 1/3. This agrees with Eq. 6.2 above.

4) As time s becomes large, the Eq. 4.16 can be simplified as

$$P * [\frac{D_0}{B} * (\frac{B - D_1}{B}) + \frac{D_2}{B} * (\frac{B - D_2''}{B})] * u(s) \leq \alpha \qquad (6.3)$$

Then, from point 2 above, the Eq. 6.3 will give the following inequality.

$$\frac{4}{9} * P \leq \alpha$$

where P is the a priori probability that a fault exists. This sets the limit of up to which confidence level a fault can be detected within the maximum testing time obtained from Equations 4.17 and 4.18.

5) From Example 1, we may note :

a) The testing times required are location dependent. When the fault gets farther from the observed output(s), the masking effect may make it less detectable and we may take longer time to manifest the effect of fault.

b) To overcome the masking effect, we need a different P(0), input probability of having logic 0. The P(0) values follow an exponential pattern. However, it approaches 0.5 and level off in the case of multiple outputs being observed as shown in Fig. 6.2.

c) Multiple faults can be detected with the estimated testing time and the P(0) requirement as the hardest detected SA0 fault, i.e. SA0 fault at inverted output of SR flip-flop A. Due to the masking effect, if the

input patterns can detect the hardest detected SA0 fault, it can detect all others.

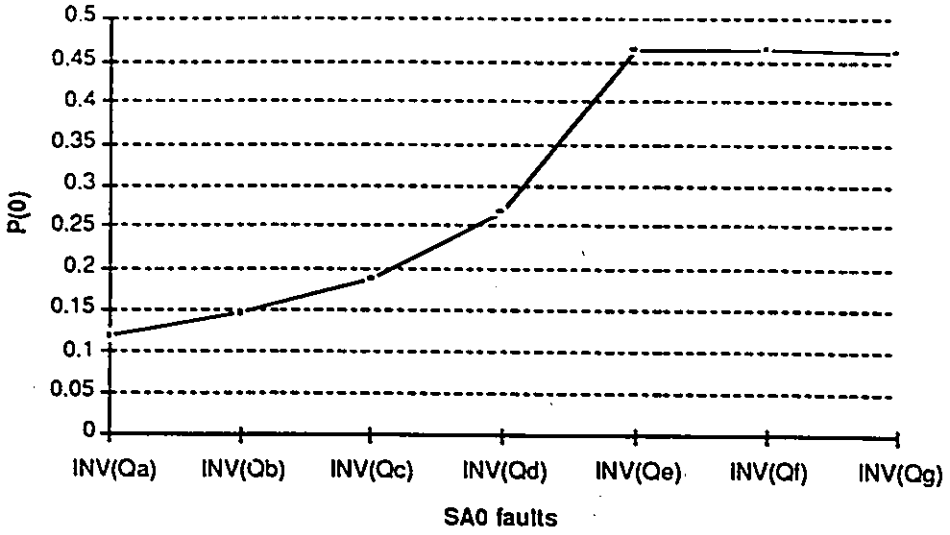

Fig 6.2: P(0) vs different stuck-at-0 faults for case 3 of example 1

6) From Example 2, we may note :

a) The sum of P(0) values in case of SA0 faults at LP_i and LP_{i+1} = 1.0, where i = 9,11,13. It follows the fact that LP9, LP11 and LP13 are related to count down operation and others are related to count up operation. The count down and count up operations are basically symmetrical.

6.2 Conclusions

A continuous parameter Markov model for detecting permanent stuck-at faults in synchronous sequential circuits by random testing is proposed in the thesis. The developed model, the related mathematical analysis and the examples provide some useful insights into the nature of faults in relation to random testing in sequential logic circuits and the associated confidence measure. The proposed approach does not require the formation of a product state table corresponding to the fault-free and the faulty state tables of the circuit; instead, the state tables of the fault free circuit and its faulty version are required to simulate the behavior of the faults.

BIBLIOGRAPHY

[1] J. J. Shedletsky and E. J. McCluskey, "The Error Latency of a Fault in a Combinational Digital Circuit", 1975 Int. Symp. on Fault-Tolerant Computing Digest (FTCS-5), pp. 210-214, June 1975.

[2] J. J. Shedletsky and E. J. McCluskey, "The Error Latency of a Fault in a Sequential Digital Circuit", IEEE Transactions on Computers, pp.655-659, June 1976.

[3] S. Y. H. Su, I. Koren, and Y. K. Malaiya, "A Continuous-Parameter Markov Model and Detection Procedures for Intermittent Faults", IEEE Transactions on Computers, pp. 567-570, June 1978.

[4] R. David and G. Blanchet, "About Random Fault Detection of Combinational Network", IEEE Transactions on Computers, pp. 659-664, June 1976.

[5] P. Agrawal and V. D. Agrawal, "Probabilistic Analysis of Random Test Generation Method for Irredundant Combinational Logic Networks", IEEE Transactions on Computers, pp. 691-694, July 1975.

[6] K. P. Parker and E. J. McCluskey, "Analysis of Logic Circuits with Faults using Input Signal Probabilities", IEEE Transactions on Computers, pp. 573-578, May 1975.

[7] K. P. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinational Networks", IEEE Transactions on Computers, pp. 668-670, June 1975.

[8] D. Bastin, E. Girard, J. C. Rault and R. Tullioue, "Probabilistic Test Generation Methods : Statistical Estimation of Sequence Length and Efficiency", FTCS-3, p. 171, 1973.

[9] H. D. Schnurmann, E. Lindbloom and R. G. Carpenter, "The Weighted Random Test-Pattern Generator", IEEE Transactions on Computers, pp. 695-700, July 1975.

[10] P. Agrawal and V. D. Agrawal, "On Improving the Efficiency of Monte Carlo Test Generation", FTCS-5, pp. 205-209, 1975.

[11] V. D. Agrawal, "When to Use Random Testing", IEEE Transactions on Computers, pp. 1054-1055, November 1978.

[12] P. B. Schneck, "Comment on 'When to Use Random Testing'", IEEE Transactions on Computers, pp. 580-581, August 1979.

[13] R. David and P. Thevenod-Fosse, "Random Testing of Integrated Circuits", IEEE Transactions on Computers, pp. 20-25, March 1981.

[14] P. Thevenod-Fosse, "Asynchronous Random Testing of Sequential Circuits", FTCS-8, p. 213, 1978.

[15] J. Savir, G. S. Ditlow, and P. H. Bardell, "Random Pattern Testability", IEEE Transactions on Computers, pp. 79-90, January 1984.

[16] E. B. Eichelberger and E. Lindbloom, "Random-Pattern Coverage Enhancement and Diagnosis for LSSD Logic Self-Test", IBM J. Res. Develop., pp. 265-272, May 1983.

[17] J. Savir and P. H. Bardell, "On Random Pattern Test Length", IEEE Transactions on Computers, pp. 467-474, June 1984.

[18] J. Losq, "Efficiency of Random Compact Testing", IEEE Transactions on Computers, pp. 516-525, June 1978.

[19] J. C. Rault, "A Graph Theoretical and Probabilistic Approach to the Fault Detection of Digital Circuits", FTCS-1, pp. 26-29, 1971.

[20] M. A. Breuer, "A Random and an Algorithmic Technique for Fault Detection Test Generation for Sequential Circuits", IEEE Transactions on Computers, pp. 1364-1370, November 1971.

[21] R. David and R. Tellez-Giron, "Comments on 'The Error Latency of a Fault in a Sequential Digital Circuit'", IEEE Transactions on Computers, pp. 85-86, January 1979.

[22] M. A. Breuer, "Testing for Intermittent Faults in Digital Circuits", IEEE Transactions on Computer, pp. 241-246, March 1973.

[23] S. Kamal and C. V. Page, "Intermittent Faults : A Model and a Detection Procedure", IEEE Transactions on Computers, pp. 713-719, July 1974.

[24] S. Kamal, "An Approach to the Diagnosis of Intermittent Faults", IEEE Transactions on Computers, pp. 461-467, May 1975.

[25] K. G. Shin and Y. H. Lee, "Measurement and Application of Fault Latency", IEEE Transactions on Computers, pp. 370-374, April 1986.

[26] E. I. Muehldorf and A. D. Savkar, "LSI Logic Testing - An Overview", IEEE Transactions on Computers, pp. 1-17, January 1981.

[27] R. David and P. Thevenod-Fosse, "Minimal Detecting Transition Sequences : Application to Random Testing", IEEE Transactions on Computers, pp. 514-518, June 1980.

[28] D. P. Siewiorek and L. K. W. Lai, "Testing of Digital Systems", IEEE Transactions on Computers, pp. 1321-1333, October 1981.

