AN ARITHMETIC APPROACH TO THE ANALYSIS OF MULTIPLE FAULTS VERIFICATION AND SYNTHESIS AT SWITCH-LEVEL

TK 7871.99 .M44 J38 2006

By

Mohammad Reza Samadpour Javaheri

Bachelor of Computer Engineering

Azad University of Tehran,

Tehran, Iran, 1997

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2006

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Signature

ABSTRACT

Thesis Title:

An Arithmetic Approach to the Analysis of Multiple Faults

Verification and Synthesis at Switch-Level

Thesis submitted by:

Mohammad Reza Samadpour Javaheri

Optimization Problems Research and Applications Laboratory (OPR-AL)

ELCE, Master of Applied Science, Ryerson University 2006

Thesis Directed by:

Dr. Reza Sedaghat

Electrical and Computer Engineering Department

Ryerson University

Switch-level modeling and simulation has become an important method for predicting the behaviour of CMOS circuits under the presence of faults. Many important phenomena in CMOS circuits, such as bi-directional signal propagation, charge sharing and variations in driving strength can be reliably modeled using this technique. This paper presents an algorithm for modeling directional and bi-directional CMOS circuits with an arithmetic solution for circuit verification and fault synthesis. This new approach is capable of simulating multiple fault injection into the circuit and speeds up switch-level simulation. Other advantages of this algorithm are its application in the mapping of single and multiple faults from switch level to gate level and the ability to function as a multi-level model. Multiple faults can be of the same or different types. Experimental results using Cadence tools show that the algorithm is successful and reliable for CMOS technology.

ACKNOWLEDGEMENT

I am deeply indebted to my advisor, Dr. Reza Sedaghat, for his constant support. Without his help, this work would not be possible. I would also like to thank the members of our OPR-AL lab who braved the *storm of the century* to help me.

I would like to thank my family for their support. Above all, I cannot express my full gratitude to my parents, who patiently advised me throughout my whole life.

I dedicate this thesis to my mother and father.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

0	Logic zero
1	Logic one
Z	High impedance
U	Unknown
F0	Forcing zero
F1	Forcing one
∇	Connection node symbol
P	P-channel transistor function
N	N-channel transistor function
G	CMOS Gate
S	CMOS Source
D	CMOS Drain
I	Input value for bi-directional functions
O	Output value for bi-directional functions
Y	Circuit's output
MFS	Multiple fault simulation software

CHAPTER 1
INTRODUCTION

1.1 Overview

Testing is an integral component of the VLSI design process[1][4][8][10][19][20] . In VLSI circuits, the set of possible faults is very large. Thus, the need for cost effective fault simulation techniques increases with the size of the circuit under test. Traditional fault simulation has been carried out at the logic gate description level, where classical node stuck-at-0, stuck-at-1 and stuck-at-open [1][22] fault models are considered. However, it has been observed that the classical stuck-at-0 and stuck-at-1 fault models may not be sufficient to represent the effect of physical defects on the behaviour of the transistor circuit design. It should be mentioned, however, that switch-level fault injection simulation is more time consuming than gate level fault injection [1][2][9]. This report introduces an experimental arithmetic switch-level fault synthesis algorithm for both combinational and sequential circuits capable of simulating extremely large designs that is more accurate than at gate level [3] and can be used both for design verification [2][9] and fault simulation [7][12][13]. Single and multiple fault models are supported since it is possible to incorporate any fault model with logical effects. It is possible to model the entire circuit at the switch level. The functionality of this algorithm has been verified with the Cadence Spectra simulator. The algorithm is based on the behaviour of CMOS transistors in digital circuits [24]. An equation is assigned for a whole path in a circuit to describe circuit behaviour in detail based on all input combinations. To generate the equation, the algorithm represents the circuit as a graph. For bigger circuits it is possible to use partitioning methods to reduce complexity.

The unique aspect of this method is that multiple physical faults [1][5][6] are simply injected into the circuit and consequently the results can be seen in the equation. This algorithm can detect all types of faults, including stuck-at -1, stuck-at-0, stuck-at-on, stuck-at-off, bridging [25] and open circuit faults. A unique aspect of the algorithm is its handling of multiple fault injection into the circuit with similar or different types of faults. Furthermore, it can easily map transistor level faults to gate level [11][14][18][23] using a truth table to demonstrate all the intermediate values in the circuit. All values are generated by the equation. In most cases, the library based approach to fault simulation is used to map switch level faults to gate level while covering only single fault injection and limited fault models.

1.2 Bi-directional behaviour of this algorithm

To improve the diagnostability and testing [1][4][8][10] of deep sub-micron high performance VLSI circuits, the need to incorporate more accurate defect models in traditional test generation schemes has become accentuated. Faults such as bridging and transistor stuck-on [1] are known to be representative of a sizeable class of physical failures occurring in CMOS circuits. Whereas some of these faults can be modeled by stuck at behaviour, electrical-level analysis is necessary to handle the majority of these faults. Since a CMOS device is bi-directional in nature, information is considered to flow through it in both directions. This usually requires two separate steps (one for each direction) to express the interaction across a single transistor, with its source and drain switching roles between steps. In this approach we simply connect VDD to drain and GND to source.

1.3 Summary of Contributions

The objective of the thesis report is to emulate many important features of switch-level models, such as bi-directional signal propagation and variations in driving strength and multiple fault injection and test synthesis for unidirectional and bi-directional CMOS VLSI design. The overall research contributions of this thesis are summarised as follows:

- Development of an arithmetic algorithm capable of addressing all possible physical problems in switch level for CMOS technology-based circuits of any size.
- Analysis of different types of multiple faults. Multiple faults were defined as any combination of bridging, stuck-at-fault and open-wire faults occurring simultaneously in the circuit.
- It was confirmed from the result that multiple fault injection into the circuit speeds up switch-level simulation.

