Development of Advance ATPG Using Probabilistic Approach

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Tarak Patel
(14MECV22)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2015

Development of Advance ATPG Using Probabilistic Approach

Major Project Report

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology
in
Electronics & Communication Engineering
(VLSI Design)

By

Tarak Patel
(14MECV22)

Under the guidance of

Dr. Usha Sandeep Mehta

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2015

NIRMA
UNIVERSITY
INSTITUTE OF TECHNOLOGY
NAAC ACCREDITED 'A' GRADE

Certificate

This is to certify that the Major Project (Phase- I) entitled **"Development of Advance ATPG Using Probabilistic Approach "** submitted by **Tarak Patel (14MECV22)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in VLSI Design, Nirma University, Ahmedabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.The results embodied in this major project, to the best of our knowledge,haven't been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Internal Guide **Program Co-ordinator**

Dr. Usha S. Mehta
(Professor,EC)

Dr. N. M. Devashrayee
(Professor,EC)

Director

Dr. P. N. Tekwani
(Head of EE Dept.)
(Director, IT-NU)

Declaration

This is to certify that

1. The thesis comprises my original work towards the degree of Master of Technology in VLSI Design at Nirma University and has not been submitted elsewhere for a degree.

2. Due acknowledgment has been made in the text to all other material used.

- Tarak Patel
14MECV22

Acknowledgements

I express my deep and sincere gratitude towards **Dr. P. N. Tekwani**, Head of Department (EE), Institute of Technology, Nirma University, for providing the facilities required for completing this project successfully.

I would like to express my sincere thanks to **Dr.N.M.Devashrayee**, PG Coordinator (VLSI Design),Institute of Technology, Nirma University, for his kind, ordinal and valuable guidance and support at every moment of the project.

I owe my most sincere gratitude to my guide, **Prof.Usha Sandeep** Mehta, Professor, PG VLSI Design, Institute of Technology, Nirma University. Her constant support, guidance and encouragement have provided me a strong basis for this project. Her dedication to work always inspired me to do task as a challenge. Her practical approach, needful help encouraged me to do better work.

- Tarak Patel
14MECV22

Abstract

To Test the functioning of IC in today's era, Automatic Test Pattern Generator (ATPG) is becoming very important as circuit become more & more complicated. Fault coverage should be high to detect maximum faults in design, while maintaining design overhead with in certain limit. Design reliability should be high with minimum cost and time. ATPG is beneficial in many ways. First, it can reduce the test time for a large circuit. In addition, ATPG can provide at speed testing. Stuck at fault model is considered throughout this project. Considering stuck at fault, the fault list becomes lengthy as circuit size increases. Fault Equivalence method is used to reduce fault in the circuit. Testability measures like controllability and Observability are considered to find how complicated is to test internal nodes. Controllability use to guides the test generation algorithms while setting a value of net in line justification problem. To generate test vector for any fault involves "line activation", "Fault Propagation", and "line justification" step.The EDA Tool is developed using C Programming language for 3 fan-in. Developed ATPG Tool is generic and can be used for any combinational logic Circuit. An efficient and simple technique used for test pattern generation is necessary with the intention of reducing number of faults. This algorithm must be simple. In this project combinational ATPG is developed based on testability measures, And Probability is used for finding fault which is hardest to find. So using Probability Undetectable Fault are found early in Test Pattern Generation so we do not have to scan all Fault list.

Contents

List of Figures

List of Tables

Abbreviations

PI	Primary Input
PO	Primary Output
SOC	System On Chip
DUT	Design Under Test
SSF	Single Stuck at Fault
FAN	Fanout Oriented ATPG Algorithm
PODEM	Path Oriented Decision Making
HDL	Hardware Description Language
ATPG	Automatic Test Pattern Generator
ISCAS	International Symposium on Circuit and System

Chapter 1

Introduction

1.1 Motivation

As we move move toward very large-scale integration (VLSI-ULSI) system design complexity and integration density of integrated circuit increased to great extent. To test this type of circuit more effective and better testing mechanism are required to ensure desired functionality from manufactured chips. As technology changes circuit increases but number of input/output pins not increased with same rate. So Pin to Gate ratio continually decreasing. Beside this chip testing is time consuming task. It take up to one-half of total product design time.

Testing techniques have to be improved, in order to achieve this higher quality at lower cost. So automatic test generation techniques need continuous improvement to handle large VLSI chips. Hence Automatic test pattern generation for digital circuits has been extensively investigated. At initial stage ATPG is form using basic concepts of fault reduced set using equivalence method, testability measures, line justification and propagation are developed using C programming. The commercial ATPG tools are available in the market like Tessent FastScan by Mentor Graphics, TetraMAx ATPG by Synopsys and open source ATPG by Atlanta. The cost of these ATPGs is high and they are complex. My aim to develop an open source advanced ATPG which is freely available and it should be less complex.

1.2 Need of ATPG

The objective to test circuits is to separate faulty ones from good ones. ATPG (Automatic Test Pattern Generator) is an EDA tool which is used to find an input or test pattern, That is when applied to a logic circuit, enables automatic test equipment to distinguish between the correct circuit behaviour and the faulty circuit behaviour caused by fault. Complete functional test is impractical, Designer generated functional patterns typically provide only 70-75% coverage. ATPG supplements to get coverage to $>$ 98%. With the help of Scan Insertion we can test sequential logic with ATPG.

1.3 Test Pattern Generation

Circuit input are called primary inputs. Test patterns can only be applied to primary input. and output are called primary outputs. These are the only places where we can observe the effects of the faults when input are applied to primary input. Thus, test pattern generation is the task of finding a input patterns that will fully test the circuit. These patterns/test set, must cause all faulty circuits to exhibit different output from good circuits. A failure can be found when at least one output is different. The test set must be such that it can be applied economically to all circuits produced. Testing that uses all input test pattern combination will certainly find all fault from circuit. but it is too expensive for VLSI circuits with large no of inputs.

For large circuits, exhaustive testing is time consuming and cumbersome task, so we made some assumption to make it simple.first we assume only stuck at fault present in circuit. In this type of fault particular net permanently assume either it is connected to V_{dd} or connected to ground. If net value is 0 than it is stuck at 0 (s_a_0) and if net value is 1 than it is stuck at 1 (s_a_1) fault. Second we assume that only stuck at fault present in circuit no other fault is present. It might looks like that single stuck at fault is not sufficient to test circuit. But it cover many other fault also, If we find large no of stuck at fault in circuit the test set we find from this fault can cover many other fault also. As definition of test pattern generation tool should cover all detectable single stuck at fault. There are some fault present in circuit that are undetectable, also called Redundant fault. Test set generated by ATPG tool consist of binary input must give different value at primary output. To generate this difference at primary output difference must be produced at fault location. This process is called fault activation. Now this fault effect must be produced at output, process of caring out fault from faulty net to output is called fault propagation. While propagating fault we have to set other net value to ensure that fault reach primary output. This process is called line Justification. And in above two process if net assignment is conflicting than it is back-traced and other path is taken for pattern generation. In this Probabilistic approach Probability is used to find fault which is hardest to find. So which ever net Probability is Minimum for logic 0 or logic 1 fault on that net have highest possibility of being undetectable. All fault in circuit is shorted according to probability and this fault list is given to ATPG to generate test pattern. So we can get undetectable fault very early in test pattern generation.

1.4 Organisation of Thesis

This report gives brief introduction of fault equivalence method to reduced number of stuck at faults, testability measures and basic ATPG concept along with their implementation. The advanced algorithm based on FAN algorithm is described in this report. The report mainly divided into nine chapters. Chapter 2 deals with the basics of fault model and types of faults. Standard net-list in text format which acts as an input the program is discussed in chapter three 3 with circuit and format. Chapter 4 includes fault reduction method, fault equivalence along with the implementation, flow chart and results. Chapter 5 includes testability measures like controllability 0, controllability 1 and observability which will helpful to implement ATPG. Chapter 6 deal with probability measure used in ATPG algorithm. Chapter 7 describes type of ATPG algorithm and flow chart of whole algorithm. And difference between different ATPG algorithm. In Chapter 8 implementation of ATPG algorithm and result are shown.

Chapter 2

Fault Modeling

In the designing of a chip, fault is required to be detected before the chip is being kept in market for sale, further it is required for to test the chip at each stage of fabrication and even after mounting it on the board, for which fault simulation is required. This simulation is done with the help of different types of fault modeling and different ways of fault detection, all these aspects are covered in this chapter along detailed explanation of types of fault.

Reliability of digital systems is of growing concern these days as they are increasingly being used in critical applications. Hence testing methods to detect faults in digital systems are assuming growing importance. Here digital systems symbolize various levels of abstraction, which are to be considered to describe the operation of a circuit to test them. These various levels of abstractions are required as, each design requires several different representations or views, which differ in the type of information that they emphasize. In addition the same representation often requires different levels of detail in different phases of the design. The three most common types of representation that are used are behavioral, structural, and physical representations. The levels of abstraction are as given below:

- Logic level
- Register level
- Instruction level
- Processor level
- System level

The lowest level of abstraction is logic level. The information processed at this level is in the terms of binary logic values i.e. 0 and 1. Here itself a distinction can be made in between the types of circuit as sequential and combinational logic circuits.The next level of abstraction is register level where the data in form of words are stored in registers, also the controlling of registers are done by the logic values only. In the next level of abstraction the controlling is also done by a set of logic values called as word and this word are termed as instructions. In this level data are also in the form of words. Now the level where the sequence of instructions is processed called program, are termed as processor level abstraction. Here the program work upon a block of data called as data structure. Now in the highest level of abstraction the system is considered as a set of independent subsystems communicating with each other via a block of words called messages, and this level is the system level.

2.1 Logical Fault Model

Fault models are required to observe the effect of test. It is representation of physical fault on operation of logic circuit. only considered logical functionality of circuit not timing. It provide mathematical model for circuit diagnosis and testing.Fault modeling is a main concept to be considered in designing, fabrication and testing of digital systems. As the modeling is the way of representation of circuits and are required to decide the way of testing a circuit for its proper functionality. The basic types of modeling are: behavioral model, functional model and structural model.

Functional Model: It is a representation of the logic function of system, Which only deal with functionality of input and output not timing relation between them.

Behavioral Model: It consists of the functional model associated with its timing relations. Depending on the level of abstraction, the behavior of circuit can be specified as the mapping of logic values, or data of data words and this transformation occurs over time.

Structural Model: It describes the system as the collection of inter-connected module which consist of components or elements. Connection between lower level components make whole system.The lowest level is termed as 'primitive element'. The function of component is shown by the types of components such as CPU, ALU etc. It always contains the information regarding the function of the components. Also the modeling can be classified as internal and external model, where external model is the model as viewed by the user and internal model is the model consisting of the structures and programs that represent system inside a computer.

Connectivity language is used to describe structural model. Also structural fault models assume that interconnection are affected by fault, Component which are used in lower level module are fault free. Fault affecting interconnection are stuck at open and short. Open fault is from by breaking of connection and short is formed by connection of net which is not intended to be connected.

2.2 Types of Error and Fault

The incorrect operation of a system being observed is termed as an error. The causes of errors in digital system can be due to fabrication errors, design error, fabrication defects and physical failure. Failure modes at logical level are

- Incorrect signal values.
- Design errors can be in the form given below Incomplete or
- Inconsistent specification Incorrect mappings between different
- Levels of design Violations of design rules

Similarly errors occurring at the time of fabrication which can be said to be human errors are

- Wrong components
- Incorrect wiring
- Shorts caused by improper soldering

Now the fabrication errors are not the human errors and they include

- Shorts and opens in manufacturing circuits
- Improper doping profile
- mask alignment error
- poor encapsulation

Physical failure occurs to the component due to various reasons given below: Wear out of internal connectors etc. with time due to corrosion or electron migration. Environmental effects such as temperature, humidity, Vibration. Cosmic radiations effecting high density random access memories. Here fabrication error, fabrication defects and physical failure are collectively termed as physical faults. And a fault is a model that represents change in system signal as effect of failure. These faults can be further classified as

- permanent : always present after their occurrence
- intermittent : existing only during certain period
- transient : one time occurrence caused due to temporary change in some environmental factors

All these faults causes incorrect operation of the circuit represented as logical faults further classified as

- explicit or implicit fault
- structural or functional fault permanent or intermittent fault
- single stuck-at-fault or multiple stuck-at-fault

Model does not represent exact fault in circuit but it helps in detecting defects. For example, stuck at fault does not represent all physical failure present in circuit. Use of stuck at fault in past is because of it is easy to use and cover many other type of faults also. However any fault model can not represent all physical failures.

2.3 Defect, Fault and Error

The physical faults like short between V_{dd} or Ground, open circuit condition is known as defects. The defects are interpret in logical way is known as fault. These stuck at faults are stuck at logical 0 (S_A_0) means that net is permanently stuck to the ground, stuck at logical 1 (S_A_1) means that net is permanently stuck to V_{dd}. The two net can get shorted is known as bridging fault. Different stuck at faults are shown in 2.2. The erroneous result produced at the output is known as error.