[29] K. P. Parker, "Probabilistic Test Generation", FTCS-3, p. 179, 1973.

[30] J. F. Meyer and K. Yeh, "Diagnosable Machine Realizations of Sequential Behavior", FTCS-1, pp. 22-25, 1971.

[31] M. A. Breuer, "Generation of Fault Detection Tests for Sequential Circuits", FTCS-1, pp. 18-21, 1971.

[32] S. W. Golomb, "Shift Register Sequences", revised ed., Laguna Hills, CA: Agegean Park, 1982.

[33] P. K. Lala, "Fault Tolerant and Fault Testable Hardware Design", Prentice-Hall International Inc., London, 1985.

[34] S. C. Lee, "Digital Circuits and Logic Design", Prentice-Hall Inc., Englewood Cliffs, N.J., 1976.

[35] L. Takacs, "Stochastic Processes, Problems and Solutions", John Wiley & Sons Inc., N.J., 1960.

[36] J. L. Doob, "Stochastic Processes", John Wiley & Sons Inc., N.J., 1958.

[37] S. R. Das, P. K. Srimani and C. R. Datta, "On Multiple Fault Analysis in Combinational Circuits by Means of Boolean Difference", Proceedings of IEEE, pp. 1447-1449, September 1976.

[38] S. R. Das and A. Bhattacharyya, "Fault Detection in Sequential Machines with Increased Fault Coverage", Electronics Letters, pp. 28-29, January 1978.

[39] S. R. Das, C. L. Sheng and Z. Chen, "An Algorithm for Finding all Maximal Complete Subgraphs and an Estimate of the Order of Computational Complexity", Computers and Electrical Engineering, pp. 365-368, December 1978.

[40] S. R. Das, Z. Chen, S. M. Wu, S. Y. Lee and A. Bhattacharyya, "On the Design of Improved Failure Detection Experiments in Synchronous Sequential Machines Based on Terminal Measurements" Computers and Electrical Engineering, pp. 293-297, December 1979.

[41] S. R. Das and W. B. Jone, "Modified Transition Matrix and Fault Testing in Sequential Logic Circuits under Random Stimuli with a Specified Measure of Confidence", Cybernetics and Systems : An International Journal, pp. 1-12, 1986.

[42] S. R. Das, "Random Test Generation for LSI/VLSI Circuits - a Survey", IEEE VLSI Technical Bulletin, pp. 4-11, March 1987.

[43] S. R. Das, K. W. Chiang and W. B. Jone, "A First-Order Optimal Algorithm for State Identification in Sequential Logic using the Concept of Entropy", Cybernetics and Systems : An International Journal, pp. 251-270, September 1987.

[44] S. R. Das, "Adaptive Scheduled Experimentation and Fault Location in Large Combinational Logic Networks", Cybernetics and Systems : An International Journal, pp. 1-12, January 1988.

[45] S. R. Das, "Nontransient Fault Testing in Sequential Logic Circuits using Stochastic Modeling and Simulation", Cybernetics and Systems : An International Journal, pp. 15-27, January 1989.

[46] S. R. Das, W. B. Jone, G. E. Fares and A. R. Nayak, "Probabilistic Fault Location in Combinational Logic Network using Concepts of Fault Distance and Input Feature", Cybernetics and Systems : An International Journal, pp. 385-399, December 1989.

APPENDIX A1:

FORTRAN program to simulate DM74LS164.

```
      INTEGER Q(8),NQ(8),OUT(64,64),NS(256,256),J(8),K(8),
     *        INP(8),TP,TP1,TP2,TP3,T(8),D(8),R(8),S(8),FFT,
     *        AND2,AND3,AND4,AND5,OR2,OR3,NAND2,NAND3,INV
C
C
C     PARAMETERS DESCRIPTION :
C
C        Q(I),NQ(I) : FLIP-FLOP I'S CONDITION
C        OUT(I,J) : OUTPUT WITH PRESENT STATE I & INPUT J
C        NS(I,J) : NEXT STATE WITH PRESENT STATE I & INPUT J
C        J(I),K(I) : INPUT CONDITIONS OF JK FLIP-FLOP I
C        R(I),S(I) : INPUT CONDITIONS OF RS FLIP-FLOP I
C        D(I) : INPUT CONDITION OF D FLIP-FLOP I
C        T(I) : INPUT CONDITION OF T FLIP-FLOP I
C        INP(I) : CONDITION OF INPUT LINE I
C        TP,TP1,TP2,TP3 : TEMPORARY STORAGE
C        FFT : VALUE FOR FLIP-FLOP TYPE :-
C              1 FOR D FLIP-FLOP
C              2 FOR T FLIP-FLOP
C              3 FOR RS FLIP-FLOP
C              4 FOR JK FLIP-FLOP
C        NFF : TOTAL NUMBER OF FLIP-FLOP
C        NINP : TOTAL NUMBER OF INPUT LINE
C
      FFT = 3
      NFF = 8
      NINP = 1
      NFF1 = 2 ** NFF
      NINP1 = 2 ** NINP
      ENG = 0
      DO 10 I = 1,NINP1
       TP = I - 1
       DO 20 I1 = 1,NINP
        INP(I1) = 0
        TP1 = TP / 2
        IF (2*TP1.NE.TP .AND. TP.NE.0) INP(I1) = 1
        TP = TP1
   20   CONTINUE
       DO 30 I2 = 1,NFF1
        TP2 = I2 - 1
        DO 40 I3 = 1,NFF
         Q(NFF-I3+1) = 0
         TP3 = TP2 / 2
         IF (2*TP3.NE.TP2 .AND. TP2.NE.0) Q(NFF-I3+1) = 1
         NQ(NFF-I3+1) = INV(Q(NFF-I3+1))
         TP2 = TP3
   40    CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     DESCRIBE THE INPUT CONDITIONS OF FLIP-FLOPS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        S(1) = INP(1)
        R(1) = INV(INP(1))
```

```
      S(2) = Q(1)
      R(2) = INV(Q(1))
      S(3) = Q(2)
      R(3) = INV(Q(2))
      S(4) = Q(3)
      R(4) = INV(Q(3))
      S(5) = Q(4)
      R(5) = INV(Q(4))
      S(6) = Q(5)
      R(6) = INV(Q(5))
      S(7) = Q(6)
      R(7) = INV(Q(6))
      S(8) = Q(7)
      R(8) = INV(Q(7))
      OUT(I,I2) = 0
      DO 55 IOUT = 1,NFF
       OUT(I,I2) = OUT(I,I2) + Q(IOUT) * 2**(NFF-IOUT)
 55   CONTINUE
      NS(I,I2) = 1
      DO 60 I5 = 1,NFF
       GOTO (1,2,3,4) FFT
  1    NQ(I5) = 0
       IF ( D(I5).EQ.1 ) NQ(I5) = 1
       GOTO 70
  2    NQ(I5) = Q(I5)
       IF ( T(I5).EQ.1 ) NQ(I5) = MOD( Q(I5)+1,2 )
       GOTO 70
  3    IF ( R(I5).EQ.S(I5) ) THEN
        NQ(I5) = Q(I5)
        IF ( R(I5).EQ.1 ) WRITE(6,101) I5
        GOTO 70
       ENDIF
 101   FORMAT(' BOTH INPUTS OF RS F.F.',I1,' ARE 1')
       NQ(I5) = S(I5)
       GOTO 70
  4    IF ( J(I5).EQ.K(I5) ) THEN
        NQ(I5) = Q(I5)
        IF ( J(I5).EQ.1 ) NQ(I5) = INV(Q(I5))
        GOTO 70
       ENDIF
       NQ(I5) = J(I5)
 70    NS(I,I2) = NS(I,I2) + NQ(I5) * 2**(NFF-I5)
 60   CONTINUE
 30  CONTINUE
     WRITE(6,222) ( NS(I,I2),OUT(I,I2), I2 = 1,NFF1 )
222  FORMAT(2I4)
 10  CONTINUE
     STOP
     END
C
     INTEGER FUNCTION AND2(I1,I2)
     AND2 = I1 * I2
     RETURN
     END
```

```
C
      INTEGER FUNCTION AND3(I1,I2,I3)
      AND3 = I1 * I2 * I3
      RETURN
      END
C
      INTEGER FUNCTION AND4(I1,I2,I3,I4)
      AND4 = I1 * I2 * I3 * I4
      RETURN
      END
C
      INTEGER FUNCTION AND5(I1,I2,I3,I4,I5)
      AND5 = I1 * I2 * I3 * I4 * I5
      RETURN
      END
C
      INTEGER FUNCTION OR2(I1,I2)
      OR2 = I1 + I2
      IF (OR2 .GT. 1) THEN OR2 = 1
      RETURN
      END
C
      INTEGER FUNCTION OR3(I1,I2,I3)
      OR3 = I1 + I2 + I3
      IF (OR3 .GT. 1) THEN OR3 = 1
      RETURN
      END
C
      INTEGER FUNCTION NAND2(I1,I2)
      NAND2 = I1 * I2
      IF (NAND2 .EQ. 1) THEN
       NAND2 = 0
      ELSE
       NAND2 = 1
      ENDIF
      RETURN
      END
C
      INTEGER FUNCTION NAND3(I1,I2,I3)
      NAND3 = I1 * I2 * I3
      IF (NAND3 .EQ. 1) THEN
       NAND3 = 0
      ELSE
       NAND3 = 1
      ENDIF
      RETURN
      END
C
      INTEGER FUNCTION INV(I1)
      INV = I1
      IF (INV .EQ. 1) THEN
       INV = 0
      ELSE
       INV = 1
```

```
ENDIF
RETURN
END
```