- The functionality of the algorithm was checked with the Cadence Spectra simulator. The experiment was completed for all CMOS gates at transistor level.
- Multiple Fault Simulation (MFS) software was developed and run on several ISCAS85 and ISCAS89 benchmark circuits.
- An algorithm is designed capable of mapping faults from a transistor level model to a gate level model.
- This algorithm has a unique ability to propagate delay faults as a constant defect at output which is the most challenging bottleneck in latest technology. This method is fundamentally different from all existing methods.

1.4 Organization of Thesis

In Section 2 fundamental definitions and logical values for CMOS transistors are presented. Section 3 demonstrates the circuit graph model for both directional and bi-directional circuits and also how to find a dominant value from a lookup table when two signals meet each other in the connection node of a CMOS transistor. Section 4 explains an arithmetic model with its equation and truth table. Section 5 presents single fault simulation. Multiple fault simulation is covered in Section 6. The experimental evaluation of the presented algorithm is discussed in Section 7. Section 8 describes the ability of the presented algorithm to propagate delay faults in future works. Section 9 concludes this paper, followed by a list of cited publications and references.

CHAPTER 2
FUNDAMENTAL DEFINITIONS

2.1 Fault models

In this section fundamental definitions for switch level are presented, which are necessary and sufficient to describe electrical behaviour of stuck-open/stuck-on faults (switch level) on the gate level. Some of these fault models are well known. In addition some new fault models will be defined, which are partly based on an extension of existing gate level fault models [15]. All of these gate level fault models can be treated efficiently by a four valued gate level fault simulator. A prototype of a gate level fault simulator (COMSIM, also used in [29]) for non-classical faults has been implemented. Most of the non-classical faults can be mapped onto the classical stuck-at faults [30], bridging faults and transition faults. The bridging fault model has been introduced in [30]. For bipolar technologies wired-OR and wired-AND type are sufficient. The wired-OR is equivalent to a bridging "1" and the wired-AND equivalent to a bridging "0". For CMOS technologies these types of bridging faults are not sufficient. When a resulting voltage between two shorted nodes can neither be interpreted as a logical "1" nor as a logical "0", this fault is defined as a bridging "unknown". This type of bridging fault is detected by a pattern, which detects both the bridging "0" and bridging "1". The transition fault is a special kind of a delay fault. It is based on the assumption that a transition at a gate never occurs in a combinational circuit. In a sequential circuit transition does not occur within a clock cycle.

2.2 Function Conversion

Defects inside a CMOS gate may result in the incorrect function of the gate. For example, with a short between input and output of an inverter the gate is no longer inverting. Furthermore, conversion of an AND to a NAND, OR to NOR etc. are possible. Although every modification inside the logic table can be considered as a function conversion, our definition is limited to simple Boolean function conversions [15].

2.3 General Method

The idea of our method is to combine the accuracy of modeling faults on the electrical or switch level with the efficiency of fault simulation and test pattern generation based on a gate level description of a circuit. The starting point for our investigations is an electrical or switch level description of a standard gate. An appropriate fault model was selected which is assumed to be accurate enough to model realistic faults [15], e.g. the hard fault model (shorts and opens) on the electrical level. For each element in the circuit the effect of each fault was determined in

Cadence tools. Usually this was done using analog simulations. This fault list accurately reflects the electrical or switch level faults which have been defined as target faults.

2.4 Logical values

The switch-level is an abstraction level that resides between the gate-level and the electrical-level and offers many advantages. Switch-level models can reliably model many important phenomena in CMOS circuits, such as bi-directional signal propagation, charge sharing and variations in driving strength [16][21]

On the switch-level, transistors are viewed as switches and the circuit state is described by discrete values, consisting of a logic state (e.g. 0, 1, U - unknown, Z - high impedance) and strength (e.g. weak, strong, medium ...) [12][13]. Various strength levels may be used in switch-level models. In some methods, only two levels of strengths are used (e.g. weak and strong) [12], while others use more levels. For example, the hardware description language Verilog supports switch-level modeling with eight different levels of strength [13].

Each transistor switch is a bi-directional element, which is controlled by the gate terminal G. A point at which two or more transistors have a common connection is called a node. If a node is driven with the driver signals SI, S2... Sn, the value (i.e. the logic state and the strength) of the node is equal to the value of the strongest driver signal. If two driver signals, with the same strengths and different logic states have the highest output strength, the logic state of the node will be U (unknown) [12][13][16][21].

In some methods, resistive switches are used. These switches reduce the strength of signals that propagate through them. For instance, in Verilog HDL both resistive and non-resistive switches can be used [13].

In CMOS technology [24] the basic components at switch-level are transistors. N-channel and P-channel transistors (Figure 1) can receive different logical values on their pins. These logical values are '0', '1', 'F1', 'F0', 'Z', and 'U' [7][12][13]. Logic value 'F1' or Forcing 1 represents power source, 'F0' or Forcing 0 represents ground, 'Z' represents the state of an isolated or floating connector and may be interpreted as the high impedance, and, finally, 'U' represents an intermediate voltage level, which occurs when '0' and '1' signals are applied simultaneously to a connector and may be interpreted as an unknown signal.