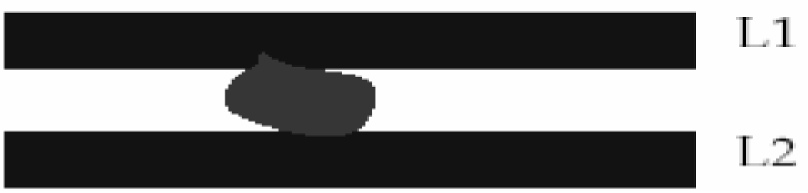


Figure 2.1: Physical Defect

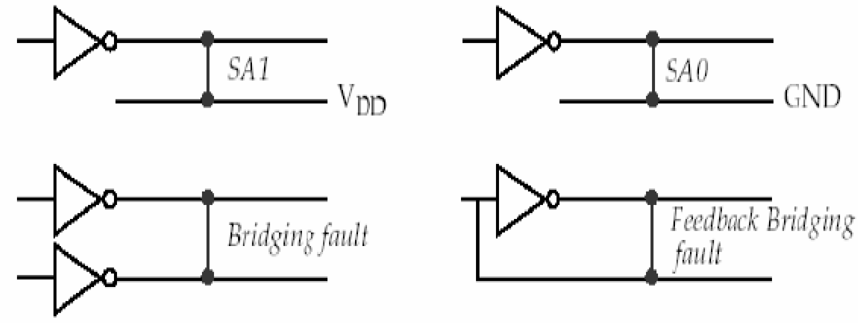


Figure 2.2: Different stuck at fault

2.4 Single Stuck at Fault

Single stuck at fault model is most popular among other fault models available. In this model, any signal lines in a circuit is assumed to be a fix value either logic 0 or Logic 1, regardless of what inputs are supplied to the circuit. A stuck at fault represents a net in the circuit that is connected to logic value 0 or 1. stuck at fault model is considered as classical fault because it is widely used and studied. Though it is not considered as universal fault but it is used due to following attributes. Many different physical fault can be represented. It is independent of technology as considered only single line being stuck at 0 or stuck at 1. Number of single stuck at fault in the circuit are less and also used to model other type of fault. stuck at fault used to represent short or open fault which is consider as directly connected to ground or power line. So, logical fault consists of the signal being stuck at a fixed logic value V (V ϵ 0, 1), and denoted by S_A_V. Due to short between any line cause change in logic function. So short between two line is referred as bridging fault. If signal line is unidirectional and it has only one fan-out make input unconnected due to stuck at open fault. Assume a constant logic value and hence appear as a stuck fault in fig 2.3(a).An open in a signal line with Fan-out may result in multiple stuck fault, as it shown in figure2.3(b)

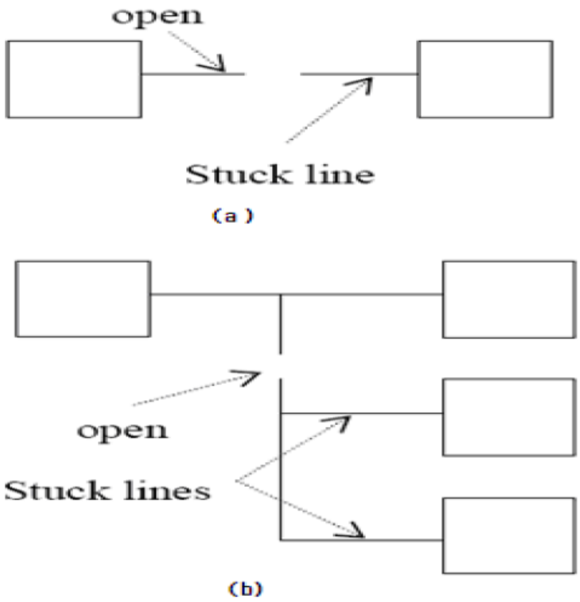


Figure 2.3: Stuck faults caused by open (a) Single stuck fault (b) Multiple stuck fault

Input A B	True Response	Faulty Response					
		A/0	B/0	Z/0	A/1	B/1	Z/1
00	0	0	0	0	0	0	1
01	0	0	0	0	1	0	1
10	0	0	0	0	0	1	1
11	0	0	0	0	1	1	1

Table 2.1: Stuck at fault on a two input AND gate

Chapter 3

Netlist And Benchmark Circuits

Netlist describes the circuit in the connection format. Netlist consists of net number and its connection with other. The netlist works as a input to any program. In this chapter standard format ISCAS is described along with some other circuits.

3.1 ISCAS Netlist

Any ATPG tool will take the Design Under Test (DUT) as an input and generate the test vector for given fault in the DUT. Most of the ATPG tool accepts DUT as an input in form of netlist (text file) and by processing on netlist, it generates the fault directory which will contain the fault and its corresponding test vector.

To prepare the input in the required format, we needed a standard format for netlist through which we can generate netlist for any given circuit. We were also requiring a standard benchmark type circuit set on which we can prove results of our future work for both of this reasons, we observed in many IEEE transactions that people are using ISCAS 85 (International Symposium for Circuits and Systems) combinational circuits and netlist format to prove the test related results.

We adopted the ISCAS 85 benchmark circuit set and netlist format for our work and proof of results. The reasons of ISCAS benchmark circuits to be selected as benchmark is given in this chapter.

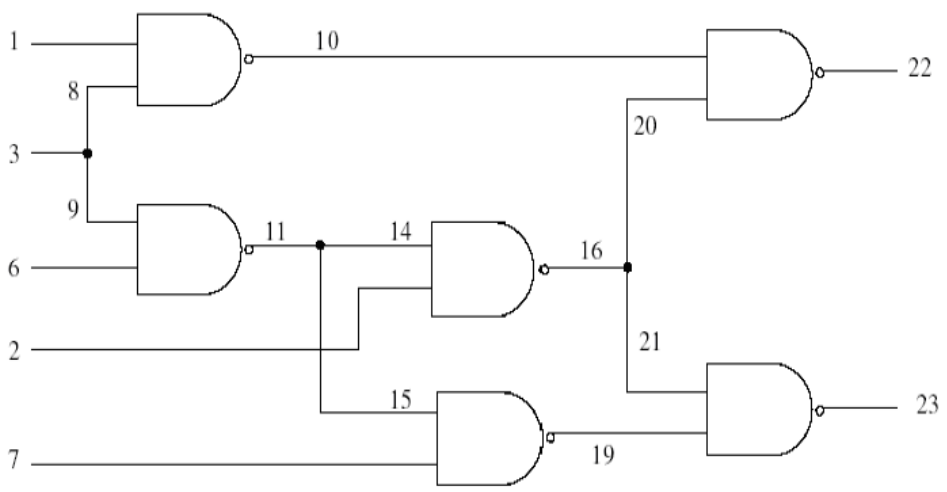


Figure 3.1: ISCAS 85 benchmark circuit C17

Netlist for C17 benchmark circuit is:

```
1 1gat inpt 1 0 0 0 0
2 2gat inpt 1 0 0 0 0
3 3gat inpt 2 0 0 0 0
8 8fan from 3gat 0 0 0 0
9 9fan from 3gat 0 0 0 0
6 6gat inpt 1 0 0 0 0
7 7gat inpt 1 0 0 0 0
10 10gat nand 1 2 1 8 0
11 11gat nand 2 2 9 6 0
14 14fan from 11gat 0 0 0 0
15 15fan from 11gat 0 0 0 0
16 16gat nand 2 2 2 14 0
20 20fan from 16gat 0 0 0 0
21 21fan from 16gat 0 0 0 0
19 19gat nand 1 2 15 7 0
22 22gat nand 0 2 10 20 0
23 23gat nand 0 2 21 19 0
```

- 1st column Represent net no.

- 2nd column: Describe type of net weather it is gate type or Fan-out branch.
- 3rd column: It represent type of Net/Gate which can be input, Stem/Branch or one of logic gate such as NOR, NAND, AND, OR, XOR, XNOR, or NOT. Inpt is a primary input of the circuit. The from is a fanout branch which is connected to a fanout stem specified in the next column.
- 4th column: It describe no of fanout branch comes out from Particular net.
- 5th column: Represent number of input particular gate has and for branch type net it represent from which stem it comes from.
- 6th column and 7th column: If net type is logic gate, these columns specify the net numbers of the inputs of the gate fan-in1 ,fan-in2.

As described above, C17 combinational circuit is used as benchmark circuit at most of the forums. The author assumes that the reason of it being accepted as bench mark is its following qualities:

- Fanout at various nodes
- Reconvergent fanout from internal node
- Reconvergent fanout at primary input

But the author has found that it contains only the NAND gate. To check the controllability, Observability and fault reduction for variety of other gates like AND, OR and NOR, we require one more circuit which contains all this four types of gates.

3.2 Other Circuit its ISCAS Net-list

Consider random logic circuit consisting of different gate. ISCAS net-list in text format is produced from the above circuit diagram. In this net-list if any fan-in input is not present than simply write 0 instead of it. So each raw has 8 column, and no of row equal to no of net in circuit.

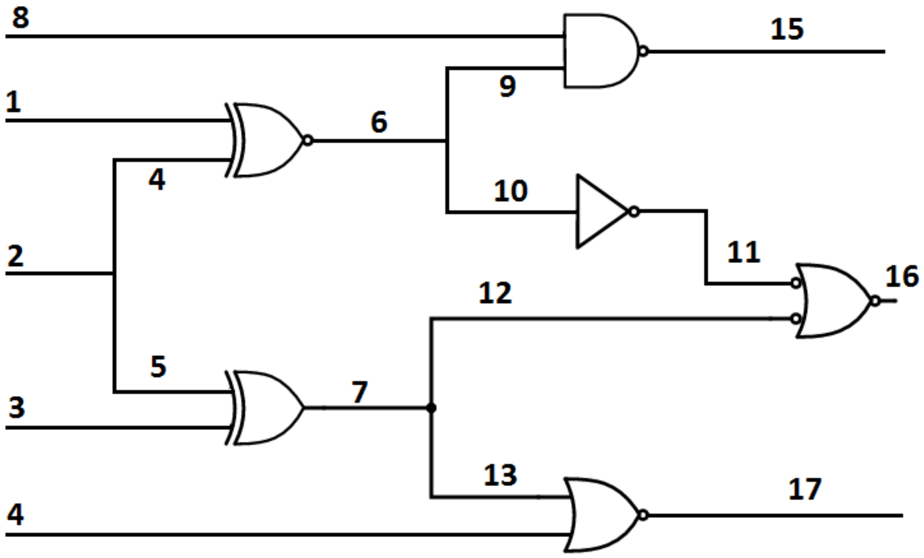


Figure 3.2: Logic circuit consisting of different gate

ISCAS netlist for this circuit

1 1gat inpt 1 0 0 0 0

2 2gat inpt 2 0 0 0 0

3 3gat inpt 1 0 0 0 0

4 4fan from 2 0 0 0 0

5 5fan from 2 0 0 0 0

6 6gat xnor 1 2 1 4 0

7 7gat xor 1 2 3 5 0

8 8gat inpt 1 0 0 0 0

9 9fan from 6 0 0 0 0

10 10fan from 6 0 0 0 0

11 11gat not 1 1 10 0 0

12 12fan from 7 0 0 0 0

13 13fan from 7 0 0 0 0

14 14gat inpt 1 0 0 0 0

15 15gat nand 0 2 8 9 0

16 16gat and 0 2 11 12 0

17 17gat nor 0 2 13 14 0

Chapter 4

Fault Detection and Reduced Fault Directory

For the combinational circuits, single stuck at fault detection method is discussed in this chapter. The number of stuck at faults in circuit is equal to 2n where n is total number net. Number of fault can be reduced by fault equivalence and fault dominance. these two concepts are discussed in this chapter.

4.1 Fault Detection

After the understanding of types of faults the fault is required to be detected and for it there are different techniques applied to detect the fault. To detect a fault a test vector 't' is applied to the circuit and the response of the faulty circuit to t denoted by $Z_f(t)$ is compared with fault free response Z(t) of the circuit and if both are equal then it indicates a fault free circuit.

For a single output circuit, a test X that detects a fault f produce function Z (X) = 0 and $Z_f(X) = 1$ or vice verse, hence the set of test vector which detects a fault is given by the solution of the equation.

$$\text{Z(x)} \oplus \text{Z}_f(x) = 1$$

Where Z (x) is the function of the fault free circuit and Z_f (x) is a function of faulty circuit. In analyzing a circuit mostly single stuck at faults are considered as it represents many other physical defect, also it is technology independent and the number of single stuck at faults are also less in the circuit. For n number of lines the single stuck at fault would be 2n. Using fault equivalence it is further reduced.

4.2 Reduced Fault Directory

If the net-list has total n lines, then it has total 2n possible faults. For large circuit fault list will be very high. Analysis of some of the faults can be avoided based on the fault equivalence, fault dominance and check point concepts.