APPENDIX A2:

FORTRAN program to simulate DM74LS191.

```
      INTEGER Q(8),NQ(8),OUT(64,64),NS(256,256),J(8),K(8),
     *        INP(8),TP,TP1,TP2,TP3,T(8),D(8),R(8),S(8),FFT,
     *        AND2,AND3,AND4,AND5,OR2,OR3,NAND2,NAND3,INV,
     *        DOWNUP,ENG
C
C
C     PARAMETERS DESCRIPTION :
C
C        Q(I),NQ(I) : FLIP-FLOP I'S CONDITION
C        OUT(I,J) : OUTPUT WITH PRESENT STATE I & INPUT J
C        NS(I,J) : NEXT STATE WITH PRESENT STATE I & INPUT J
C        J(I),K(I) : INPUT CONDITIONS OF JK FLIP-FLOP I
C        R(I),S(I) : INPUT CONDITIONS OF RS FLIP-FLOP I
C        D(I) : INPUT CONDITION OF D FLIP-FLOP I
C        T(I) : INPUT CONDITION OF T FLIP-FLOP I
C        INP(I) : CONDITION OF INPUT LINE I
C        TP,TP1,TP2,TP3 : TEMPORARY STORAGE
C        FFT : VALUE FOR FLIP-FLOP TYPE :-
C              1 FOR D FLIP-FLOP
C              2 FOR T FLIP-FLOP
C              3 FOR RS FLIP-FLOP
C              4 FOR JK FLIP-FLOP
C        NFF : TOTAL NUMBER OF FLIP-FLOP
C        NINP : TOTAL NUMBER OF INPUT LINE
C
      FFT = 4
      NFF = 4
      NINP = 1
      NFF1 = 2 ** NFF
      NINP1 = 2 ** NINP
      ENG = 0
      DO 10 I = 1,NINP1
       TP = I - 1
       DO 20 I1 = 1,NINP
        INP(I1) = 0
        TP1 = TP / 2
        IF (2*TP1.NE.TP .AND. TP.NE.0) INP(I1) = 1
        TP = TP1
  20   CONTINUE
       DO 30 I2 = 1,NFF1
        TP2 = I2 - 1
        DO 40 I3 = 1,NFF
         Q(NFF-I3+1) = 0
         TP3 = TP2 / 2
         IF (2*TP3.NE.TP2 .AND. TP2.NE.0) Q(NFF-I3+1) = 1
         NQ(NFF-I3+1) = INV(Q(NFF-I3+1))
         TP2 = TP3
  40    CONTINUE
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     DESCRIBE THE INPUT CONDITIONS OF FLIP-FLOPS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
        DOWNUP = INP(1)
```

```
      LP1 = INV(DOWNUP)
      LP2 = INV(LP1)
      LP3 = INV(ENG)
      LP4 = INV(LOAD)
      LP5 = AND5(LP1,Q(1),Q(2),Q(3),Q(4))
      LP6 = AND5(LP2,INV(Q(1)),INV(Q(2)),INV(Q(3)),INV(Q(4)))
      MMOUT = OR2(LP5,LP6)
      LP7 = AND2(INV(LP2),LP3)
      LP8 = AND2(LP3,INV(LP1))
      LP9 = AND2(LP8,INV(Q(1)))
      LP10 = AND2(Q(1),LP7)
      LP11 = AND3(LP8,INV(Q(1)),INV(Q(2)))
      LP12 = AND3(Q(1),Q(2),LP7)
      LP13 = AND4(LP8,INV(Q(1)),INV(Q(2)),INV(Q(3)))
      LP14 = AND4(Q(1),Q(2),Q(3),LP7)
      J(1) = LP3
      K(1) = J(1)
      J(2) = OR2(LP9,LP10)
      K(2) = J(2)
      J(3) = OR2(LP11,LP12)
      K(3) = J(3)
      J(4) = OR2(LP13,LP14)
      K(4) = J(4)
      OUT(I,I2) = 0
      DO 55 IOUT = 1,NFF
       OUT(I,I2) = OUT(I,I2) + Q(IOUT) * 2**(NFF-IOUT)
 55   CONTINUE
      NS(I,I2) = 1
      DO 60 I5 = 1,NFF
       GOTO (1,2,3,4) FFT
  1    NQ(I5) = 0
       IF ( D(I5).EQ.1 ) NQ(I5) = 1
       GOTO 70
  2    NQ(I5) = Q(I5)
       IF ( T(I5).EQ.1 ) NQ(I5) = MOD( Q(I5)+1,2 )
       GOTO 70
  3    IF ( R(I5).EQ.S(I5) ) THEN
        NQ(I5) = Q(I5)
        IF ( R(I5).EQ.1 ) WRITE(6,101) I5
        GOTO 70
       ENDIF
101    FORMAT(' BOTH INPUTS OF RS F.F.',I1,' ARE 1')
       NQ(I5) = S(I5)
       GOTO 70
  4    IF ( J(I5).EQ.K(I5) ) THEN
        NQ(I5) = Q(I5)
        IF ( J(I5).EQ.1 ) NQ(I5) = INV(Q(I5))
        GOTO 70
       ENDIF
       NQ(I5) = J(I5)
 70    NS(I,I2) = NS(I,I2) + NQ(I5) * 2**(NFF-I5)
 60   CONTINUE
 30  CONTINUE
     WRITE(6,222) ( NS(I,I2),OUT(I,I2), I2= 1,NFF1 )
```

```
 222   FORMAT(2I4)
  10   CONTINUE
       STOP
       END
C
       INTEGER FUNCTION AND2(I1,I2)
       AND2 = I1 * I2
       RETURN
       END
C
       INTEGER FUNCTION AND3(I1,I2,I3)
       AND3 = I1 * I2 * I3
       RETURN
       END
C
       INTEGER FUNCTION AND4(I1,I2,I3,I4)
       AND4 = I1 * I2 * I3 * I4
       RETURN
       END
C
       INTEGER FUNCTION AND5(I1,I2,I3,I4,I5)
       AND5 = I1 * I2 * I3 * I4 * I5
       RETURN
       END
C
       INTEGER FUNCTION OR2(I1,I2)
       OR2 = I1 + I2
       IF (OR2 .GT. 1) THEN OR2 = 1
       RETURN
       END
C
       INTEGER FUNCTION OR3(I1,I2,I3)
       OR3 = I1 + I2 + I3
       IF (OR3 .GT. 1) THEN OR3 = 1
       RETURN
       END
C
       INTEGER FUNCTION NAND2(I1,I2)
       NAND2 = I1 * I2
       IF (NAND2 .EQ. 1) THEN
        NAND2 = 0
       ELSE
        NAND2 = 1
       ENDIF
       RETURN
       END
C
       INTEGER FUNCTION NAND3(I1,I2,I3)
       NAND3 = I1 * I2 * I3
       IF (NAND3 .EQ. 1) THEN
        NAND3 = 0
       ELSE
        NAND3 = 1
       ENDIF
```

```
      RETURN
      END
C
      INTEGER FUNCTION INV(I1)
      INV = I1
      IF (INV .EQ. 1) THEN
       INV = 0
      ELSE
       INV = 1
      ENDIF
      RETURN
      END
```

APPENDIX A3:

FORTRAN program to calculate testing time.

```
      INTEGER R1,NS1(256,256),NS2(256,256),OUT1(256,256),
     *        OUT2(256,256),INP(8),NFF,NFF1,NINP,NINP1,L1,
     *        RATE1
C
      REAL P0(8),PF,A,B,C1,C22,D0,D1,D2,D22,CHECK1,CHECK2,
     *     ALPHA,G1,G2,H1,H2,OME1,OME2,W,STIME,LOWER,UPPER,
     *     TEMP(6),LPROB,STEP,ERR
C
      EXTERNAL RNSET,RNUNF
      COMMON A,B,C1,C22,D0,D1,D2,D22,NS1,NS2,OUT1,OUT2,PF,
     *       NINP,ALPHA,P0
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     PARAMETERS DESCRIPTION :
C
C        S1,S2 : THE PRESENT STATE OF THE FAULT-FREE AND FAULTY
C                MACHINE
C        R1 : TOTAL NUMBER OF SIMULATION RUN
C        L1 : NUMBER OF INPUT PATTERN AT EACH RUN
C        NS1(I,J),NS2(I,J) : NEXT STATE OF THE TWO MACHINES
C                WITH PRESENT STATE I AND INPUT J
C        OUTP1,OUTP2 : TEMPORARY STORAGE FOR MACHINE OUTPUTS
C        OUT1(I,J),OUT2(I,J) : MACHINE OUTPUTS WITH PRESENT
C                              STATE I AND INPUT J
C        CTTOT(I,J) : COUNT TOTAL OF ALL SIMULATION RUN FROM
C                     STATE I TO J
C        COUNT(R,I,J) : THE COUNT FROM STATE I TO J FOR THE  R th
C                       RUN
C        STATE(I) : TEMPORARY STORAGE FOR PRESENT STATE AND
C                   NEXT STATE :-
C                   1 FOR NEITHER OUTPUT NOR STATE
C                     TRANSITION BEING DIFFERNT,
C                   2 FOR OUTPUT BEING DIFFERNT,
C                   3 FOR ONLY STATE TRANSITION DIFFERNT
C        INP(I) : CONDITION FOR INPUT LINE I
C        P0(I) : PROBABILITY OF INPUT LINE I HAVING LOGIC 0
C        STEP : CHANGING STEP OF P0(I)
C        CTAVE(I,J) : THE COUNT AVERAGE OF ALL SIMULATION RUN
C        PF : THE PROBABILITY OF HAVING STUCK-AT FAULT
C        LD0,LD1,LD2,MU0,MU1,MU2 : PARAMETERS OBTAINED FROM THE
C                                  COUNTS
C        ALPHA : THE CONFIDENCE INTERVAL
C        STIME : THE TIME REQUIRED FOR DETECTION FAULT WITH THE
C                GIVEN CONFIDENCE INTERVAL
C        RATE1 : THE RATE OF GENERATE INPUT PATTERN
C        TEMP(I) : TEMPORARY STORAGE FOR CHECK1 AND P0(1)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  DEFINE PARAMETERS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      READ(3,131) NFF,NINP,R1
      READ(3,132) L1,RATE1
```

```
      READ(3,133) PF,ALPHA,STEP
      WRITE(6,131) NFF,NINP,R1
      WRITE(6,132) L1,RATE1
      WRITE(6,133) PF,ALPHA,STEP
 131  FORMAT(3I2)
 132  FORMAT(2I7)
 133  FORMAT(3F6.3)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  READ STATE TABLES
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 5 I = 1,NINP1
         READ(4,101) (NS1(J,I), OUT1(J,I), J=1,NFF1)
         READ(5,101) (NS2(J,I), OUT2(J,I), J=1,NFF1)
   5  CONTINUE
 101  FORMAT(2I4)
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  SET PARAMETERS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL RNSET(0)
      NFF1 = 2 ** NFF
      NINP1 = 2 ** NINP
      P0(1) = 0.0
      ERR = 0.0
      TEMP(1) = 0.0
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  MAIN LOOP
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 301  CALL PARA (L1,R1,RATE1,CHECK1,CHECK2)
      IF (CHECK1 .GT. ERR) ERR = CHECK1
      IF (ABS(B) .GT. 0.001) THEN
         IF (CHECK1 .GT. 0) CALL CAL (PF,ALPHA,CHECK1,CHECK2)
         IF (TEMP(1) .EQ. 0) GOTO 904
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  FIND ROOT
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
         IF (CHECK1/TEMP(1) .LT. 0) THEN
            ERR = 0.03 * ERR
            IF (ERR .LT. 0.001) ERR = 0.001
            TEMP(2) = CHECK1
            P0(1) = LOWER + TEMP(1)*STEP/(TEMP(1)-TEMP(2))
 302        CALL PARA (L1,R1,,RATE1,CHECK1,CHECK2)
            IF (ABS(B).LT.0.0001) THEN
               P0(1) = LPROB + STEP
               GOTO 301
            ENDIF
            IF (CHECK1.GT.0 .AND. CHECK1.LT.ERR) THEN
```

```
               CALL CAL (PF,ALPHA,CHECK1,CHECK2)
               P0(1) = 1.0
               GOTO 904
            ENDIF
            IF (CHECK1/TEMP(1).LT.0) THEN
               UPPER = P0(1)
               TEMP(2) = CHECK1
            ELSE
               LOWER = P0(1)
               TEMP(1) = CHECK1
            ENDIF
            P0(1) = LOWER + TEMP(1)*(UPPER-LOWER)/(TEMP(1)-
     *            TEMP(2))
            IF ((CHECK1.GT.0) .AND. ((P0(1)-LOWER).LT.0.001)) THEN
               CALL CAL (PF,ALPHA,CHECK1,CHECK2)
               P0(1) = 1.0
               GOTO 904
            ENDIF
            GOTO 302
         ENDIF
      ENDIF
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  CHANGING STEP FOR P0(1)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
 904  TEMP(1) = CHECK1
      LOWER = P0(1)
      P0(1) = P0(1) + STEP
      LPROB = P0(1)
      UPPER = P0(1)
      IF (P0(1) .LT. 1.0) GOTO 301
      STOP
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  SUBROUTINE TO OBTAIN MARKOV MODEL PARAMETERS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PARA (L1,R1,RATE1,CHECK1,CHECK2)
      INTEGER S,S1,S2,R1,NS1(256,256),NS2(256,256),OUTP1,
     *        OUTP2,OUT1(256,256),OUT2(256,256),CTTOT(3,3),
     *        COUNT(10,3,3),STATE(2),INP(8),L1
C
      REAL P0(8),CTAVE(3,3),PF,LD0,LD1,LD2,MU0,MU1,MU2,A,B,C1,
     *     C22,D0,D1,D2,D22,CHECK1,CHECK2,ALPHA,RATE
C
      EXTERNAL RNSET,RNUNF
      COMMON A,B,C1,C22,D0,D1,D2,D22,NS1,NS2,OUT1,OUT2,PF,
     *       NINP,ALPHA,P0
      RATE = L1 / RATE1
      WRITE(6,116) RATE1
 116  FORMAT(' THE RATE OF GENERATING INPUT SEQUENCE = ',
     *       F10.2,' / MS')
```

```
      DO 10 I = 1,3
         DO 11 J = 1,3
            CTTOT(I,J) = 0
11       CONTINUE
10    CONTINUE
      DO 20 R = 1,R1
         DO 30 I = 1,3
            DO 40 J = 1,3
               COUNT(R,I,J) = 0
40          CONTINUE
30       CONTINUE
         STATE(1) = 1
         STATE(2) = 1
         S1 = 1
         S2 = 1
         L = 1
50       IF ( L.GT.L1 ) GOTO 65
            I1 = 1
            DO 60 I = 1,NINP
               IF ( RNUNF().GT.P0(I) ) I1 = I1 + 2**(I-1)
60          CONTINUE
            OUTP1 = OUT1( S1,I1 )
            OUTP2 = OUT2( S2,I1 )
            IF ( OUTP1.NE.OUTP2 ) STATE(2) = 2
            S1 = NS1( S1,I1 )
            S2 = NS2( S2,I1 )
            IF ( STATE(2).NE.2 .AND. S1.NE.S2 )
     *         STATE(2) = 3
            COUNT( R,STATE(1),STATE(2) ) =
     *         COUNT( R,STATE(1),STATE(2) ) + 1
            L = L + 1
            STATE(1) = STATE(2)
            STATE(2) = 1
            GOTO 50
65       CONTINUE
         DO 70 I = 1,3
            DO 71 J = 1,3
               CTTOT(I,J) = CTTOT(I,J) + COUNT(R,I,J)
71          CONTINUE
70       CONTINUE
20    CONTINUE
      DO 80 I = 1,3
         DO 90 J = 1,3
            CTAVE(I,J) = 1.*CTTOT(I,J) / R1
            WRITE(6,104) I-1,J-1,( COUNT(R,I,J), R = 1,R1 ),
     *                   CTAVE(I,J)
90       CONTINUE
80    CONTINUE
      WRITE(6,111)
111   FORMAT(' ')
104   FORMAT(' COUNT ( ',I1,' , ',I1,' ) =',I10,',AVE = ',
     *       F10.2)
      LD0 = CTAVE(1,2) / RATE
      LD1 = CTAVE(2,3) / RATE
```

```
      LD2 = CTAVE(3,1) / RATE
      MU0 = CTAVE(2,1) / RATE
      MU1 = CTAVE(3,2) / RATE
      MU2 = CTAVE(1,3) / RATE
      A = MU0+MU1+MU2+LD0+LD1+LD2
      B = MU0*MU1+MU0*MU2+MU1*MU2+LD0*LD1+LD0*LD2+
     *    LD1*LD2+LD0*MU1+LD1*MU2+LD2*MU0
      C1 = LD0
      C22 = MU1
      D0 = MU0*LD2+MU0*MU1+LD1*LD2
      D1 = LD0*LD2+LD0*MU1+MU1*MU2
      D2 = MU0*MU2+MU2*LD1+LD0*LD1
      D22 = LD0*LD2+LD0*MU1+MU1*MU2
      CHECK1 = B - A**2/4
      IF (B.EQ.0) THEN
         WRITE(6,131) P0(1)
131      FORMAT(' B EQUALS ZERO AT P0(1) = ',F5.2)
         CHECK2 = 9999
      ELSE
         CHECK2 = PF * (D0*(B-D1) + D2*(B-D22)) / B**2
      ENDIF
      WRITE(6,113) P0(1)
      WRITE(6,114) CHECK1,CHECK2
      WRITE(6,111)
113   FORMAT(' THE PROBABILITY OF HAVING INPUT LINE WITH',
     *       ' LOGIC 0 = ', F12.9)
114   FORMAT(' CHECK1 = ',F12.7,5X,'CHECK2 = ',F12.7)
      IF ( CHECK1.LT.0 ) THEN
         WRITE(6,130)
130      FORMAT(' B - A**2/4 LESS THAN ZERO')
      ENDIF
      IF ( ALPHA.LT.CHECK2 ) THEN
         WRITE(6,105) P0(1),ALPHA
105      FORMAT('NO TEST WILL BE GOOD ENOUGH FOR P0(1) =',
     *          F8.5, ' WITH ALPHA = ', F5.2)
      ENDIF
      WRITE(6,111)
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  SUBROUTINE TO CALCULATE TESTING TIME
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CAL (PF,ALPHA,CHECK1,CHECK2)
C
      REAL P0(8),CTAVE(3,3),PF,A,B,C1,
     *     C22,D0,D1,D2,D22,CHECK1,CHECK2,PF,ALPHA
C
      COMMON A,B,C1,C22,D0,D1,D2,D22
      G1 = -D1 / B
      G2 = -D22 / B
      H1 = ( 2*B*C1 - A*D1 ) / ( 2*B*SQRT(CHECK1) )
      H2 = ( 2*B*C22 - A*D22 ) / ( 2*B*SQRT(CHECK1) )
```

```
      OME1 = SQRT( G1**2 + H1**2 )
      OME2 = SQRT( G2**2 + H2**2 )
      W = PF * (D0*OME1 + D2*OME2) / (B*(ALPHA - CHECK2))
      WRITE(6,801) A,B,C1,C22
      WRITE(6,801) D0,D1,D2,D22
      WRITE(6,801) G1,G2,H1,H2
      WRITE(6,801) OME1,OME2,CHECK1,CHECK2
801   FORMAT(4F15.8)
308   STIME = 2.0 * ALOG(W) / A
      FORMAT(6,802)
      WRITE(6,106) STIME
      FORMAT(6,802)
106   FORMAT(' THE TESTING TIME REQUIRED = ',F16.8)
802   FORMAT('')
      RETURN
      END
```