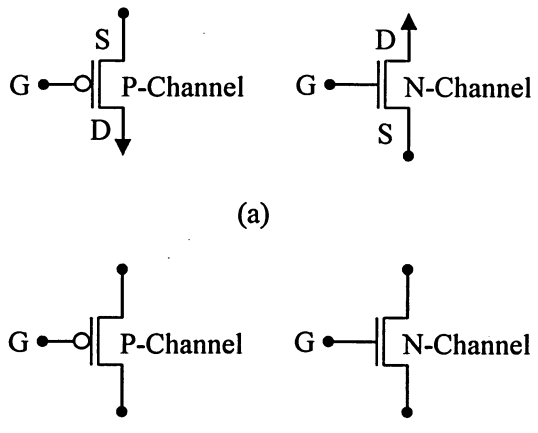

Figure 1. (a) Normal behaviour (b) Switching behaviour or bi-directional

Logic values 'F1' and 'F0' are always in Forcing 1 and Forcing 0, but the logic value of '1' and '0' can be Weak 1 and Weak 0 especially when there is fan-out.

Gate	Input	Output	
G	I	P(G,I)	N(G,I)
0	0	0	Z
0	1	1	Z
0	Z	Z	Z
0	F1	F1	Z
0	F0	F0	Z
0	U	U	Z
1	0	Z	0
1	1	Z	1
1	Z	Z	Z
1	F1	Z	F1
1	F0	Z	F0
1	U	Z	U
Z	0	U	U
Z	1	U	U
Z	Z	U	U
Z	F1	U	U
Z	F0	U	U
Z	U	U	U
F1	0	Z	0
F1	1	Z	1
F1	Z	Z	Z
F1	F1	Z	F1
F1	F0	Z	F0
F1	U	Z	U
F0	0	0	Z
F0	1	1	Z
F0	Z	Z	Z
F0	F1	F1	Z
F0	F0	F0	Z
F0	U	U	Z
U	0	U	U
U	1	U	U
U	Z	U	U
U	F1	U	U
U	F0	U	U
U	U	U	U

Table 1. Bi-directional behaviour lookup table for CMOS

Table 1 displays all the possible states of a transistor in a digital circuit. Later we will see how to use this look-up table to find the output of our functions.

2.5 Connection node

An important subject in digital circuits is networks and connections. A connection node (Figure 2) defines where two or more signals meet each other [14] and generate a network. In Figure 2 for example, the drain of a P1 transistor is connected to the drain of an N2 transistor each with its own logical value. The outcome is a dominant value for this connection. For instance, if the drain of the P1 transistor has the value 'F1' and the drain of N2 transistor has the value 'Z' then the dominant value 'F1' is considered for the connection node, which is symbolized as '∇'. Table 2 shows the dominant logic value for connection nodes.

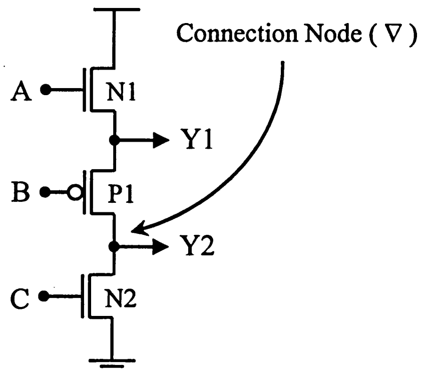

Figure 2. CMOS Circuit

In Figure 2 transistor P1 shows the bi-directional functionality of a CMOS transistor. This transistor can either propagate the ground signal to output Y1 or the VDD signal to output Y2. The arithmetic equation for Y1 and Y2 outputs are shown in Equation (EQ-2-1).

$$Y1 = N1\ (A, F1)\ \nabla\ P1\ (B, (N2\ (C, F0)\ \nabla\ Y2))$$

$$Y2 = N2\ (C, F0)\ \nabla\ P1\ (B, (N1\ (A, F1)\ \nabla\ Y1)) \qquad \text{(EQ-2-1)}$$

Section 4 will discuss in detail the arithmetic model to generate these equations for any CMOS circuit.

∇	**0**	**1**	**Z**	**F1**	**F0**
0	0	U	0	F1	F0
1	U	1	1	F1	F0
Z	0	1	Z	F1	F0
F1	F1	F1	F1	F1	U
F0	F0	F0	F0	U	F0

Table 2. Connection node equivalent value lookup table

CHAPTER 3
CIRCUIT GRAPH MODEL

With an increase in circuit size it becomes more complex to analyze and observe its behaviour especially when there is fault in the circuit [1][4][8][10][19]. In our algorithm a graph model converts a transistor level model to a graph and is independent of circuit complexity.

3.1 Bi-directional graph model

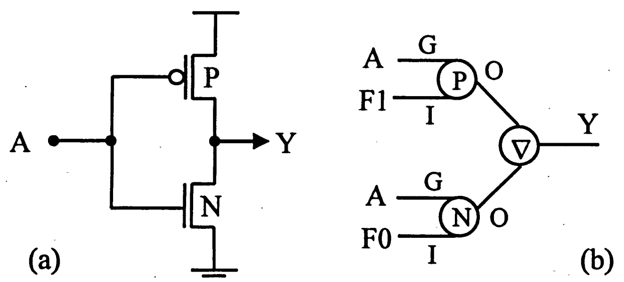

Figure 3. (a) NOT gate, (b) NOT gate bi-directional graph

This graph model simplifies the use of the algorithm and fault injection. [5][6]. In this graph nodes represent a transistor or a connection and the edges represent wires. Figure 3 (a) shows a NOT gate circuit with (b) its related graph. In this graph P represents a P-type transistor, 'N' represents an N-type transistor and '∇' shows connection node. 'A' is the primary input signal, 'Y' the primary output signal, 'F1' represents the power source, and 'F0' represents ground. 'P' and 'N' are transistors with 'G', 'I' and 'O' edges that represent gate, input and output, respectively, for transistors.

3.2 Directional graph model

The Directional graph model is similar to bi-directional graph. However there is a distinction between source and drain as shown in Figure 4 (b).