4.2.1 Fault Equivalence

Faults of logic circuit are called equivalent if they produce same output function for all input test pattern. Equivalent fault have exactly same set of test for any logic circuit.(1)

Circuit with n number of net have 2n single stuck at fault present. Fault equivalence can further reduce stuck at fault from circuit. Fault equivalence first applied to simple logic gate, than it will be applied to whole circuit. Two Fault F G are consider as equivalent if $Z_F(x) = Z_G(x)$. No input test pattern can change functionality of F and G. Any test that can detect fault F can also detect fault G also. i.e. consider AND gate with input A , B and output C for input A stuck at 0 is equivalent to output C stuck at 0. It will produce same test pattern for both fault A and C stuck at 0.

Figure 4.1 shows the reduction in faults using fault equivalence method. Here both the faults s-a-0 and s-a-1 are present for wire, stem and primary output (PO). Fault collapsing is started from inputs to output. Figure 4.3 shows one of the examples to reduced fault directory using fault equivalence method.

Here • represent s_a_1 fault and o represent s_a_0 fault. Fault equivalence partition all fault in to equivalence classes, Only one fault from each class represent whole class. It very important for test pattern generation. Reduce size of fault list. Also useful in finding location of fault.

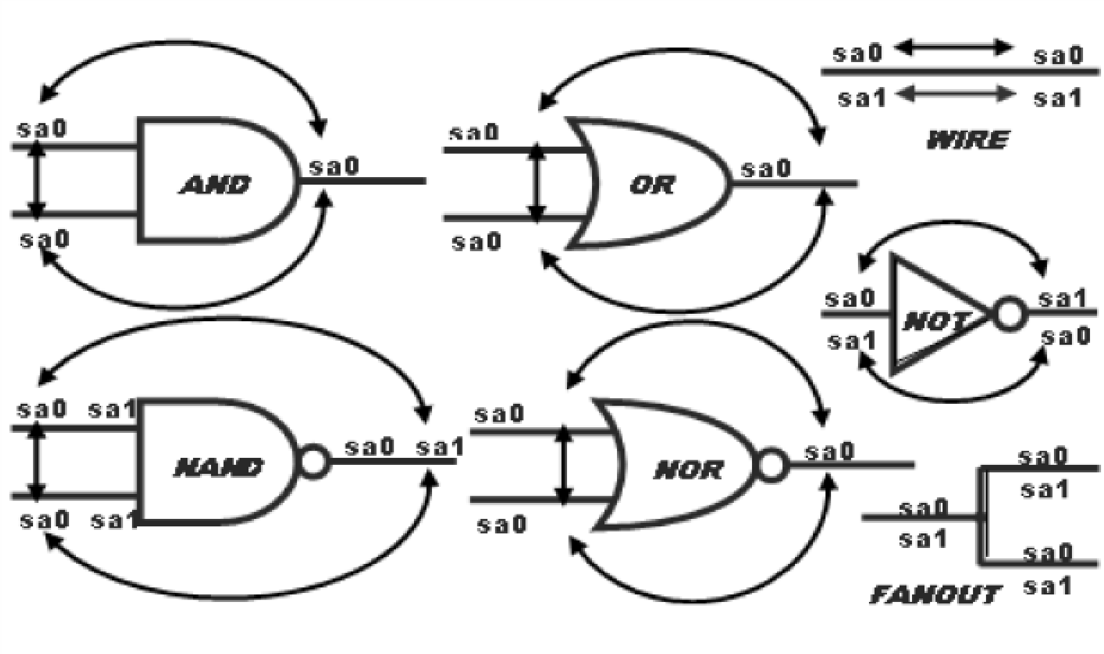


Figure 4.1: Fault equivalence concept

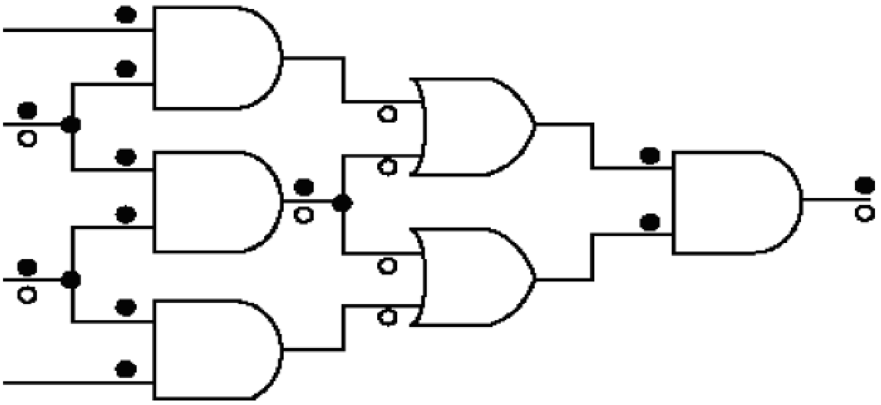


Figure 4.2: Example of Fault equivalence concept

Consider above circuit in which fault equivalence is started from output side. Both stuck at fault present on primary output. As it is AND gate at PO stuck at 0 fault can be removed from output. next stage consisting of OR gate so we can remove stuck at 1 fault from its both input. Now consider stem/Branch case, at stem both fault are present.

Fault Collapsing factor

In above circuit there are 16 net so 32 single stuck at fault present in circuit but fault equivalence reduced fault-list to 20. So 12 fault are removed from fault list. Fault collapsing factor can be defined as ratio of number of available fault after fault equivalence to total no of single stuck at fault available in circuit.

Fault Collapsing factor = 20/32

4.2.2 Fault Dominance

Fault Dominance can be used to further reduce fault list when fault detection is objective not diagnosis. If fault detection is the objective (not diagnosis), then fault dominance can be used to further reduce the fault list. Fault f said to be dominating fault g if all test pattern used to find g T_g, is subset of all test pattern of T_f.

Definition: Fault f dominates g if f and g are functionally equivalent under T_g

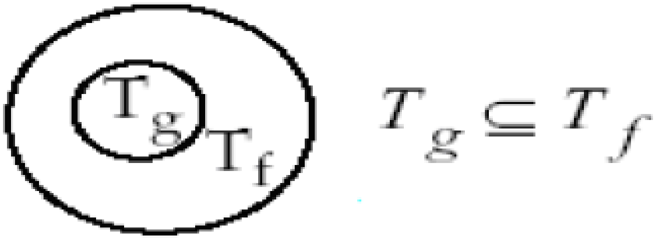


Figure 4.3: Fault Dominance

Test pattern which can detect fault g can also detect fault f. Since g implies f, Discard fault list of f and consider only g fault list. For example, S_A_1 test pattern for fault g for one input of AND gate can also used to detect f which is S_A_0 at Output. Than S_A_0 can be removed from fault list.

4.2.3 Implementation of Fault Equivalence

The program is implemented using C language .The logic to implement the concept of reduced fault directory using fault equivalence is given in this section. Flow chart is also shown.

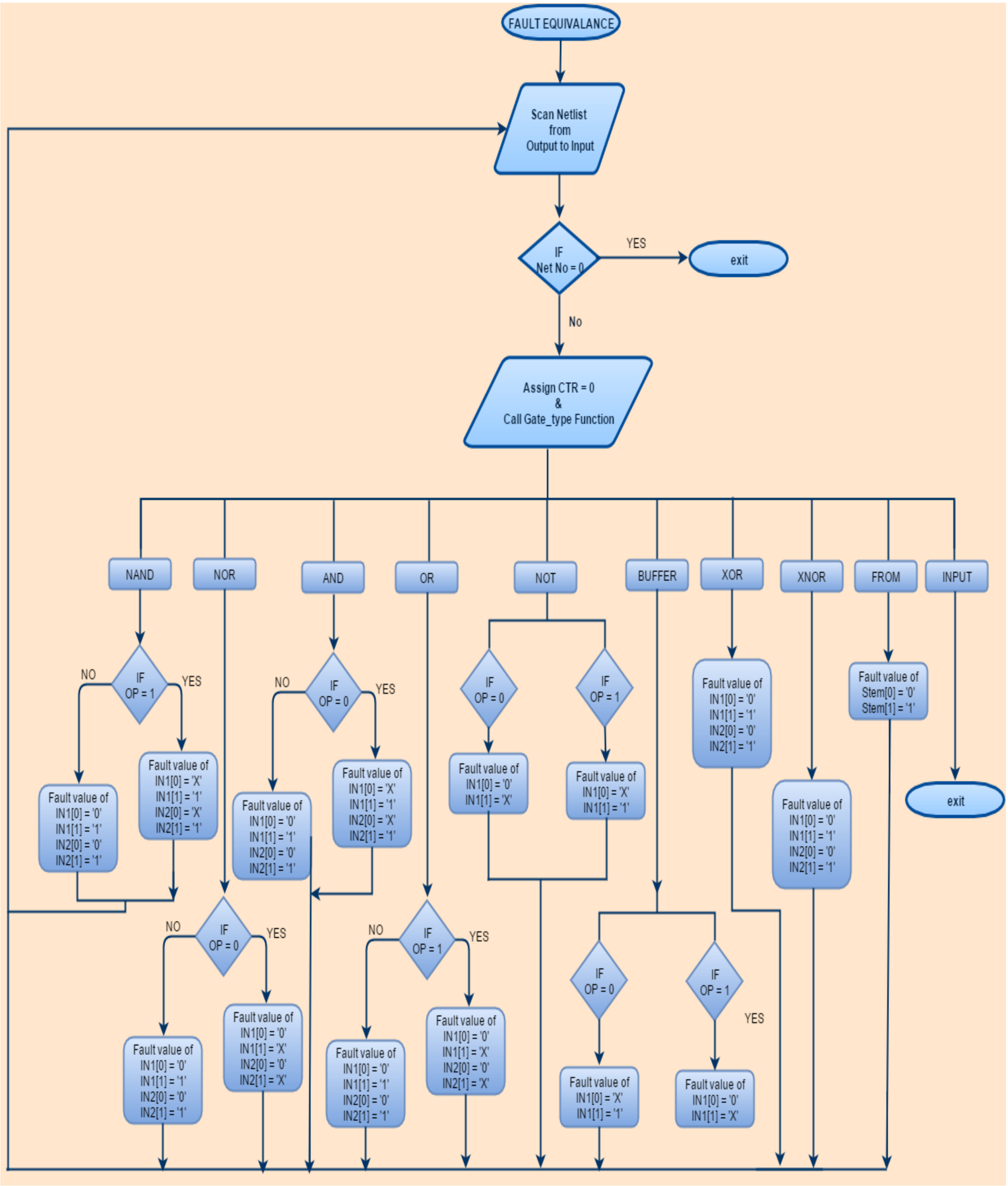

FAULT EQUIVALANCE
Scan Netlist from Output to Input
IF Net No = 0
YES
exit
No
Assign CTR = 0 & Call Gate_type Function
NAND
NOR
AND
OR
NOT
BUFFER
XOR
XNOR
FROM
INPUT
NO
IF OP = 1
YES
Fault value of IN1[0] = '0' IN1[1] = '1' IN2[0] = '0' IN2[1] = '1'
Fault value of IN1[0] = 'X' IN1[1] = '1' IN2[0] = 'X' IN2[1] = '1'
NO
IF OP = 0
YES
Fault value of IN1[0] = '0' IN1[1] = '1' IN2[0] = '0' IN2[1] = '1'
Fault value of IN1[0] = 'X' IN1[1] = '1' IN2[0] = 'X' IN2[1] = '1'
IF OP = 0
Fault value of IN1[0] = '0' IN1[1] = 'X'
IF OP = 1
Fault value of IN1[0] = 'X' IN1[1] = '1'
Fault value of IN1[0] = '0' IN1[1] = '1' IN2[0] = '0' IN2[1] = '1'
Fault value of Stem[0] = '0' Stem[1] = '1'
Fault value of IN1[0] = '0' IN1[1] = '1' IN2[0] = '0' IN2[1] = '1'
exit
NO
IF OP = 0
YES
Fault value of IN1[0] = '0' IN1[1] = '1' IN2[0] = '0' IN2[1] = '1'
Fault value of IN1[0] = '0' IN1[1] = 'X' IN2[0] = '0' IN2[1] = 'X'
NO
IF OP = 1
YES
Fault value of IN1[0] = '0' IN1[1] = '1' IN2[0] = '0' IN2[1] = '1'
Fault value of IN1[0] = '0' IN1[1] = 'X' IN2[0] = '0' IN2[1] = 'X'
IF OP = 0
Fault value of IN1[0] = 'X' IN1[1] = '1'
IF OP = 1
YES
Fault value of IN1[0] = '0' IN1[1] = 'X'

Figure 4.4: Flow chart for Fault Equivalence

Implementation of Fault Equivalence for C17 Benchmark circuit

for net 1 fault is = 1

for net 2 fault is = 1

for net 3 fault is = 0

for net 3 fault is = 1

for net 8 fault is = 1

for net 9 fault is = 1

for net 6 fault is = 1

for net 7 fault is = 1

for net 10 fault is = 1

for net 11 fault is = 0

for net 11 fault is = 1

for net 14 fault is = 1

for net 15 fault is = 1

for net 16 fault is = 0

for net 16 fault is = 1

for net 20 fault is = 1

for net 21 fault is = 1

for net 19 fault is = 1

for net 22 fault is = 0

for net 22 fault is = 1

for net 23 fault is = 0

for net 23 fault is = 1

Chapter 5

Testability Measures

Testability measure are used to guide ATPG algorithm. In this chapter will discuss about testability measures like controllability and observability. And its Logic, Flowchart and advantage of using testability measure in ATPG algorithm.