APPENDIX B1:

Solving of steady state probabilities.

$$\frac{\partial P0(t)}{\partial t} = P1(t) * \mu 0 + P2(t) * \lambda 2 - P0(t) * (\mu 2 + \lambda 0)$$

$$\frac{\partial P1(t)}{\partial t} = P0(t) * \lambda 0 + P2(t) * \mu 1 - P1(t) * (\mu 0 + \lambda 1)$$

$$\frac{\partial P2(t)}{\partial t} = P0(t) * \mu 2 + P1(t) * \lambda 1 - P2(t) * (\lambda 2 + \mu 1)$$

with $P0(0) = 1$, $P1(0) = P2(0) = 0$.

Take Laplace Transform, we have

$$S * P0(S) - 1 = P1(S) * \mu 0 + P2(S) * \lambda 2 - P0(S) * (\mu 2 + \lambda 0)$$

$$S * P1(S) = P0(S) * \lambda 0 + P2(S) * \mu 1 - P1(S) * (\mu 0 + \lambda 1)$$

$$S * P2(S) = P0(S) * \mu 2 + P1(S) * \lambda 1 - P2(S) * (\lambda 2 + \mu 1)$$

$$P0(S) * [S + (\mu 2 + \lambda 0)] = P1(S) * \mu 0 + P2(S) * \lambda 2 + 1 \qquad -- (1)$$

$$P1(S) * [S + (\mu 0 + \lambda 1)] = P0(S) * \lambda 0 + P2(S) * \mu 1 \qquad -- (2)$$

$$P2(S) * [S + (\lambda 2 + \mu 1)] = P0(S) * \mu 2 + P1(S) * \lambda 1 \qquad -- (3)$$

Sub. (3) into (1) and (2),

$$P0(S) * [S + (\mu 2 + \lambda 0)] = P1(t) * \mu 0 + \frac{P0(S) * \mu 2 + P1(t) * \lambda 1}{S + (\lambda 2 + \mu 1)} * \lambda 2 + 1$$

$$P1(S) * [S + (\mu 0 + \lambda 1)] = P0(t) * \lambda 0 + \frac{P0(S) * \mu 2 + P1(S) * \lambda 1}{S + (\lambda 2 + \mu 1)} * \mu 1$$

$$P0(S) * \left[S + (\mu2+\lambda0) - \frac{\mu2*\lambda2}{S+(\lambda2+\mu1)} \right] = P1(S) * \left[\mu0 + \frac{\lambda1*\lambda2}{S+(\lambda2+\mu1)} \right] + 1 \quad \text{-- (4)}$$

$$P1(S) * \left[S + (\mu0+\lambda1) - \frac{\lambda1*\mu1}{S+(\lambda2+\mu1)} \right] = P0(S) * \left[\lambda0 + \frac{\mu1*\mu2}{S+(\lambda2+\mu1)} \right] \quad \text{-- (5)}$$

Sub. (5) into (4)

$$P0(S) * \left[S + (\mu2+\lambda0) - \frac{\mu2*\lambda2}{S+\lambda2+\mu1} \right] = P0(S) * \frac{\left[\lambda0 + \frac{-\mu1*\mu2}{S+\lambda2+\mu1} \right] * \left[\mu0 + \frac{\lambda1*\lambda2}{S+\lambda2+\mu1} \right]}{S+\mu0+\lambda1 - \frac{\lambda1*\mu1}{S+\lambda2+\mu1}} + 1$$

$$P0(S) * \left[\frac{(S+\mu2+\lambda0)*(S+\lambda2+\mu1) - \mu2*\lambda2}{S+12+m1} - \frac{[\lambda0*(S+\lambda2+\mu1)+\mu1*\mu2]*[\mu0*(S+\lambda2+\mu1)+\lambda1*\lambda2]}{(S+\lambda2+\mu1)*[(S+\mu0+\lambda1)*(S+\lambda2+\mu1) - \lambda1*\mu1]} \right] = 1$$

$$\frac{\left\{ \left[S2+S*(\lambda0+\lambda2+\mu1+\mu2)+\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2 \right] * \left[S2+S*(\lambda1+\lambda2+\mu0+\mu1)+\mu0*\lambda2+\mu0*\mu1+\lambda1*\lambda2 \right] - \left[\mu0*\lambda0*S2 + S*\left\{ \mu0*(\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2)+\lambda0*(\mu0*\lambda2+\mu0*\mu1+\lambda1*\lambda2) \right\} + (\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2)*(\mu0*\lambda2+\mu0*\mu1+\lambda1*\lambda2) \right] \right\} * P0(S)}{(S+\lambda2+\mu1)*(S2+C0*S+D0)} = 1 \quad \text{-- (6)}$$

where $C0 = \mu0+\mu1+\lambda1+\lambda2$

$D0 = \mu0*\lambda2 + \mu0*\mu1 + \lambda1*\lambda2$

And, the ( ) in (6) can be simplified as :