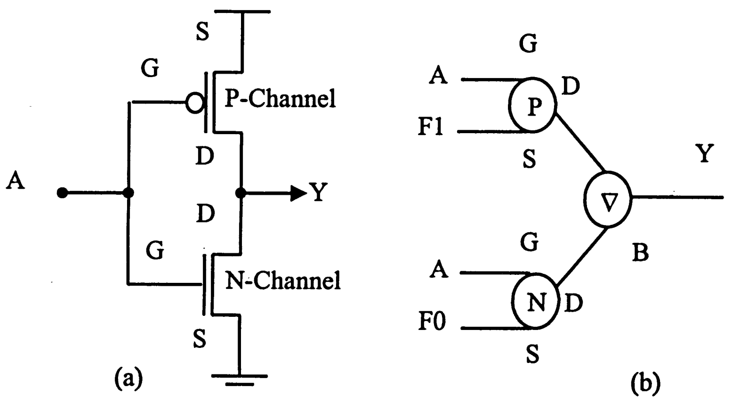

Figure 4. (a) NOT gate, (b) NOT gate directional graph

In this graph 'A' is the primary input signal, 'Y' the primary output signal, 'F1' represents the power source and 'F0' represents ground. 'P' and 'N' are transistors with 'G', 'S' and 'D' edges that represent gate, source and drain, respectively, for transistors.

CHAPTER 4
ARITHMETIC MODEL AND EQUATION

4.1 Function definitions

In this arithmetic model each transistor is considered a function such as ‘P’ and ‘N’. Each has two arguments as inputs and a returning logical value that is considered for output. P (G, I), N (G, I) are the syntaxes for our functions. The ‘P’ function is used for P-type transistors and ‘N’ function is used for N-type transistors. The first argument or ‘G’ is the value of gate and the second argument or ‘I’ is the value of input for each transistor. To handle bi-directional signals I can be considered either for source (S) or for drain (D). The return value of these functions will be considered as output (O). The result of the function is analyzed with the logic value at the output. The value of each function can be taken from Table 1. For example, given the logical values G = 0 and I = F1 then, according to Table 1, these functions will be P (0, F1) = F1 and N (0, F1) = Z. In Figure 3 (b) the outputs of the two functions collide at a connection node. This is represented in the following circuit equation:

$$Y = P\,(G, I)\ \nabla\ N\,(G, I) \qquad \text{(EQ-4-1)}$$

The symbol ∇ in equation (EQ-4-1) represents the connection point of two signals [14]. The actual input values for the function according to Figure 3 (b) should be replaced in the equation (EQ-4-1).

$$Y = P\,(A, F1)\ \nabla\ N\,(A, F0) \qquad \text{(EQ-4-2)}$$

The primary output 'Y' can be calculated for different values of the primary input 'A' using equation (EQ-4-2). Table 1 is a lookup table to return the equivalent value for each function. Table 2 is a lookup table to return the value for each connection node.

For each value of primary input 'A' the primary output 'Y' for Figure 3 is calculated as:

$$A = 0,\quad Y = P\,(0, F1)\ \nabla\ N\,(0, F0) = F1\ \nabla\ Z = F1$$

$$A = 1,\quad Y = P\,(1, F1)\ \nabla\ N\,(1, F0) = Z\ \nabla\ F0 = F0$$

Table 3 displays the truth table for the circuit equation. 'A' is the primary input for the NOT gate, 'P' is the output of the function 'P' and 'N' is the output of the function 'N'. The first 'Y' in column 4 of the table is the equivalent value for the connection node at switch-level and the second 'Y' in column 5 (greyed) is the output of the gate at gate level. Although both 'Y' values are equal, the second 'Y' indicates gate behaviour.

A	P	N	Y	Y
0	F1	Z	F1	1
1	Z	F0	F0	0

Table 3. NOT truth table

The greyed columns above correspond to primary input and output and represent gate or system level behaviour for the circuit

4.2 Function definitions for directed signals

Each function is considered as a directed switch. Thus the input signal always goes through source (S) and the output will be drive from drain (D). Except for some slight changes in parameters, the function definition for the algorithm will remain the same. As discussed in the previous section, both functions ‘P’ and ‘N’ have two arguments as inputs and a returning logical value that is considered for Drain pin. P (G, S), N (G, S) are the models for our functions. The ‘P’ function is used for P-type transistors and ‘N’ function is used for N-type transistors. The first argument or ‘G’ is the value of Gate and the second argument or ‘S’ is the value of Source for each transistor. The result of the function is analyzed with the logic value at the drain. The value of each function can be derived from Table 4. For example, given the logical values G = 0 and S = F1 then, according to Table 4, these functions will be P (0, F1) = F1 and N (0, F1) = Z. In Figure 4 (b) the outputs of two functions collide at a connection node. This is represented in the following circuit equation:

$$Y = P(G, S) \nabla N(G, S) \qquad \text{(EQ-4-3)}$$

Gate	Source	Drain P(G, S)	Drain N(G, S)
0	0	0	Z
0	1	1	Z
0	Z	Z	Z
0	F1	F1	Z
0	F0	F0	Z
0	U	U	Z
1	0	Z	0
1	1	Z	1
1	Z	Z	Z
1	F1	Z	F1
1	F0	Z	F0
1	U	Z	U
Z	0	U	U
Z	1	U	U
Z	Z	U	U
Z	F1	U	U
Z	F0	U	U
Z	U	U	U
F1	0	Z	0
F1	1	Z	1
F1	Z	Z	Z
F1	F1	Z	F1
F1	F0	Z	F0
F1	U	Z	U
F0	0	0	Z
F0	1	1	Z
F0	Z	Z	Z
F0	F1	F1	Z
F0	F0	F0	Z
F0	U	U	Z
U	0	U	U
U	1	U	U
U	Z	U	U
U	F1	U	U
U	F0	U	U
U	U	U	U

Table 4. Directional behaviour lookup table for CMOS

The symbol ∇ in equation (EQ-4-3) represents the connection point of two signals [16][17][21]. The actual input values for the function according to Figure 3 (b) should be replaced in the equation.

$$Y = P\ (A, F1)\ \nabla\ N\ (A, F0) \qquad \text{(EQ-4-4)}$$

The primary output 'Y' can be calculated for different values of the primary input 'A' using equation (EQ-4-4). The rest of the calculations are the same as in the previous section for bi-directional signals.