5.1 Controllability and Observability

Testability measure are used to define how difficult to Observe/Control internal node of circuit from Primary output/input. In ATPG algorithm search process involve to make important decision. First is to solve problem at existing stage at execution of algorithm. And second is to select possible way to solve problem. Testability measure are helpful in different selection criteria of algorithm.

Goldstein was the first one to implement a computer program to calculate those controllability and Observability values which he named as SCOAP acronym for Sandia Controllability and Observability Analysis Program.Testability Measure are of two type.

1. Controllability

2. Observability

5.1.1 Controllability

Controllability for digital circuit is defined as difficulty of setting particular net to logic level 0 or logic level 1.

Controllability measures are used in line justification problem in ATPG Algorithm. It is selecting net which is easiest to control from primary input. Controllability value is minimum for primary input because user can directly control it. Let assume all input have controllability value (1,1). It is calculated from input to Output. Controllability value is increased as we move toward output net. For the fanout the controllability values remain to be same as for the line from which it is emerging.

If a and b are the inputs of a gate and z is the output then the following are the controllability values for different gates.

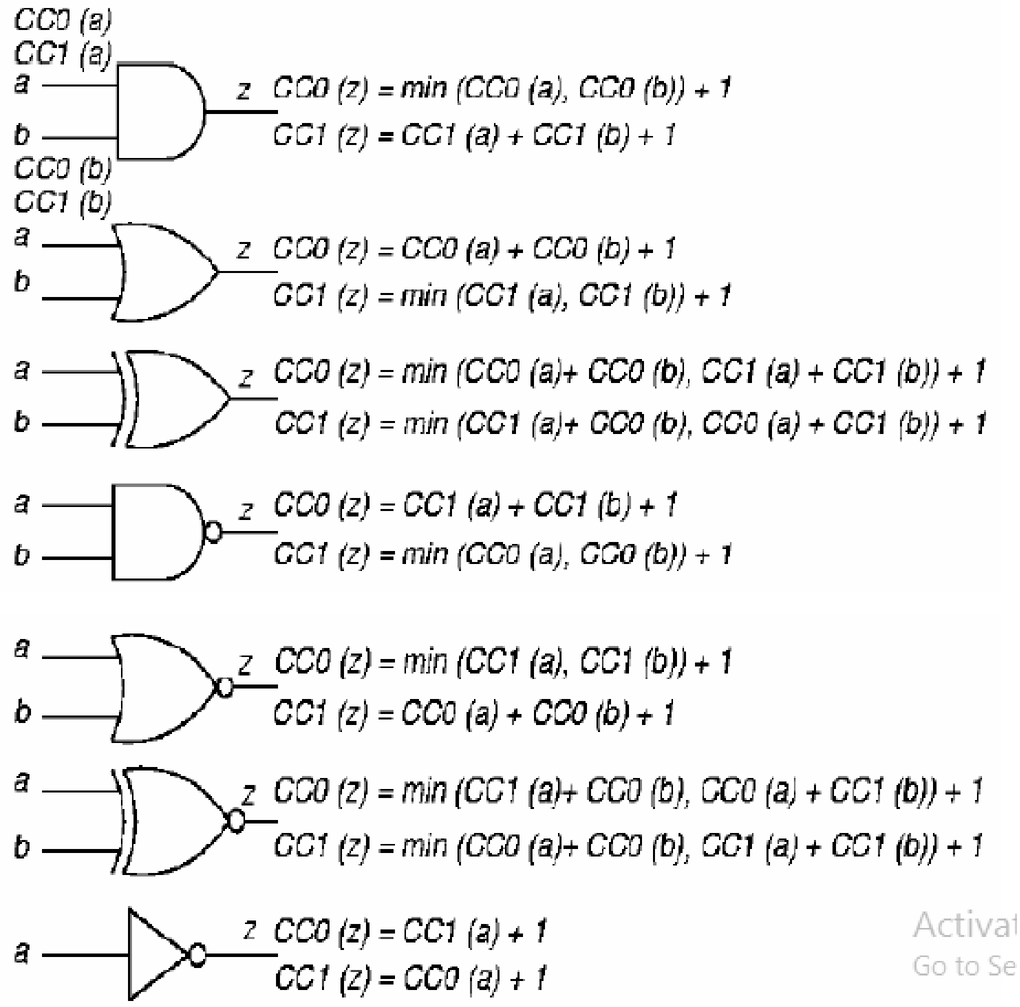

CC0 (a)
CC1 (a)
a
b
CC0 (b)
CC1 (b)
z
CC0 (z) = min (CC0 (a), CC0 (b)) + 1
CC1 (z) = CC1 (a) + CC1 (b) + 1
a
b
z
CC0 (z) = CC0 (a) + CC0 (b) + 1
CC1 (z) = min (CC1 (a), CC1 (b)) + 1
a
b
z
CC0 (z) = min (CC0 (a)+ CC0 (b), CC1 (a) + CC1 (b)) + 1
CC1 (z) = min (CC1 (a)+ CC0 (b), CC0 (a) + CC1 (b)) + 1
a
b
z
CC0 (z) = CC1 (a) + CC1 (b) + 1
CC1 (z) = min (CC0 (a), CC0 (b)) + 1
a
b
z
CC0 (z) = min (CC1 (a), CC1 (b)) + 1
CC1 (z) = CC0 (a) + CC0 (b) + 1
a
b
z
CC0 (z) = min (CC1 (a)+ CC0 (b), CC0 (a) + CC1 (b)) + 1
CC1 (z) = min (CC0 (a)+ CC0 (b), CC1 (a) + CC1 (b)) + 1
a
z
CC0 (z) = CC1 (a) + 1
CC1 (z) = CC0 (a) + 1
Activat
Go to Se

Figure 5.1: CC0 and CC1 values

5.1.2 Observability

Observability for a digital circuit is defined as how difficult to observe particular net from Primary output.

Observability measure are used in fault propagation. provide net which are easy for propagation of fault to output. It is used mainly at stem to decide which branch is easy for fault propagation. Observability value is calculated from output to input. Any value is more observable at output so its value is minimum at primary output. Let us assume Observability value = 1 at output. Its value is increased as we move toward input. Observability value at stem calculated differently. For different gate Observability value is given below.5.2

Moreover for the branches emerging from the same stem the observability value is defined as follows

CO(a)=min(CO(z1),CO(z2),,CO(zn))

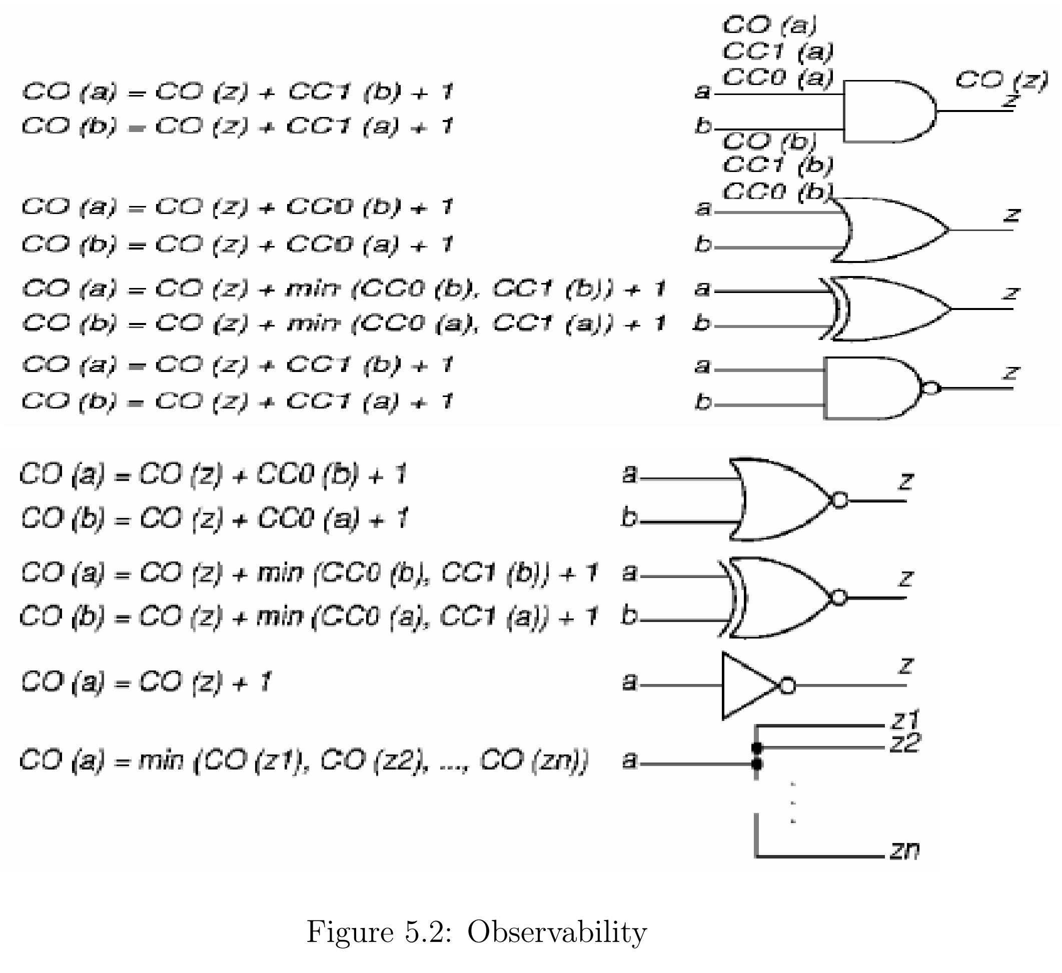

CO (a) = CO (z) + CC1 (b) + 1
CO (b) = CO (z) + CC1 (a) + 1
CO (a)
CC1 (a)
CC0 (a)
CO (z)
a
b
z
CO (b)
CC1 (b)
CC0 (b)
CO (a) = CO (z) + CC0 (b) + 1
CO (b) = CO (z) + CC0 (a) + 1
a
b
z
CO (a) = CO (z) + min (CC0 (b), CC1 (b)) + 1
CO (b) = CO (z) + min (CC0 (a), CC1 (a)) + 1
a
b
z
CO (a) = CO (z) + CC1 (b) + 1
CO (b) = CO (z) + CC1 (a) + 1
a
b
z
CO (a) = CO (z) + CC0 (b) + 1
CO (b) = CO (z) + CC0 (a) + 1
a
b
z
CO (a) = CO (z) + min (CC0 (b), CC1 (b)) + 1
CO (b) = CO (z) + min (CC0 (a), CC1 (a)) + 1
a
b
z
CO (a) = CO (z) + 1
a
z
CO (a) = min (CO (z1), CO (z2), ..., CO (zn))
a
z1
z2
zn

Figure 5.2: Observability

5.2 Implementation of Testability Measures

The testability measures controllability 0 (CC0), controllability 1 (CC1) and observability is calculated using C language and stored in the form of array contains net number, CC0 ,CC1 , O .The logic for implementation, flow chart is given in this section.