$$
\begin{aligned}
( ) &= S4 + S3*(A+\mu1+\lambda2) + S2*(\mu0*\lambda2+\mu0*\mu1+\lambda1*\lambda2+\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2+\lambda0*\mu1+\lambda0*\lambda1+\lambda0*\lambda2+\lambda2*\mu0+\lambda2*\mu1 \\
&\quad +\lambda1*\lambda2+\lambda22+\mu0*\mu1+\mu12+\mu1*\lambda1+\mu1*\lambda2+\mu0*\mu2+\mu1*\mu2+\mu2*\lambda1+\mu2*\lambda2) \\
&\quad + S*[(\mu1+\lambda2)*(\mu0*\lambda2+\mu0*\mu1+\lambda1*\lambda2+\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2) \\
&\quad +\mu2*\mu0*\lambda2+\mu0*\mu1*\mu2+\mu2*\lambda1*\lambda2+\lambda0*\lambda1*\lambda2+\lambda0*\lambda1*\mu1+\mu1*\mu2*\lambda1] \\
&= S4 + S3*(A+\mu1+\lambda2) + S2*[B+A*(\mu1+\lambda2)] + S*[(\mu1+\lambda2)*(\mu0*\lambda2+\mu0*\mu1+\lambda1*\lambda2+\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2) \\
&\quad +(\mu1+\lambda2)*(\mu0*\mu2+\mu2*\lambda1+\lambda0*\lambda1)] \\
&= S4 + S3*(A+\mu1+\lambda2) + S2*[B+A*(\mu1+\lambda2)] + S*B*(\mu1+\lambda2)
\end{aligned}
$$

where $A = \mu0+\mu1+\mu2+\lambda0+\lambda1+\lambda2$

$B = \mu0*\mu1+\mu1*\mu2+\mu0*\mu2+\lambda0*\lambda1+\lambda1*\lambda2+\lambda0*\lambda2+\lambda0*\mu1+\lambda2*\mu0+\mu2*\lambda1$

Then, (6) becomes

$$
P0(S) * \frac{S * (S+\lambda2+\mu1) * (S2+A*S+B)}{(S+\lambda2+\mu1) * (S2+C0*S+D0)} = 1
$$

$$
P0(S) = \frac{S2 + C0*S + D0}{S * (S2+A*S+B)} \quad -- (7)
$$

$$
= \frac{S2 + C0*S + D0}{S * [(S + A/2)2 + (B - A2/4)]}
$$

$$
= K1/S + \frac{K2 * (S+A/2)}{(S + A/2)2 + (B - A2/4)} + \frac{K3 * SQRT(B - A^2/4)}{(S + A/2)2 + (B - A2/4)} \quad -- (8)
$$

To solve K1, K2, K3, we have

$$S2 + C0*S + D0 = K1*(S2+A*S+B) + K2*S*(S+A/2) + K3*S*SQRT(B - A^2/4)$$

$$= S2*(K1+K2) + S*(K1*A + K2*A/2 + K3*SQRT(B - A^2/4)) + K1*B$$

Then, we have

$$K1 = D0 / B$$

$$K2 = (B - D0) / B$$

$$K3 = \frac{2*B*C0 - A*B - A*D0}{2*B*SQRT(B - A^2/4)}$$

Sub. into (8),

$$P0(S) = D0/(B*S) + \frac{B-D0}{B} * \frac{S + A/2}{(S+A/2)2 + (B - A2/4)} + \frac{2*B*C0 - A*B - A*D0}{2*B*SQRT(B - A^2/4)} * \frac{SQRT(B - A^2/4)}{(S+A/2)2 + (B - A^2/4)}$$

Take Laplace Inverse Transform,

$$P0(t) = (D0/B)*U(t) + e^{-(A/2)*t}*[\frac{B-D0}{B} * COS(SQRT(B-A^2/4)*t) + \frac{2*B*C0-A*B-A*D0}{2*B*SQRT(B-A2/4)} * SIN(SQRT(B-A^2/4)*t)$$

To find P1(S), sub (7) into (5), we have

$$P1(S) * \frac{(S+\mu0+\lambda1)*(S+\lambda2+\mu1)-\lambda1*\mu1}{S+\lambda2+\mu1} = \frac{S2+C0*S+D0}{S*(S2+A*S+B)} * \frac{\lambda0*(S+\lambda2+\mu1)+\mu1*\mu2}{S+\lambda2+\mu1}$$

$$P1(S) = \frac{(S2+C0*S+D0) * (\lambda0*S+\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2)}{S * (S2+A*S+B) * (S2+C0*S+D0)}$$

$$= \frac{C1*S + D1}{S * (S2+A*S+B)} \quad -- (9)$$

where $C1 = \lambda 0$

$D1 = \lambda 0*\lambda 2 + \lambda 0*\mu 1 + \mu 1*\mu 2$

$$P1(S) = K4/S + \frac{K5*(S+A/2)}{(S+A/2)2 + (B-A^2/4)} + \frac{K6*SQRT(B-A^2/4)}{(S+A/2)2 + (B-A^2/4)} \quad -- (10)$$

To solve K4, K5, and K6, we use the similar approach as before,

$$C1*S + D1 = S2*(K4+K5) + S*(K4*A+K5*A/2+K6*SQRT(B - A^2/4)) + K4*B$$

Then, we have

$K4 = D1 / B$

$K5 = -D1 / B$

$$K6 = \frac{2 * B * C1 - A * D1}{2 * B * SQRT(B - A^2/4)}$$

Sub. into (10), we have

$$P1(S) = D1/(B*S) + \frac{-D1}{B} * \frac{(S+A/2)}{(S+A/2)2 + (B-A^2/4)} * \frac{2*B*C1-A*D1}{2*B*SQRT(B-A^2/4)} * \frac{SQRT(B-A^2/4)}{(S+A/2)2 + (B-A^2/4)}$$

Take Laplace Inverse Transform,

$$P1(t) = (D1/B)*U(t) + e^{-(A/2)*t}*[\frac{-D1}{B} * COS(SQRT(B-A^2/4)*t) + \frac{2*B*C1 - A*D1}{2*B*SQRT(B-A^2/4)} * SIN(SQRT(B-A^2/4)*t)]$$

To find P2(S), sub. (7) and (9) into (3),

$$P2(S) * (S+\lambda 2+\mu 1) = \frac{\mu 2*(S2+C0*S+D0) + \lambda 1*(C1*S+D1)}{S * (S2+A*S+B)}$$

$$P2(S) = \frac{\mu 2*(S2+S*(\mu 0+\mu 1+\lambda 1+\lambda 2)+\mu 0*\lambda 2+\mu 0*\mu 1+\lambda 1*\lambda 2) + \lambda 1*(\lambda 0*S+\lambda 0*\lambda 2+\lambda 0*\mu 1+\mu 1*\mu 2)}{S * (S2 + A*S + B) * (S +\lambda 2+\mu 1)}$$

$$= \frac{\mu 2*S2 + S*(\mu 0*\mu 2+\mu 2*\lambda 1+\lambda 0*\lambda 1+(\mu 1+\lambda 2)*\mu 2) + (\mu 1+\lambda 2)*(\mu 0*\mu 2+\lambda 1*\mu 2+\lambda 0*\lambda 1)}{S * (S2 + A*S + B) * (S +\lambda 2+\mu 1)}$$

$$= \frac{[\mu 2*S + (\mu 0*\mu 2+\mu 2*\lambda 1+\lambda 0*\lambda 1)] * (S+\mu 1*\lambda 2)}{S * (S2 + A*S + B) * (S +\lambda 2+\mu 1)}$$

$$= \frac{C2 * S + D2}{S * (S2 + A*S + B)}$$

where $C2 = \mu 2$

$D2 = \mu 0*\mu 2 + \mu 2*\lambda 1 + \lambda 0*\lambda 1$

$$P2(S) = K7/S + \frac{K8 * (S+A/2)}{(S+A/2)2 + (B-A^2/4)} + \frac{K9*SQRT(B-A^2/4)}{(S+A/2)2 + (B-A^2/4)} \qquad -- (11)$$

Similarly, we have

$$C2*S + D2 = S2*(K7+K8) + S*(K7*A+K8*A/2+K9*SQRT(B-A^2/4)) + K7*B$$

Then, we have

$$K7 = D2 / B$$

$$K8 = -D2 / B$$

$$K9 = \frac{2*B*C2 - A*D2}{2*B*SQRT(B-A2/4)}$$

Sub. into (11),

$$P2(S) = D2/(B*S) + \frac{-D2}{B} * \frac{(S+A/2)}{(S+A/2)2 + (B-A^2/4)} + \frac{2*B*C2-A*D2}{2*B*SQRT(B-A^2/4)} * \frac{SQRT(B-A^2/4)}{(S+A/2)2 + (B-A^2/4)}$$

Take Laplace Inverse Transform,

$$P2(t) = (D2/B)*U(t) + e^{-(A/2)*t}*[\frac{-D2}{B} * COS(SQRT(B-A^2/4)*t) + \frac{2*B*C2-A*D2}{2*B*SQRT(B-A^2/4)} * SIN(SQRT(B-A^2/4)*t)]$$

As $B - A2/4 = 0$ or $A = 2 * SQRT(B)$ and,

$$\lim \frac{SIN(x*t)}{x} = t \quad \text{as x approaches 0,}$$

$$P0(t) = (D0/B)*U(t) + e^{-(A/2)*t} * [(B-D0)/B + \frac{2*B*C0 - A*B - A*D0}{2 * B} * t]$$

$$P1(t) = (D1/B)*U(t) + e^{-(A/2)*t} * [-D1/B + \frac{2*B*C1 - A*D1}{2 * B} * t]$$

$$P2(t) = (D2/B)*U(t) + e^{-(A/2)*t} * \left[-D2/B + \frac{2*B*C2 - AD2}{2 * B} * t \right]$$

As t approaches ∞, $e^{-(A/2)*t}$ approaches 0, and $t * e^{-(A/2)*t}$ approaches 0.