4.3 Gate equations

The ISCAS85 and ISCAS89 benchmark circuits have several kinds of gates that are used in this thesis report and have been applied in our simulation software. These gates and their equivalent equations are listed as follows.

4.3.1 NOT

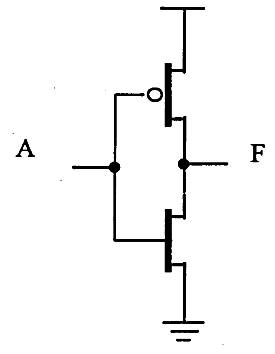

Figure 5. Not Gate

$$f_{not} = P(A, F1) \nabla N(A, F0)$$

4.3.2 NAND2

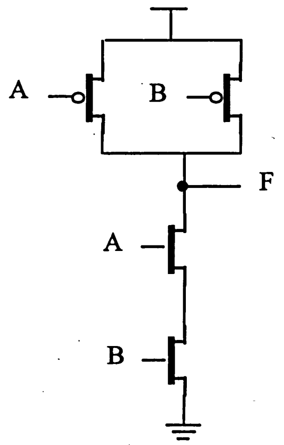

Figure 6. NAND2 Gate

$$f_{nand2} = (P(A, F1) \nabla P(B, F1)) \nabla N(A, N(B, F0))$$

4.3.3 NAND3

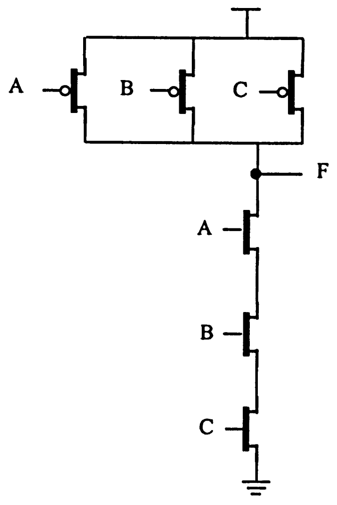

Figure 7. NAND3 Gate

$$f_{nand3} = (P(A, F1) \nabla P(B, F1) \nabla P(C, F1)) \nabla N(A, N(B, N(C, F0)))$$

4.3.4 NAND4

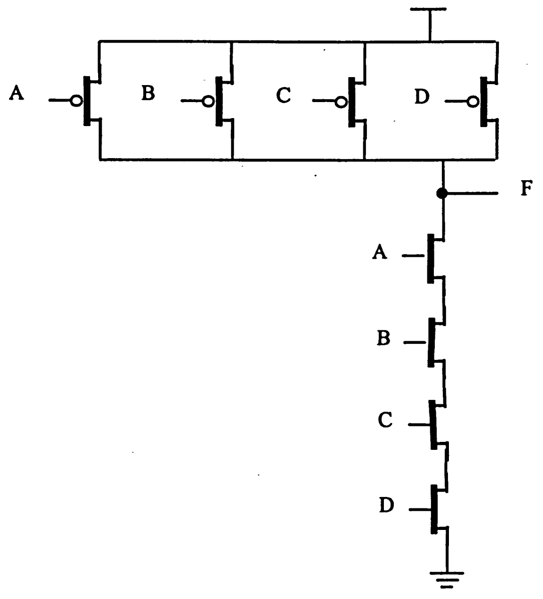

Figure 8. NAND4 Gate

$$f_{nand4} = (P(A, F1) \nabla P(B, F1) \nabla P(C, F1) \nabla P(D, F1)) \nabla N(A, N(B, N(C, N(D, F0))))$$

4.3.5 NAND5

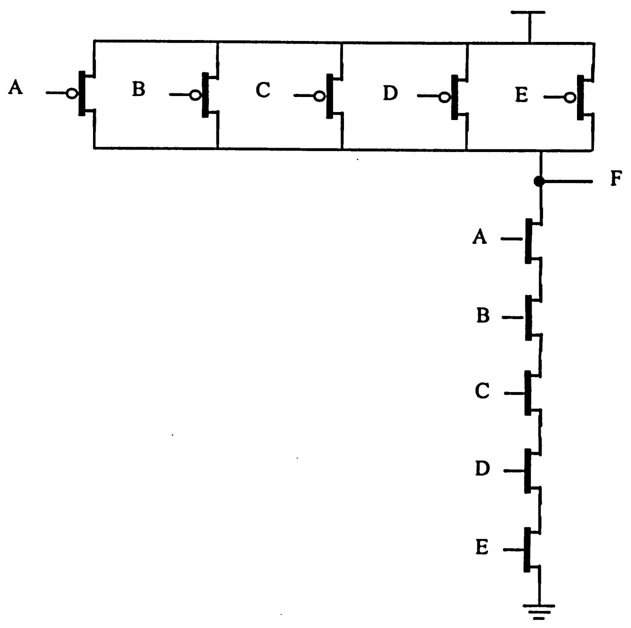

Figure 9. NAND5 Gate

f_{nand5} = (P(A, F1) ∇ P(B, F1) ∇ P(C, F1) ∇ P(D, F1) P(E, F1)) ∇ N(A, N(B, N(C, N(D, N(E, F0)))))