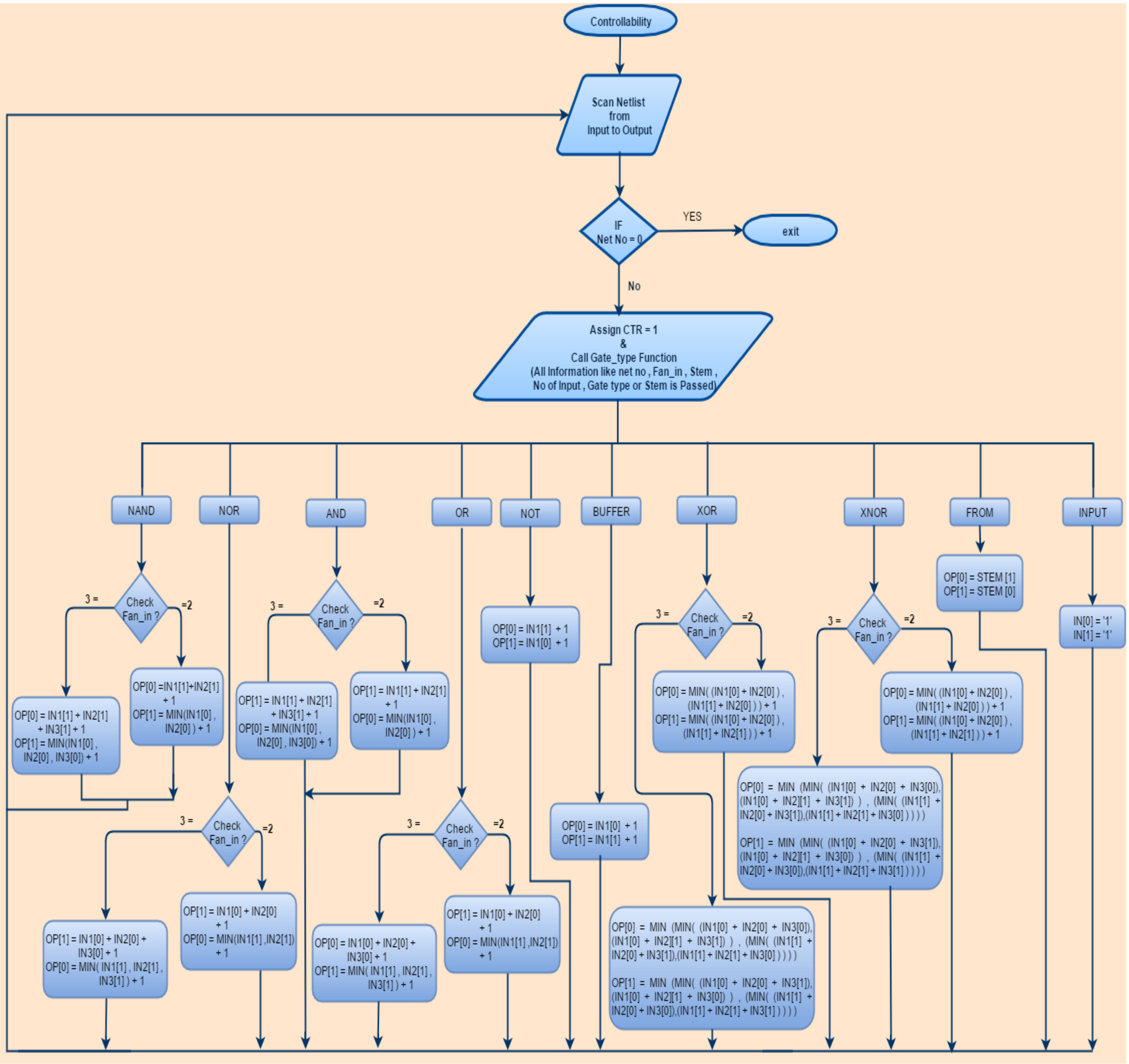

Controllability
Scan Netlist from Input to Output
IF Net No = 0
YES
exit
No
Assign CTR = 1 & Call Gate_type Function (All Information like net no , Fan_in , Stem , No of Input , Gate type or Stem is Passed)
NAND
NOR
AND
OR
NOT
BUFFER
XOR
XNOR
FROM
INPUT
3 =
Check Fan_in ?
=2
OP[0] = IN1[1] + IN2[1] + IN3[1] + 1
OP[1] = MIN(IN1[0] , IN2[0] , IN3[0]) + 1
OP[0] =IN1[1]+IN2[1] + 1
OP[1] = MIN(IN1[0] , IN2[0]) + 1
OP[1] = IN1[1] + IN2[1] + IN3[1] + 1
OP[0] = MIN(IN1[0] , IN2[0] , IN3[0]) + 1
OP[1] = IN1[1] + IN2[1] + 1
OP[0] = MIN(IN1[0] , IN2[0]) + 1
OP[0] = IN1[1] + 1
OP[1] = IN1[0] + 1
OP[0] = MIN((IN1[0] + IN2[0]) , (IN1[1] + IN2[0])) + 1
OP[1] = MIN((IN1[0] + IN2[0]) , (IN1[1] + IN2[1])) + 1
OP[0] = STEM [1]
OP[1] = STEM [0]
IN[0] = '1'
IN[1] = '1'
OP[0] = MIN (MIN((IN1[0] + IN2[0] + IN3[0]), (IN1[0] + IN2][1] + IN3[1])) , (MIN((IN1[1] + IN2[0] + IN3[1]),(IN1[1] + IN2[1] + IN3[0]))))
OP[1] = MIN (MIN((IN1[0] + IN2[0] + IN3[1]), (IN1[0] + IN2][1] + IN3[0])) , (MIN((IN1[1] + IN2[0] + IN3[0]),(IN1[1] + IN2[1] + IN3[1]))))
OP[1] = IN1[0] + IN2[0] + IN3[0] + 1
OP[0] = MIN(IN1[1] , IN2[1] , IN3[1]) + 1
OP[1] = IN1[0] + IN2[0] + 1
OP[0] = MIN(IN1[1] ,IN2[1]) + 1
OP[0] = IN1[0] + IN2[0] + IN3[0] + 1
OP[1] = MIN(IN1[1] , IN2[1] , IN3[1]) + 1
OP[1] = IN1[0] + IN2[0] + 1
OP[0] = MIN(IN1[1] ,IN2[1]) + 1
OP[0] = IN1[0] + 1
OP[1] = IN1[1] + 1

Figure 5.3: Controllability

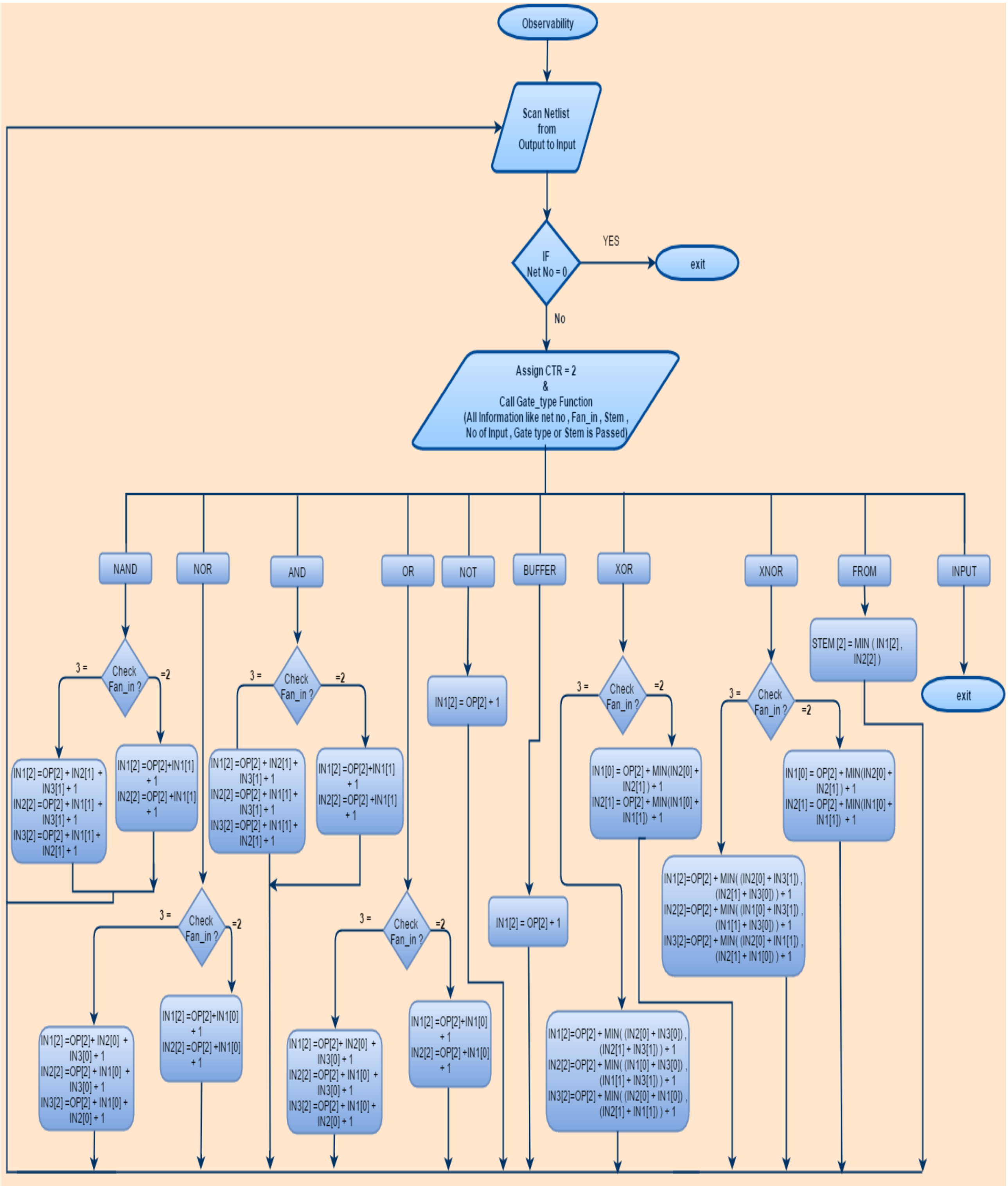

Observability
Scan Netlist from Output to Input
IF Net No = 0
YES
exit
No
Assign CTR = 2 & Call Gate_type Function (All Information like net no , Fan_in , Stem , No of Input , Gate type or Stem is Passed)
NAND
NOR
AND
OR
NOT
BUFFER
XOR
XNOR
FROM
INPUT
Check Fan_in ?
3 =
=2
IN1[2] =OP[2] + IN2[1] + IN3[1] + 1
IN2[2] =OP[2] + IN1[1] + IN3[1] + 1
IN3[2] =OP[2] + IN1[1] + IN2[1] + 1
IN1[2] =OP[2]+IN1[1] + 1
IN2[2] =OP[2] +IN1[1] + 1
IN1[2] =OP[2]+ IN2[0] + IN3[0] + 1
IN2[2] =OP[2] + IN1[0] + IN3[0] + 1
IN3[2] =OP[2] + IN1[0] + IN2[0] + 1
IN1[2] =OP[2]+IN1[0] + 1
IN2[2] =OP[2] +IN1[0] + 1
IN1[2] = OP[2] + 1
IN1[0] = OP[2] + MIN(IN2[0] + IN2[1]) + 1
IN2[1] = OP[2] + MIN(IN1[0] + IN1[1]) + 1
IN1[2]=OP[2] + MIN((IN2[0] + IN3[0]) , (IN2[1] + IN3[1])) + 1
IN2[2]=OP[2] + MIN((IN1[0] + IN3[0]) , (IN1[1] + IN3[1])) + 1
IN3[2]=OP[2] + MIN((IN2[0] + IN1[0]) , (IN2[1] + IN1[1])) + 1
IN1[2]=OP[2] + MIN((IN2[0] + IN3[1]) , (IN2[1] + IN3[0])) + 1
IN2[2]=OP[2] + MIN((IN1[0] + IN3[1]) , (IN1[1] + IN3[0])) + 1
IN3[2]=OP[2] + MIN((IN2[0] + IN1[1]) , (IN2[1] + IN1[0])) + 1
STEM [2] = MIN (IN1[2] , IN2[2])
exit

Figure 5.4: Observability

Implementation of Testability Measure for C17 Benchmark circuit is given below, First column represent net no in bracket first column is controllability 0, Second column is Controllability 1 and third column is Observability value.

net1 *cont_obs* is (1 1 6)

net2 *cont_obs* is (1 1 7)

net3 *cont_obs* is (1 1 6)

net8 *cont_obs* is (1 1 6)

net9 *cont_obs* is (1 1 8)

net6 *cont_obs* is (1 1 8)

net7 *cont_obs* is (1 1 7)

net10 *cont_obs* is (3 2 4)

net11 *cont_obs* is (3 2 6)

net14 *cont_obs* is (3 2 6)

net15 *cont_obs* is (3 2 6)

net16 *cont_obs* is (4 2 4)

net20 *cont_obs* is (4 2 4)

net21 *cont_obs* is (4 2 4)

net19 *cont_obs* is (4 2 4)

net22 *cont_obs* is (5 4 1)

net23 *cont_obs* is (5 5 1)

Chapter 6

Probability

Probability define possibility of getting logic 0 or Logic 1 on any net of circuit. High Probability means more possibility of getting particular logic. Like testability measure it is not always linearly increase from PI to PO or vice verse. There may be such internal node present in circuit whose probability is less than output node.

In test pattern generation probability measure is used to find nets that have minimum probability. So we can say that net having minimum probability, fault on that net is hardest to find. We short all net according to probability wise and provide this fault list to ATPG algorithm. So fault that is undetectable or redundant is more likely to be find in early test pattern generation.

Probability of getting logic 1 or logic 0 on primary input net is (0.5 , 0.5).than it is propagated toward output. First we will apply probability value for simple gate and than we will carry forward it to whole circuit. and finding all net probability.

6.1 Implementation of Probability

Implementation of Probability is similar to testability measure first we find Primary input from net-list and assign probability value 0.5 to both logic 0 and logic 1. Than according to Gate type its value is changing from input to output.