So, P0(t), P1(t), P2(t) will converge to D0/B, D1/B and D2/B respectively.

APPENDIX B2:

Solving of transition probabilities.

$$\frac{\partial P10(t)}{\partial t} = P11(t) * \mu 0 + P12(t) * \lambda 2 - P10(t) * (\mu 2 + \lambda 0)$$

$$\frac{\partial P11(t)}{\partial t} = P10(t) * \lambda 0 + P12(t) * \mu 1 - P11(t) * (\mu 0 + \lambda 1)$$

$$\frac{\partial P12(t)}{\partial t} = P10(t) * \mu 2 + P11(t) * \lambda 1 - P12(t) * (\lambda 2 + \mu 1)$$

with $P11(0) = 1$, $P10(0) = P12(0) = 0$.

Take Laplace Transform, we have

$$S * P10(S) = P11(S) * \mu 0 + P12(S) * \lambda 2 - P10(S) * (\mu 2 + \lambda 0)$$

$$S * P11(S) - 1 = P10(S) * \lambda 0 + P12(S) * \mu 1 - P11(S) * (\mu 0 + \lambda 1)$$

$$S * P12(S) = P10(S) * \mu 2 + P11(S) * \lambda 1 - P12(S) * (\lambda 2 + \mu 1)$$

$$P10(S) * [S + (\mu 2 + \lambda 0)] = P11(S) * \mu 0 + P12(S) * \lambda 2 \qquad -- (1)$$

$$P11(S) * [S + (\mu 0 + \lambda 1)] = P10(S) * \lambda 0 + P12(S) * \mu 1 + 1 \qquad -- (2)$$

$$P12(S) * [S + (\lambda 2 + \mu 1)] = P10(S) * \mu 2 + P11(S) * \lambda 1 \qquad -- (3)$$

Sub. (3) into (1) and (2),

$$P10(S) * [S + (\mu 2 + \lambda 0)] = P11(t) * \mu 0 + \frac{P10(S) * \mu 2 + P11(t) * \lambda 1}{S + (\lambda 2 + \mu 1)} * \lambda 2$$

$$P11(S) * [S + (\mu 0 + \lambda 1)] = P10(t) * \lambda 0 + \frac{P10(S) * \mu 2 + P11(S) * \lambda 1}{S + (\lambda 2 + \mu 1)} * \mu 1 + 1$$

$$P10(S) * \left[S + (\mu 2 + \lambda 0) - \frac{\mu 2 * \lambda 2}{S + (\lambda 2 + \mu 1)} \right] = P11(S) * \left[\mu 0 + \frac{\lambda 1 * \lambda 2}{S + (\lambda 2 + \mu 1)} \right] \quad -- (4)$$

$$P11(S) * \left[S + (\mu 0 + \lambda 1) - \frac{\lambda 1 * \mu 1}{S + (\lambda 2 + \mu 1)} \right] = P10(S) * \left[\lambda 0 + \frac{\mu 1 * \mu 2}{S + (\lambda 2 + \mu 1)} \right] + 1 \quad -- (5)$$

Sub. (4) into (5)

$$P11(S) * \left[S + (\mu 2 + \lambda 0) - \frac{\mu 1 * \lambda 1}{S + \lambda 2 + \mu 1} \right] = P11(S) * \frac{\left[\mu 0 + \frac{\lambda 1 * \lambda 2}{S + \lambda 2 + \mu 1} \right] * \left[\lambda 0 + \frac{\mu 1 * \mu 2}{S + \lambda 2 + \mu 1} \right]}{S + \mu 2 + \lambda 0 - \frac{\lambda 1 * \mu 1}{S + \lambda 2 + \mu 1}} + 1$$

$$P11(S) * \left[\frac{(S+\mu 0+\lambda 1) * (S+\lambda 2+\mu 1) - \mu 1*\lambda 1}{S + \lambda 2 + \mu 1} - \frac{[\lambda 0*(S+\lambda 2+\mu 1)+\mu 1*\mu 2] * [\mu 0*(S+\lambda 2+\mu 1)+\lambda 1*\lambda 2]}{(S+\lambda 2+\mu 1) * [(S+\mu 2+\lambda 0) * (S+\lambda 2+\mu 1) - \lambda 2*\mu 2]} \right] = 1$$

$$\frac{\left([S2+S*(\lambda 1+\lambda 2+\mu 0+\mu 1)+\lambda 0*\lambda 2+\lambda 2*\mu 0+\mu 0*\mu 1] * [S2+S*(\lambda 0+\lambda 2+\mu 1+\mu 2)+\mu 1*\lambda 0+\mu 1*\mu 2+\lambda 0*\lambda 2] - [\mu 0*\lambda 0*S2 + S * [\mu 0*(\lambda 0*\lambda 2+\lambda 0*\mu 1+\mu 1*\mu 2)+\lambda 0*(\mu 0*\lambda 2+\mu 0*\mu 1+\lambda 1*\lambda 2)] + (\lambda 0*\lambda 2+\lambda 0*\mu 1+\mu 1*\mu 2) * (\mu 0*\lambda 2+\mu 0*\mu 1+\lambda 1*\lambda 2)]) * P11(S) \right)}{(S + \lambda 2 + \mu 1) * (S2 + C10 * S + D10)} = 1 \quad -- (6)$$

where $C10 = \mu 1 + \mu 2 + \lambda 0 + \lambda 2$

$D10 = \mu 1*\lambda 0 + \mu 1*\mu 2 + \lambda 0*\lambda 2$

Similarly, the expression in (6) can be simplified as :

$$P11(S) * \frac{S4 + S3*(A+\mu1+\lambda2) + S2*(B+(\mu1+\lambda2)*A) + S*B*(\mu1+\lambda2)}{(S+\lambda2+\mu1) * (S2+C10*S+D10)} = 1$$

$$P11(S) = \frac{(S+\lambda2+\mu1) * (S2+C10*S+D10)}{S * (S+\lambda2+\mu1) * (S2+A*S+B)}$$

where $A = \mu0+\mu1+\mu2+\lambda0+\lambda1+\lambda2$

$B = \mu0*\mu1+\mu1*\mu2+\mu0*\mu2+\lambda0*\lambda1+\lambda1*\lambda2+\lambda0*\lambda2+\lambda0*\mu1+\lambda2*\mu0+\mu2*\lambda1$

Then,

$$P11(S) = \frac{S2 + C10*S + D10}{S * (S2+A*S+B)} \quad \text{-- (7)}$$

To find P10(S), sub (7) into (5), we have

$$P10(S) * \frac{(S+\mu2+\lambda0)*(S+\lambda2+\mu1)-\lambda2*\mu2}{S+\lambda2+\mu1} = \frac{S2+C10*S+D10}{S*(S2+A*S+B)} * \frac{\mu0*(S+\lambda2+\mu1)+\lambda1*\lambda2}{S+\lambda2+\mu1}$$

$$= \frac{C12*S + D12}{S * (S2+A*S+B)} \quad \text{-- (8)}$$

where $C12 = \mu0$

$D12 = \lambda1*\lambda2 + \lambda2*\mu0 + \mu0*\mu1$

To find P12(S), sub. (7) and (8) into (3),

$$P12(S) * (S+\lambda 2+\mu 1) = \frac{\lambda 1 * (S2+C10*S+D10) + \mu 2 * (C12*S+D12)}{S * (S2+A*S+B)}$$

$$P12(S) = \frac{\lambda 1*(S2+S*(\mu 1+\mu 2+\lambda 0+\lambda 2)+\mu 1*\lambda 0+\mu 1*\mu 2+\lambda 0*\lambda 2) + \mu 2*(\mu 0*S+\lambda 1*\lambda 2+\lambda 2*\mu 0+\mu 0*\mu 1)}{S * (S2 + A*S + B) * (S +\lambda 2+\mu 1)}$$

$$= \frac{[\lambda 1*S + (\mu 0*\mu 2+\mu 2*\lambda 1+\lambda 0*\lambda 1)] * (S+\mu 1*\lambda 2)}{S * (S2 + A*S + B) * (S +\lambda 2+\mu 1)}$$

$$= \frac{C11 * S + D11}{S * (S2 + A*S + B)}$$

where $C11 = \lambda 1$

$D11 = \mu 0*\mu 2 + \mu 2*\lambda 1 + \lambda 0*\lambda 1$

$$\frac{\partial P20(t)}{\partial t} = P21(t) * \mu 0 + P22(t) * \lambda 2 - P20(t) * (\mu 2 + \lambda 0)$$

$$\frac{\partial P21(t)}{\partial t} = P20(t) * \lambda 0 + P22(t) * \mu 1 - P21(t) * (\mu 0 + \lambda 1)$$

$$\frac{\partial P22(t)}{\partial t} = P20(t) * \mu 2 + P21(t) * \lambda 1 - P22(t) * (\lambda 2 + \mu 1)$$

with $P22(0) = 1$, $P20(0) = P21(0) = 0$.