4.3.6 NOR2

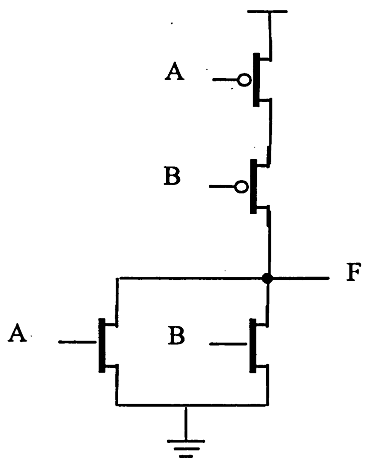

Figure 10. NOR2 Gate

$$f_{nor2} = P(B, P(A, F1)) \nabla (N(A, F0) \nabla N(B, F0))$$

4.3.7 NOR3

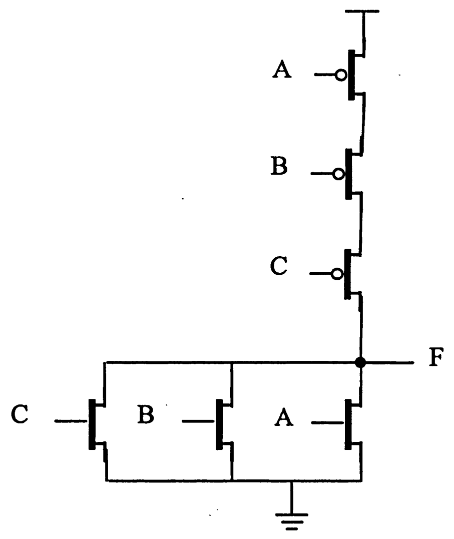

Figure 11. NOR3 Gate

$$f_{nor3} = P(B, P(A, P(C, F1))) \nabla (N(A, F0) \nabla N(B, F0) \nabla N(C, F0))$$

4.3.8 NOR4

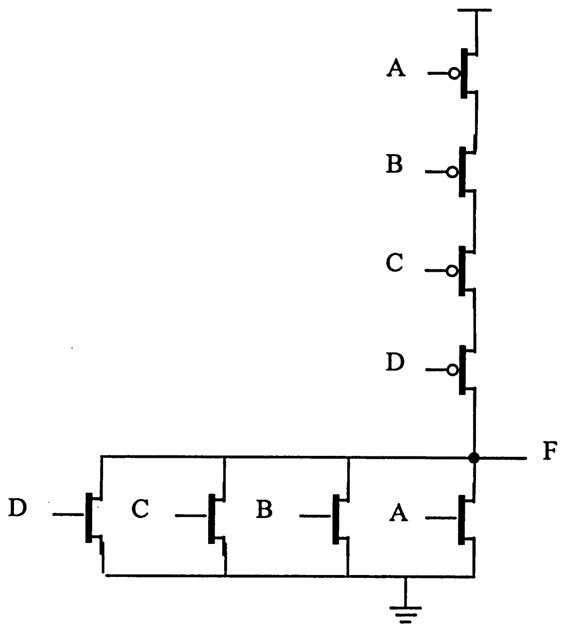

Figure 12. NOR4 Gate

f_{nor4} = P(B, P(A, P(C, P(D, F1)))) ∇ (N(A, F0) ∇ N(B, F0) ∇ N(C, F0) ∇ N(D, F0))

4.3.9 NOR5

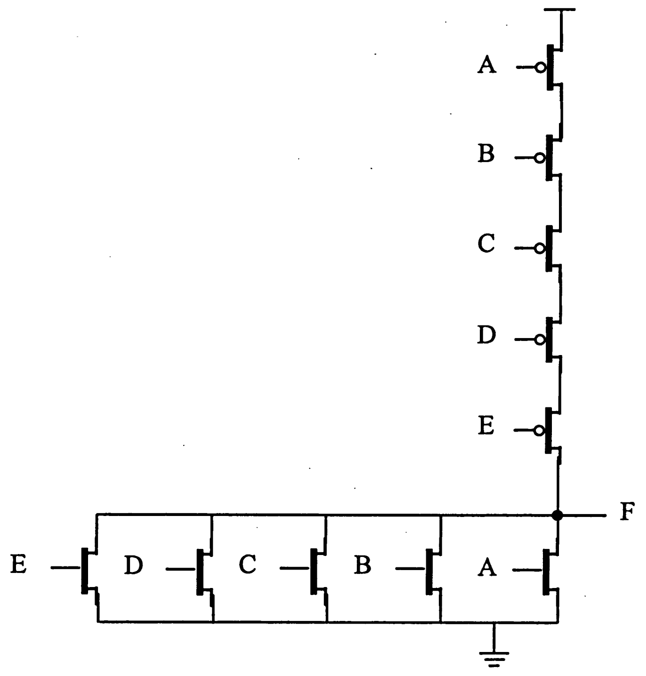

Figure 13. NOR5 Gate

f_{nor5} = P(B, P(A, P(C, P(D, P(E, F1))))) ∇ (N(A, F0) ∇ N(B, F0) ∇ N(C, F0) ∇ N(D, F0) ∇ N(E, F0))

4.3.10 AND2

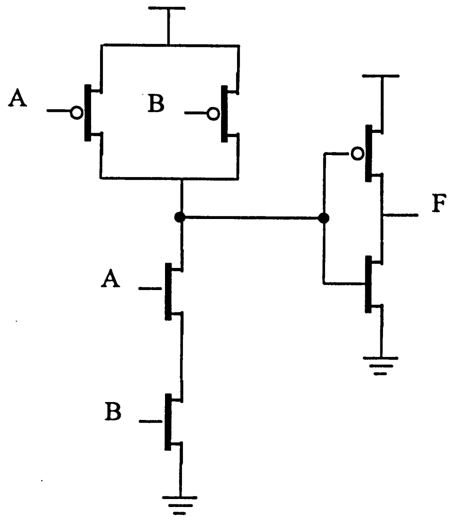

Figure 14. AND2 Gate

$$f_{nand2} = (\mathrm{P}(\mathrm{A}, \mathrm{F1}) \nabla \mathrm{P}(\mathrm{B}, \mathrm{F1})) \nabla \mathrm{N}(\mathrm{A}, \mathrm{N}(\mathrm{B}, \mathrm{F0}))$$

$$f_{and2} = \mathrm{P}(f_{nand2}, \mathrm{F1}) \nabla \mathrm{N}(f_{nand2}, \mathrm{F0})$$