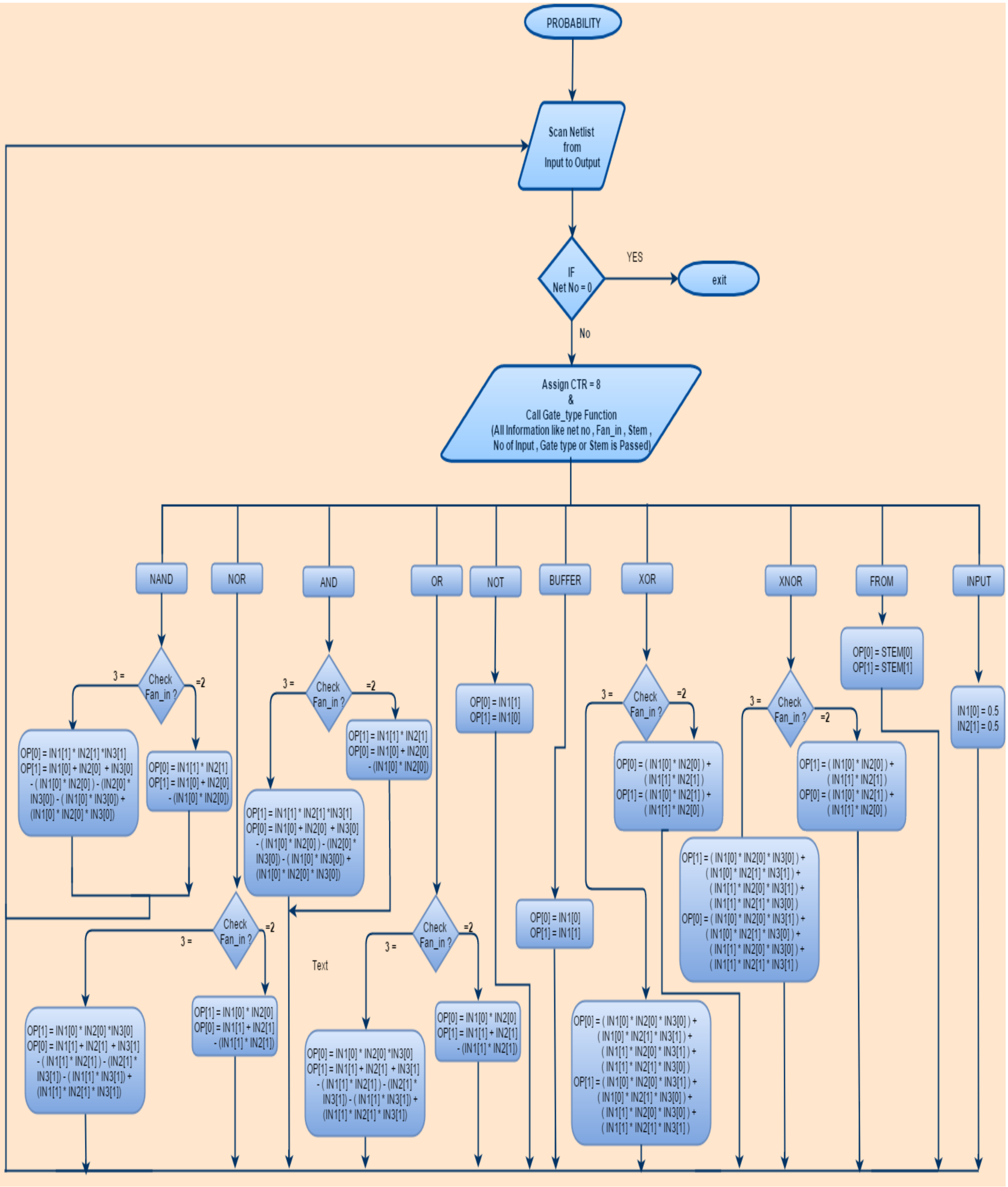

PROBABILITY
Scan Netlist from Input to Output
IF Net No = 0
YES
exit
No
Assign CTR = 8 & Call Gate_type Function (All Information like net no , Fan_in , Stem , No of Input , Gate type or Stem is Passed)
NAND
NOR
AND
OR
NOT
BUFFER
XOR
XNOR
FROM
INPUT
Check Fan_in ?
3 =
=2
OP[0] = IN1[1] * IN2[1] *IN3[1] OP[1] = IN1[0] + IN2[0] + IN3[0] - (IN1[0] * IN2[0]) - (IN2[0] * IN3[0]) - (IN1[0] * IN3[0]) + (IN1[0] * IN2[0] * IN3[0])
OP[0] = IN1[1] * IN2[1] OP[1] = IN1[0] + IN2[0] - (IN1[0] * IN2[0])
OP[1] = IN1[1] * IN2[1] *IN3[1] OP[0] = IN1[0] + IN2[0] + IN3[0] - (IN1[0] * IN2[0]) - (IN2[0] * IN3[0]) - (IN1[0] * IN3[0]) + (IN1[0] * IN2[0] * IN3[0])
OP[1] = IN1[1] * IN2[1] OP[0] = IN1[0] + IN2[0] - (IN1[0] * IN2[0])
OP[0] = IN1[1] OP[1] = IN1[0]
OP[0] = (IN1[0] * IN2[0]) + (IN1[1] * IN2[1]) OP[1] = (IN1[0] * IN2[1]) + (IN1[1] * IN2[0])
OP[1] = (IN1[0] * IN2[0]) + (IN1[1] * IN2[1]) OP[0] = (IN1[0] * IN2[1]) + (IN1[1] * IN2[0])
OP[0] = STEM[0] OP[1] = STEM[1]
IN1[0] = 0.5 IN2[1] = 0.5
OP[1] = (IN1[0] * IN2[0] * IN3[0]) + (IN1[0] * IN2[1] * IN3[1]) + (IN1[1] * IN2[0] * IN3[1]) + (IN1[1] * IN2[1] * IN3[0]) OP[0] = (IN1[0] * IN2[0] * IN3[1]) + (IN1[0] * IN2[1] * IN3[0]) + (IN1[1] * IN2[0] * IN3[0]) + (IN1[1] * IN2[1] * IN3[1])
OP[0] = IN1[0] OP[1] = IN1[1]
Text
OP[1] = IN1[0] * IN2[0] *IN3[0] OP[0] = IN1[1] + IN2[1] + IN3[1] - (IN1[1] * IN2[1]) - (IN2[1] * IN3[1]) - (IN1[1] * IN3[1]) + (IN1[1] * IN2[1] * IN3[1])
OP[1] = IN1[0] * IN2[0] OP[0] = IN1[1] + IN2[1] - (IN1[1] * IN2[1])
OP[0] = IN1[0] * IN2[0] *IN3[0] OP[1] = IN1[1] + IN2[1] + IN3[1] - (IN1[1] * IN2[1]) - (IN2[1] * IN3[1]) - (IN1[1] * IN3[1]) + (IN1[1] * IN2[1] * IN3[1])
OP[0] = IN1[0] * IN2[0] OP[1] = IN1[1] + IN2[1] - (IN1[1] * IN2[1])
OP[0] = (IN1[0] * IN2[0] * IN3[0]) + (IN1[0] * IN2[1] * IN3[1]) + (IN1[1] * IN2[0] * IN3[1]) + (IN1[1] * IN2[1] * IN3[0]) OP[1] = (IN1[0] * IN2[0] * IN3[1]) + (IN1[0] * IN2[1] * IN3[0]) + (IN1[1] * IN2[0] * IN3[0]) + (IN1[1] * IN2[1] * IN3[1])

Figure 6.1: Flow chat of Probability

Implementation of Probability for C17 Benchmark circuit

```
net1 probability is (0.5000 0.5000)
net2 probability is (0.5000 0.5000)
net3 probability is (0.5000 0.5000)
net8 probability is (0.5000 0.5000)
net9 probability is (0.5000 0.5000)
net6 probability is (0.5000 0.5000)
net7 probability is (0.5000 0.5000)
net10 probability is (0.2500 0.7500)
net11 probability is (0.2500 0.7500)
net14 probability is (0.2500 0.7500)
net15 probability is (0.2500 0.7500)
net16 probability is (0.3750 0.6250)
net20 probability is (0.3750 0.6250)
net21 probability is (0.3750 0.6250)
net19 probability is (0.3750 0.6250)
net22 probability is (0.4688 0.5312)
```

```
net23 probability is (0.3906 0.6094)
```

Chapter 7

Automatic Test Pattern Generation

Basic process of test vector using line justification and fault propagation is discussed in this chapter along with the results generated by C program.

7.1 Basic of ATPG

The objective to test circuits is to separate faulty ones from good ones. ATPG (Automatic Test Pattern Generator) is an EDA tool which is used to find an input or test pattern, That is when applied to a logic circuit, enables automatic test equipment to distinguish between the correct circuit behaviour and the faulty circuit behaviour caused by fault. Test pattern are used to test logic circuit after manufacture and also used in failure analysis. Effectiveness of ATPG is measured in term no of fault and generated test pattern. Higher fault detection larger test application time and test quality will be higher. It depend very much on type of fault model, The type of circuit under test can be full scan, Partial scan, synchronous/asynchronous or sequential. Circuit on which ATPG is applied can be gate, register-transistor or switch level. During manufacturing process defect can be introduced in the circuit. Fault model used to describe actual defect in circuit. Fault models are required to observe the effect of test. It is representation of physical fault on operation of logic circuit. Only considered logical functionality of circuit not timing. It provide mathematical model for circuit diagnosis and testing.Fault modeling is a main concept to be considered in designing, fabrication and testing of digital systems. ATPG process for any fault can be mainly divided in to 4 steps.

- Fault activation

- Fault Propagation
- line Justification
- Back-tracing

In Fault activation opposite value is generated on faulty net. for example if stuck at 0 fault is present than fault free value 1 is passed to faulty net. Now this fault must be propagated to output in order to observe faulty behavior of circuit which is called fault propagation. To propagate fault properly other net must be justified this process is called line justification. While line justification if any net value is conflicting than it must be back traced and other net selected for fault propagation/justification. After back-tracing also test pattern can not be generated for any fault than that fault is called undetectable fault. In such a circuit, any single fault will be inherently undetectable.

7.2 Test Generation

Test Generation (TG) is the process of determining the stimuli necessary to test a digital system. TG depends primarily on the testing method employed. Online testing methods do not require TG. Little TG efforts is needed when the input patterns are provided by a feedback shift register working as a pseudo random sequence generator. In contrast TG for design verification testing and the development of diagnostic programs involved a large effort that, unfortunately a still mainly a manual activity. Automatic TG refers to TG algorithms that, given a model of system can generate tests for it. A TG has been developed mainly for edged pin stored pattern testing. TG can be fault oriented or function oriented. In fault oriented TG one tries to generate test that will detect (and possibly locate) specific faults. In function oriented TG one tries to generate a test that, if it is passes shows that a system performs its specified function. Here, in this circuit if we want to detect fault at first fan out from x2, we will give x2=0. Because in case of x2=1, we will have ideal and erroneous output similar. But taking x2=0 in this circuit will give z1=1 instead of z1=0.
Here, Z1 , Z2 are fault free output and Z1f , Z2f are faulty outputs as follows

Implementation of Testability Measure for C17 Benchmark circuit
Z1=X1X2
Z2=X2X3
Z1f=X1

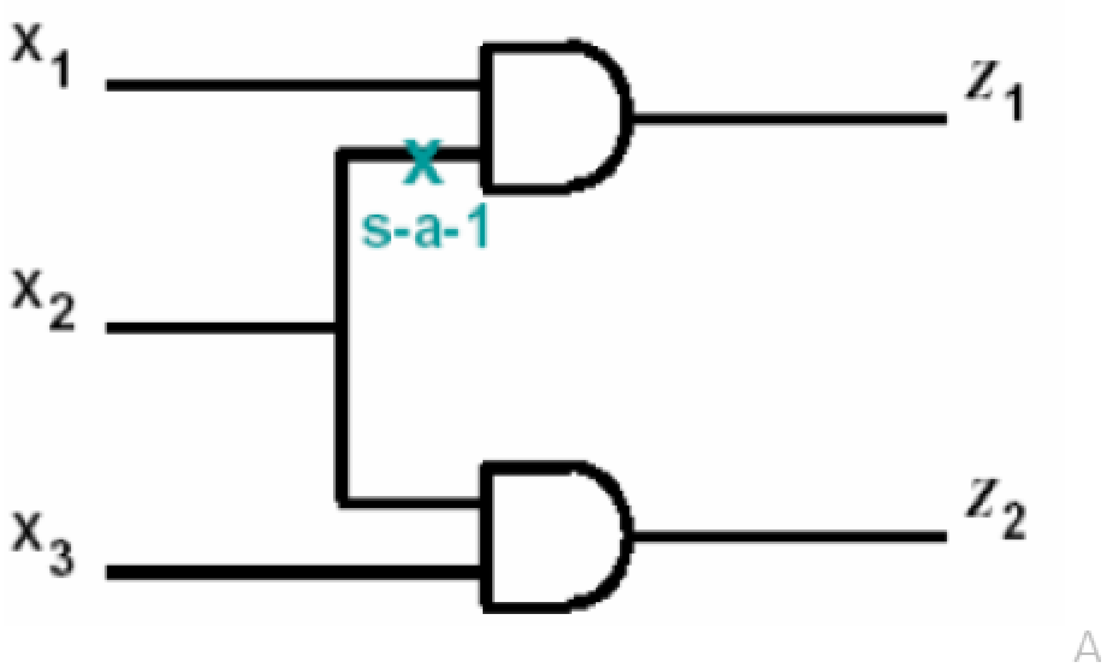


Figure 7.1: Example of Fault Detection

Z2f=X2X3
Test (X1, X2, X3) = (100) detects fault s a 1 because Z1 (100)=0 and Z1f (100)=1

7.2.1 Fault Sensitization

A net whose value is changed due to fault present on that net by test pattern t than it is said that fault f is sensitized by test t. A path that consist of sensitized net is called sensitized path.Here in the example shown in figure 7.2 due to fault on the output of G2 the affected path is shown by different color, which is sensitized path in this case.