Take Laplace Transform, we have

$$S * P20(S) = P21(S) * \mu 0 + P22(S) * \lambda 2 - P20(S) * (\mu 2 + \lambda 0)$$

$$S * P21(S) = P20(S) * \lambda 0 + P22(S) * \mu 1 - P21(S) * (\mu 0 + \lambda 1)$$

$$S * P22(S) - 1 = P20(S) * \mu 2 + P21(S) * \lambda 1 - P22(S) * (\lambda 2 + \mu 1)$$

$$P20(S) * [S + (\mu 2 + \lambda 0)] = P21(S) * \mu 0 + P22(S) * \lambda 2 \qquad -- (1)$$

$$P21(S) * [S + (\mu 0 + \lambda 1)] = P20(S) * \lambda 0 + P22(S) * \mu 1 \qquad -- (2)$$

$$P22(S) * [S + (\lambda 2 + \mu 1)] = P20(S) * \mu 2 + P21(S) * \lambda 1 + 1 \qquad -- (3)$$

Sub. (1) into (2) and (3),

$$P21(S) * [S + (\mu 0 + \lambda 1)] = P22(t) * \mu 1 + \frac{P21(S) * \mu 0 + P22(t) * \lambda 2}{S + (\lambda 0 + \mu 2)} * \lambda 0$$

$$P22(S) * [S + (\mu 1 + \lambda 2)] = P21(t) * \lambda 1 + \frac{P21(S) * \mu 0 + P22(S) * \lambda 2}{S + (\lambda 0 + \mu 2)} * \mu 2 + 1$$

$$P21(S) * \left[S + (\mu0+\lambda1) - \frac{\mu0 * \lambda0}{S + (\lambda0+\mu2)} \right] = P22(S) * \left[\mu1 + \frac{\lambda0 * \lambda2}{S + (\lambda0+\mu2)} \right] \quad -- (4)$$

$$P22(S) * \left[S + (\mu1+\lambda2) - \frac{\lambda2 * \mu2}{S + (\lambda0+\mu2)} \right] = P21(S) * \left[\lambda1 + \frac{\mu0 * \mu2}{S + (\lambda0+\mu2)} \right] + 1 \quad -- (5)$$

Sub. (4) into (5)

$$P22(S) * \left[S + (\mu1+\lambda2) - \frac{\mu2 * \lambda2}{S+\lambda0+\mu2} \right] = P22(S) * \frac{\left(\mu1 + \frac{\lambda0 * \lambda2}{S+\lambda0+\mu2} \right) * \left(\lambda1 + \frac{\mu0 * \mu2}{S+\lambda0+\mu2} \right)}{S+\mu0+\lambda1 - \frac{\lambda0 * \mu0}{S+\lambda0+\mu2}} + 1$$

$$P22(S) * \left[\frac{(S+\mu1+\lambda2) * (S+\lambda0+\mu2) - \mu1*\lambda1}{S+\lambda0+\mu2} - \frac{[\lambda1*(S+\lambda0+\mu2)+\mu0*\mu2] * [\mu1*(S+\lambda0+\mu2)+\lambda0*\lambda2]}{(S+\lambda0+\mu2) * [(S+\mu0+\lambda1) * (S+\lambda0+\mu2) - \lambda0*\mu0]} \right] = 1$$

$$\frac{\left([S2+S*(\lambda0+\lambda2+\mu1+\mu2)+\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2] * [S2+S*(\lambda0+\lambda1+\mu0+\mu2)+\mu2*\lambda1+\mu0*\mu2+\lambda0*\lambda1] - [\mu1*\lambda1*S2 + S * [\mu1*(\lambda0*\lambda1+\lambda1*\mu2+\mu0*\mu2)+\lambda1*(\mu1*\lambda0+\mu1*\mu2+\lambda0*\lambda2)] + (\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2) * (\mu2*\lambda1+\mu0*\mu2+\lambda0*\lambda1)]) * P22(S) \right)}{(S+\lambda0+\mu2) * (S2 + C20 * S + D20)} + 1 \quad -- (6)$$

where $C20 = \mu0 + \mu2 + \lambda0 + \lambda1$

$D20 = \mu2*\lambda1 + \mu0*\mu2 + \lambda0*\lambda1$

Similarly, the expression in (6) can be simplified as :

$$P22(S) * \frac{S4 + S3*(A+\mu2+\lambda0) + S2*(B+(\mu2+\lambda0)*A) + S*B*(\mu2+\lambda0)}{(S+\lambda0+\mu2) * (S2+C20*S+D20)} = 1$$

$$P22(S) = \frac{(S+\lambda0+\mu2) * (S2+C20*S+D20)}{S * (S+\lambda0+\mu2) * (S2+A*S+B)}$$

where $A = \mu0+\mu1+\mu2+\lambda0+\lambda1+\lambda2$

$B = \mu0*\mu1+\mu1*\mu2+\mu0*\mu2+\lambda0*\lambda1+\lambda1*\lambda2+\lambda0*\lambda2+\lambda0*\mu1+\lambda2*\mu0+\mu2*\lambda1$

Then,

$$P22(S) = \frac{S2 + C20*S + D20}{S * (S2+A*S+B)} \quad -- (7)$$

To find P21(S), sub (7) into (4), we have

$$P21(S) * \frac{(S+\mu0+\lambda1)*(S+\lambda0+\mu2)-\lambda0*\mu0}{S+\lambda0+\mu2} = \frac{S2+C20*S+D20}{S*(S2+A*S+B)} * \frac{\mu1*(S+\lambda0+\mu2)+\lambda0*\lambda2}{S+\lambda0+\mu2}$$

$$= \frac{C22*S + D22}{S * (S2+A*S+B)} \quad -- (8)$$

where $C22 = \mu1$

$D22 = \lambda0*\lambda2 + \lambda0*\mu1 + \mu1*\mu2$

To find P20(S), sub. (7) and (8) into (1),

$$P20(S) * (S+\lambda0+\mu2) = \frac{\lambda2*(S2+C20*S+D20) + \mu0*(C22*S+D22)}{S * (S2+A*S+B)}$$

$$P20(S) = \frac{\lambda2*(S2+S*(\mu0+\mu2+\lambda0+\lambda1)+\mu2*\lambda1+\mu0*\mu2+\lambda0*\lambda1) + \mu0*(\mu1*S+\lambda0*\lambda2+\lambda0*\mu1+\mu1*\mu2)}{S * (S2 + A*S + B) * (S +\lambda0+\mu2)}$$

$$= \frac{[\lambda2*S + (\mu0*\mu1+\mu0*\lambda2+\lambda1*\lambda2)] * (S+\mu0*\lambda0)}{S * (S2 + A*S + B) * (S +\lambda0+\mu2)}$$

$$= \frac{C21 * S + D21}{S * (S2 + A*S + B)}$$

where $C21 = \lambda2$

$D21 = \mu0*\mu1 + \mu0*\lambda2 + \lambda1*\lambda2$

Papers by the Author on which Portions of the Present Thesis is Based

[1] S. R. Das, W. B. Jone and K. L. Wong, "Probabilisitc Modeling and Fault Analysis in Sequential Logic using Computer Simulation", Computer Aided Design, Modelling and Simulation, pp. 87-93, 1986.

[2] S. R. Das, W. B. Jone and K. L. Wong, "Probabilisitc Modeling and Fault Analysis in Sequential Logic using Computer Simulation", IEEE Transactions on Systems, Man and Cybernetics, pp. 490-498, March/April 1990.