4.3.11 AND3

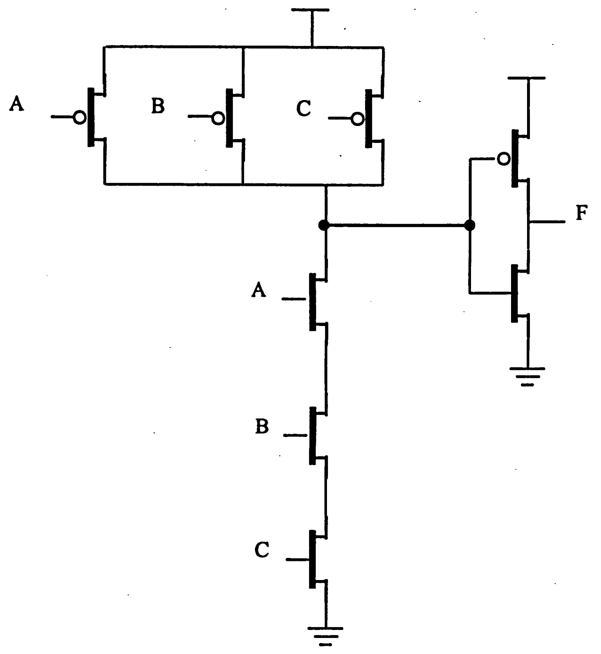

Figure 15. AND3 Gate

$$f_{nand3} = (\mathrm{P(A, F1)} \nabla \mathrm{P(B, F1)} \nabla \mathrm{P(C, F1)}) \nabla \mathrm{N(A, N(B, N(C, F0)))}$$

$$f_{and3} = \mathrm{P}(f_{nand3}, \mathrm{F1}) \nabla \mathrm{N}(f_{nand3}, \mathrm{F0})$$

4.3.12 AND4

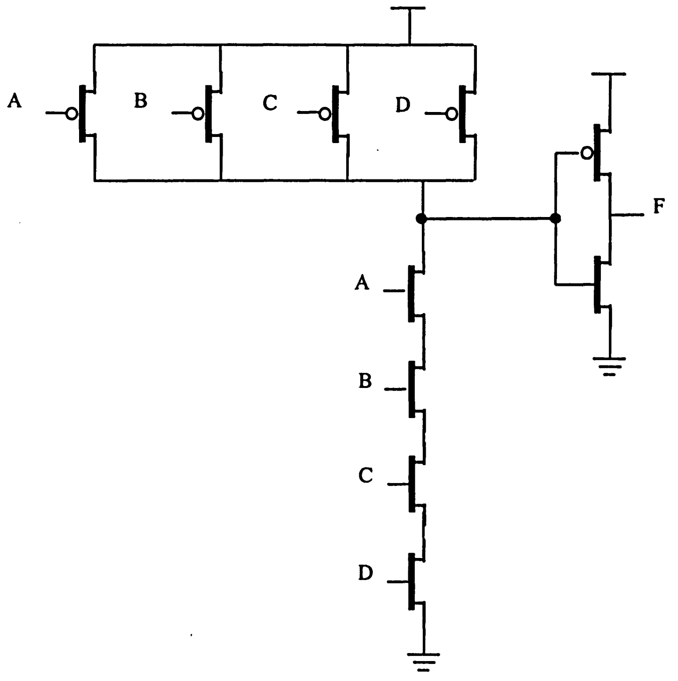

Figure 16. AND4 Gate

$$f_{nand4} = (\mathrm{P}(\mathrm{A}, \mathrm{F1}) \nabla \mathrm{P}(\mathrm{B}, \mathrm{F1}) \nabla \mathrm{P}(\mathrm{C}, \mathrm{F1}) \nabla \mathrm{P}(\mathrm{D}, \mathrm{F1})) \nabla \mathrm{N}(\mathrm{A}, \mathrm{N}(\mathrm{B}, \mathrm{N}(\mathrm{C}, \mathrm{N}(\mathrm{D}, \mathrm{F0}))))$$

$$f_{and4} = \mathrm{P}(f_{nand4}, \mathrm{F1}) \nabla \mathrm{N}(f_{nand4}, \mathrm{F0})$$