7.2.2 Basic Steps of Fault Oriented ATPG

- Fault Activation
- Fault Propagation

Fault Activation means to set primary input PI values such that it causes the line having the fault v to the value v'. This is an instance of the line justification problem which deals with finding an assignment of PI values those results in a desired value setting on a specified

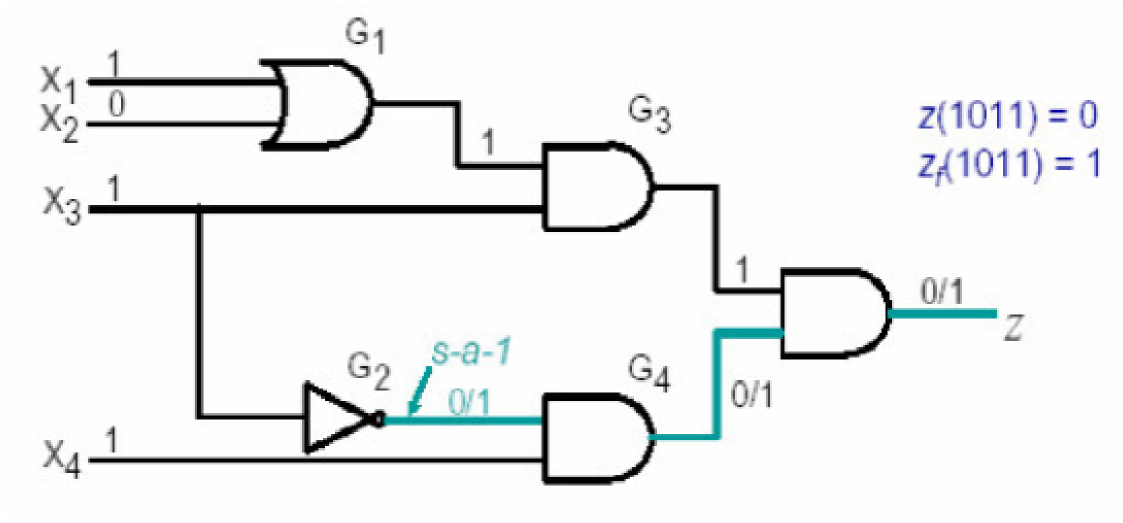


Figure 7.2: Fault sensitisation

line in the circuit. Whereas the term Fault Propagation means to make the primary output bear value such as the fault is on that particular point itself. Which in other words mean that we must propagate the erroneous values to the PO via the best sensitized path? Justify and Propagate both are the basic steps of any generic path sensitized algorithm.

7.3 Different Methods of ATPG Algorithm

Three of the best known algorithms for combinational ATPG are

- D - Algorithm:
  First ATPG algorithm was developed by Roth, which is called D-Algorithm, D-cubes are used to propagate fault in circuit. Basic Algorithm for fault propagation, justification. In which search space for fault is very high.

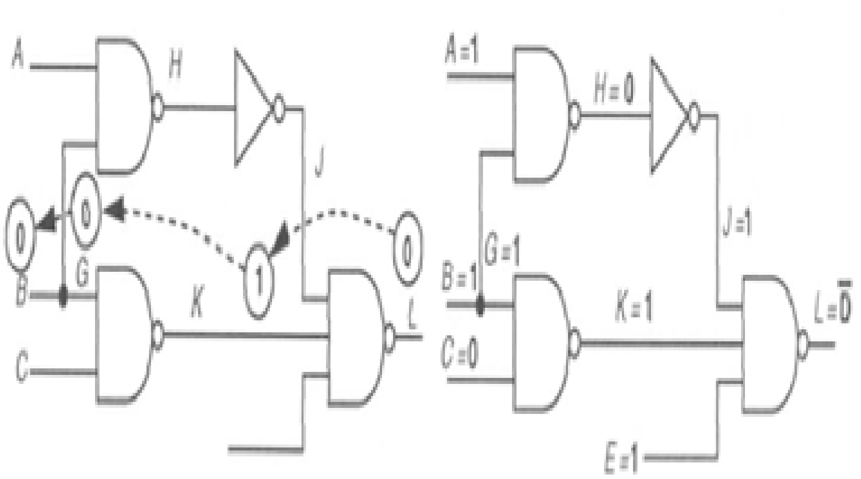
- PODEM: (Path oriented decision making)
  Goel's PODEM algorithm, This is first algorithm which used testability measure to effectively guide fault propagation and line justification problem in D algorithm.He efficiently used path propagation constraints to limit the ATPG algorithm search space, and introduced the notion of back-trace.

- FAN - Algorithm: (Fan out oriented)

Fujiwara and Shimono's FAN algorithm is advancement to D and PODEM algorithm. Search space is further reduced by effectively selecting particular net.hey efficiently constrained the backtrace to speed up search. This algorithm use multiple line justification and head line concept to further speed up ATPG operation.

7.3.1 Differences Between FAN And PODEM

Immediate Implications:
PODEM algorithm select one one net at a time and assign value to it, while misses opportunity to assign immediate value to uniquely defined net. Fujiwara and Shimono developed FAN algorithm does unique detemination of signal. Figure 7.4 shows how FAN and PODEM backtrace from output to input for signal L at NAND gate.



(a) Behavior of PODEM. (b) Behavior of FAN.

Figure 7.3: Immediate Implication

Unique Sensitization:
Fujiwara and Shimono observed that, Signals that are uniquely determined add opportunity to fault propagation. In Figure 7.26(b), we see that in some part of circuit there is only one path at which fault can be propagated. we can do immediate assignment to off path input along with signal propagation. Rather than discovering these signal assignments during search.

Headlines:

Fujiwara and Shimono developed the new concept of headlines, in which circuit is partitioned from the point where PI can be isolated from remaining circuit by cutting only single line. This line is called headline. Advantage of using headline is search space representing family of PI can be replaced with single headline value. This eliminates the unnecessary work of searching the decision tree shown in Figure 7.4 that PODEM does, and allows FAN to search the much simpler decision tree.

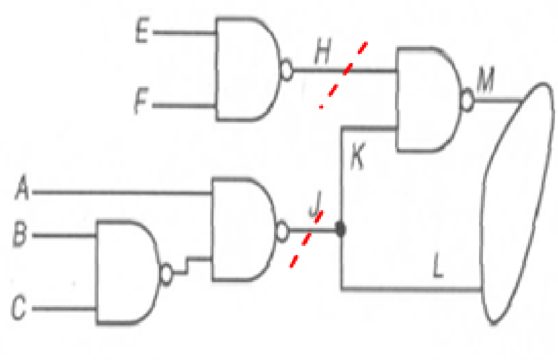


Figure 7.4: Unique Sensitization

Multiple Backtrace

FAN is breadth first multiple back-tracing procedure. While, PODEM is depth first back tracing procedure.

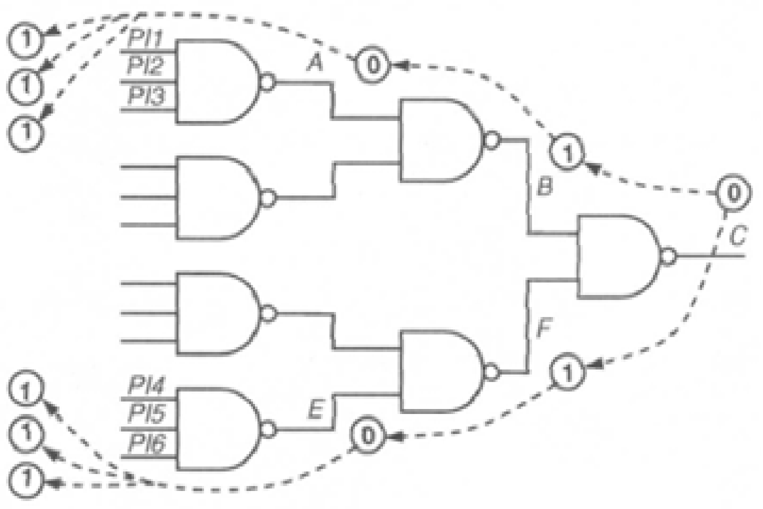


Figure 7.5: Headlines

7.4 Implementation of ATPG using Line Justification and Error Propagation

ATPG flow for any Fault in the Circuit

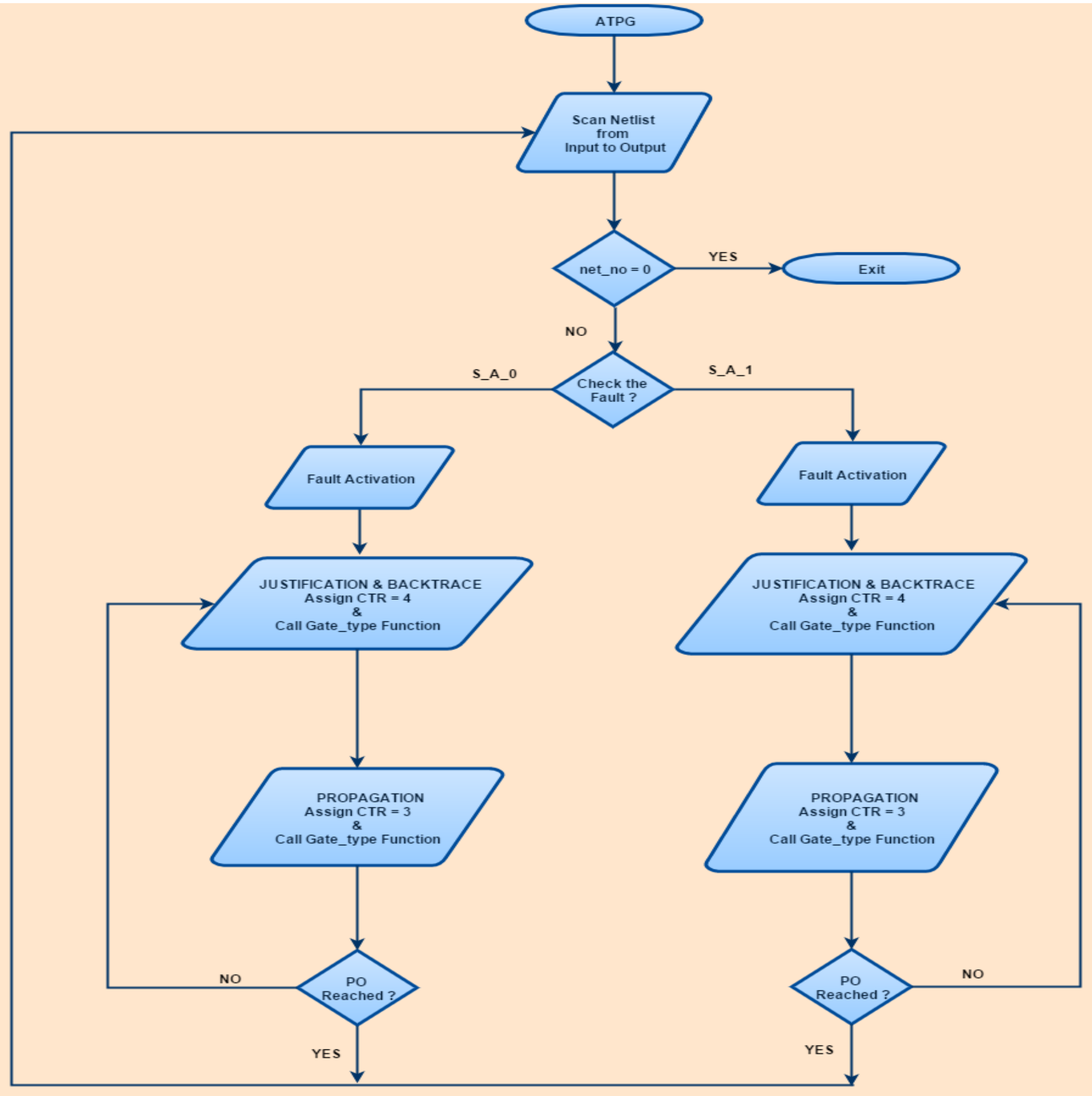

ATPG
Scan Netlist from Input to Output
net_no = 0
YES
Exit
NO
Check the Fault ?
S_A_0
S_A_1
Fault Activation
Fault Activation
JUSTIFICATION & BACKTRACE Assign CTR = 4 & Call Gate_type Function
JUSTIFICATION & BACKTRACE Assign CTR = 4 & Call Gate_type Function
PROPAGATION Assign CTR = 3 & Call Gate_type Function
PROPAGATION Assign CTR = 3 & Call Gate_type Function
NO
PO Reached ?
YES
PO Reached ?
NO
YES

Figure 7.6: ATPG Flow

Chapter 8

Implementation of Probability base ATPG

Succesfully implemented Probability base ATPG algorithm on C17 benchmark circuit and many other circuit, Flow chart and Result are given in following section. It Consists of Following Module

1. Scanning ISCAS Netlist
2. Fault Equivalence
3. Controllability
4. Observability
5. Probability
6. Shorting of Probability
7. Fault List According to Shorting of Net Probability Value
8. Selection of S_A_1 or S_A_0 Fault
9. Fault Activation
10. Fault Propagation
11. line Justification and Back-tracing
12. Generation of Pattern

8.1 Flowchart of Probability base ATPG

Flowchart for any Netlist starting from scanning of netlist to Generation of Pattern.

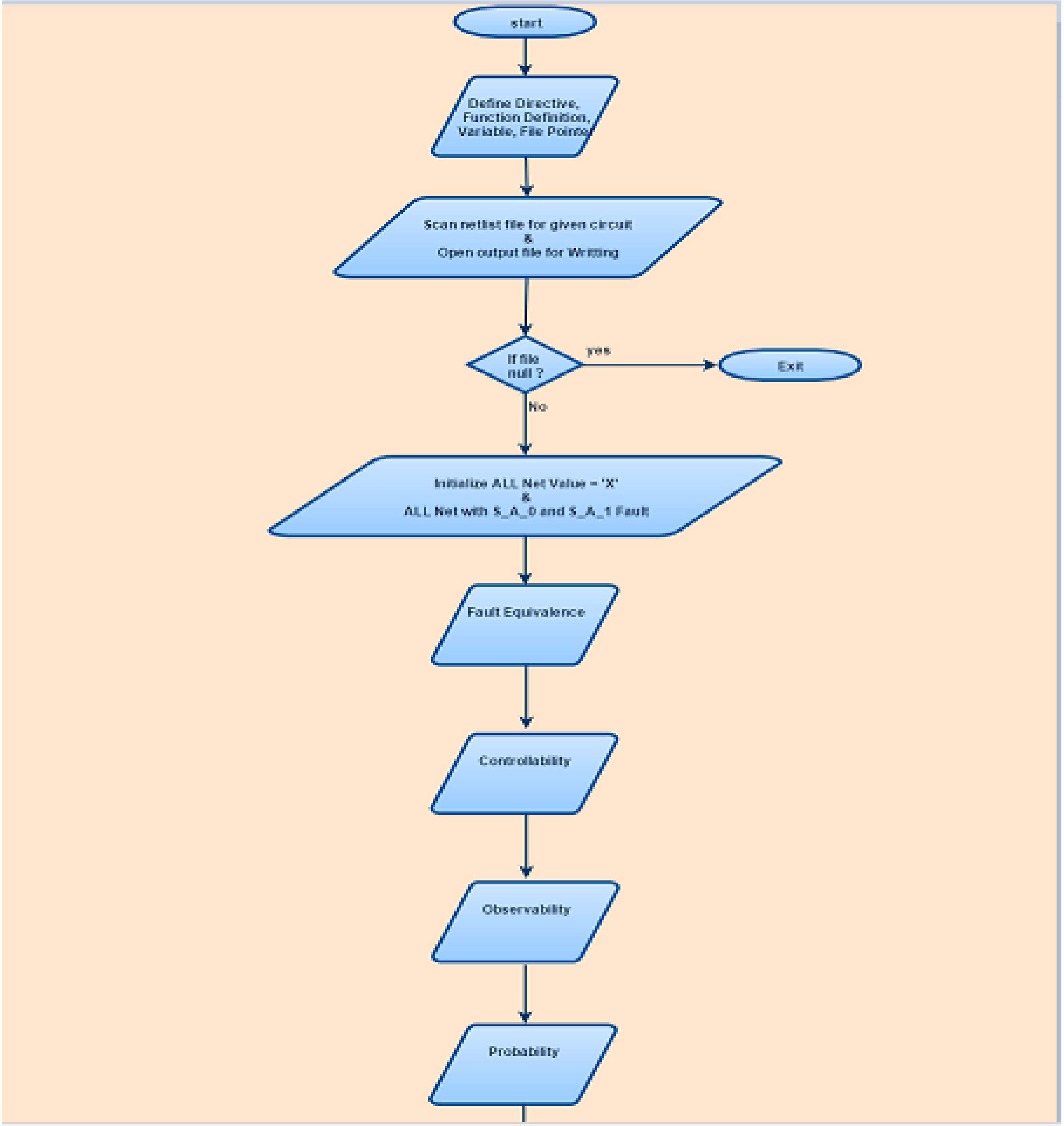


Figure 8.1: Probability base ATPG Flow

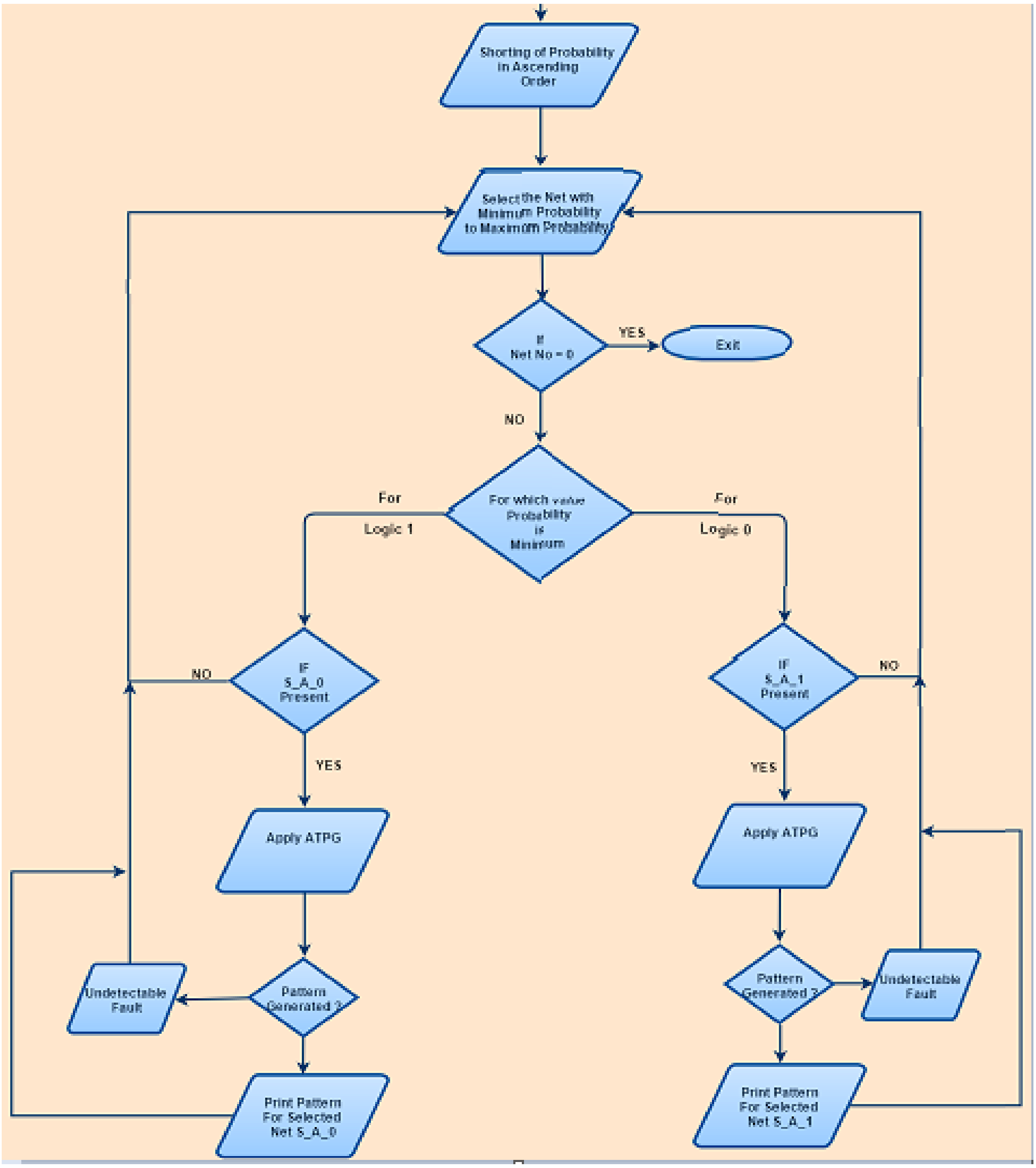

Shorting of Probability in Ascending Order
Select the Net with Minimum Probability to Maximum Probability
If Net No = 0
YES
Exit
NO
For which value Probability is Minimum
For Logic 1
For Logic 0
IF S_A_0 Present
NO
YES
Apply ATPG
Pattern Generated ?
Undetectable Fault
Print Pattern For Selected Net S_A_0
IF S_A_1 Present
NO
YES
Apply ATPG
Pattern Generated ?
Undetectable Fault
Print Pattern For Selected Net S_A_1

Figure 8.2: Probability base ATPG Flow

8.2 result

This are the final test Pattern Generated for Probability base ATPG algorithm for C17 Benchmark circuit.

- NET10 —— S_A_1
  input test pattern = 1 0 1 x x
- NET11 —— S_A_1
  input test pattern = 0 1 1 1 x
- NET14 —— S_A_1
  input test pattern = 0 1 1 1 x
- NET15 —— S_A_1
  input test pattern = x x 1 1 1
- NET16 —— S_A_1
  input test pattern = x 1 0 x x
- NET19 —— S_A_1
  input test pattern = x 0 0 x 1
- NET20 —— S_A_1
  input test pattern = x 1 0 x x
- NET21 —— S_A_1
  input test pattern = x 1 0 x x
- NET23 —— S_A_1
  input test pattern = x 0 x x 0
- NET22 —— S_A_1
  input test pattern = 0 0 x x x
- NET9 —— S_A_1
  input test pattern = x 1 0 1 x
- NET1 —— S_A_1
  input test pattern = D 0 1 x x

- NET2 —— S_A_1
  input test pattern = x D 0 x x

- NET3 —— S_A_1
  input test pattern = 1 0 D x x

- NET3 —— S_A_0
  input test pattern = 1 0 d x x

- NET6 —— S_A_1
  input test pattern = 0 1 1 D x

- NET7 —— S_A_1
  input test pattern = x 0 0 x D

- NET8 —— S_A_1
  input test pattern = 1 0 0 x x

- NET22 —— S_A_0
  input test pattern = 1 x 1 x x

- NET23 —— S_A_0
  input test pattern = x 1 0 x x

- NET16 —— S_A_0
  input test pattern = 0 0 x x x

- NET11 —— S_A_0
  input test pattern = x 1 0 x x

Chapter 9

Conclusion

Developed C Program for Generic ATPG algorithm which can effectively apply Fault equivalence, Find testability measure for given circuit and use it to effectively guide ATPG algorithm. Probability measure use to find undetectable fault very early in test pattern generation.

Successfully implemented PODEM FAN algorithm in C programming language. And Observed difference between this two algorithm in term of Fault Propagation, line-justification, search space. Also added Probability in above algorithm to find undetectable fault at very early stage of test pattern generation.

References

[1] M.L. Bushnell, V.D. Agrawal, *“ Essentials of Electronics Testing for Digital, Memory Mixed Signal VLSI Circuits"*, Kluwer Academic Publishers, Boston MA, 2000

[2] Abramovici, Melvin A. Breuer, and Arthur D. Friedman, *"Digital Systems Testing and Testable Design"*, Jaico Publication.

[3] P. Goel, *“An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits”*, IEEE transactions on Computers, Vol. C-30, no. 3, pp. 215-222, 1981

[4] H. Fujiwara and T. Shimono, *"On the Acceleration of Test Generation Algorithms"*, IEEE Trans. Computers, pp.1137 -1144, 1983.

[5] Vaishali H. Dhare, Usha Mehta, *“Development of Controllability Observability Aided Combinational ATPG with Fault Reduction”*, First International Workshop on VLSI Design 2010, Chennai , July2010, Pg. No 682-692

[6] Vaishali H. Dhare, Usha Mehta, *“Implementation of Compaction Algorithm for ATPG Generated Partially Specified Test Data”*, International Journal of VLSI design Communication Systems, Pg.no.93-103,Vol.4, No.1, February 2013.

[7] `nptel.ac.in/courses/106103116/handout/mod9.pdf`