4.3.13 AND5

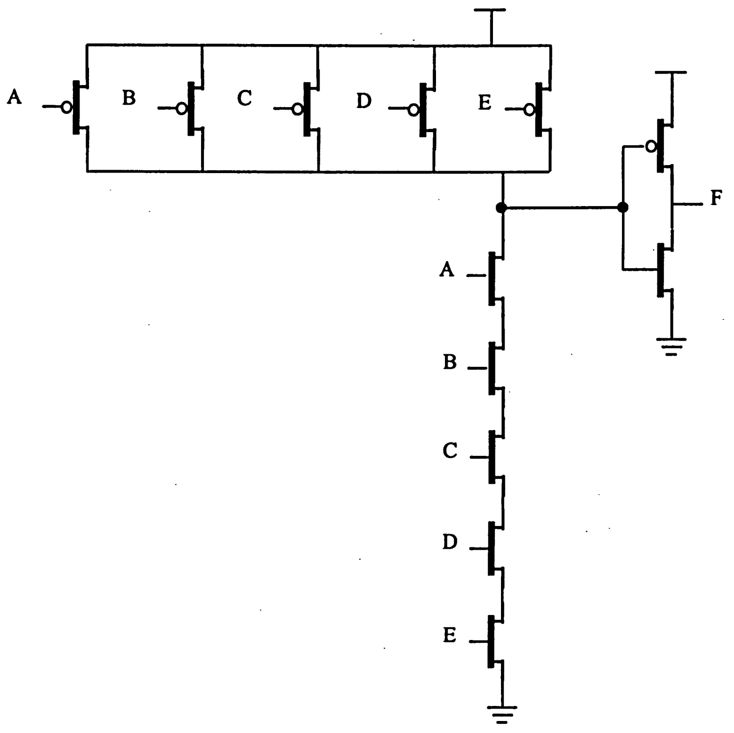

Figure 17. AND5 Gate

f_{nand5} = (P(A, F1) ∇ P(B, F1) ∇ P(C, F1) ∇ P(D, F1) P(E, F1)) ∇ N(A, N(B, N(C, N(D, N(E, F0)))))

f_{and5} = P(f_{nand5}, F1) ∇ N(f_{nand5}, F0)

4.3.14 OR2

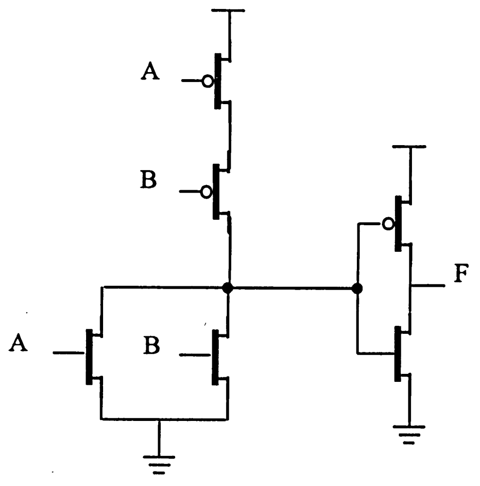

Figure 18. OR2 Gate

$$f_{nor2} = P(B, P(A, F1)) \nabla (N(A, F0) \nabla N(B, F0))$$

$$f_{or2} = P(f_{nor2}, F1) \nabla N(f_{nor2}, F0)$$

4.3.15 OR3

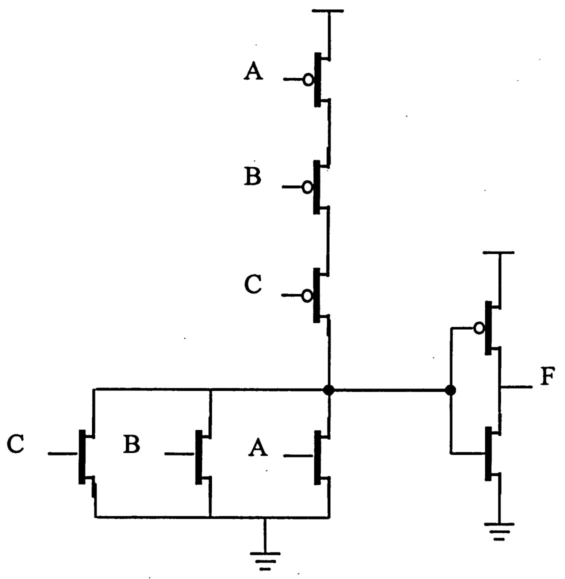

Figure 19. OR3 Gate

$$f_{nor3} = P(B, P(A, P(C, F1))) \nabla (N(A, F0) \nabla N(B, F0) \nabla N(C, F0))$$

$$f_{or3} = P(f_{nor3}, F1) \nabla N(f_{nor3}, F0)$$

4.3.16 OR4

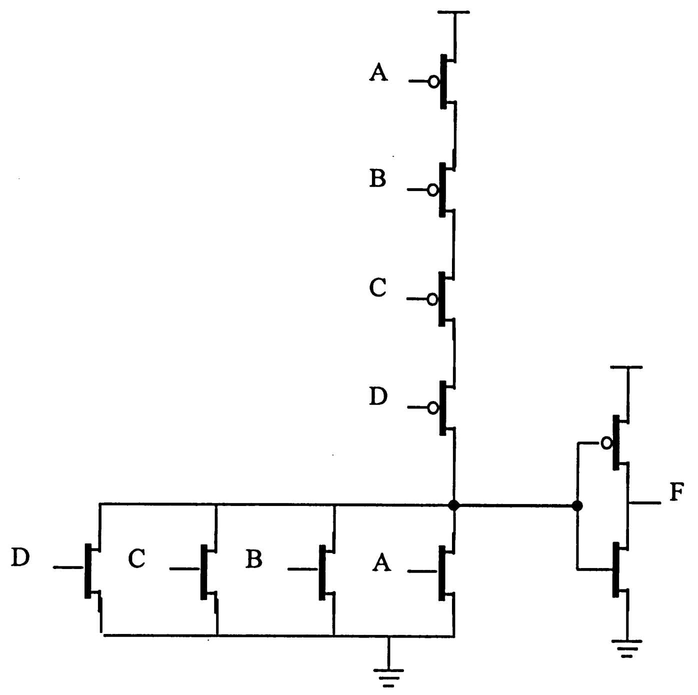

Figure 20. OR4 Gate

$$f_{nor4} = P(B, P(A, P(C, P(D, F1)))) \nabla (N(A, F0) \nabla N(B, F0) \nabla N(C, F0) \nabla N(D, F0))$$

$$f_{or4} = P(f_{nor4}, F1) \nabla N(f_{nor4}, F0)$$

4.3.17 OR5

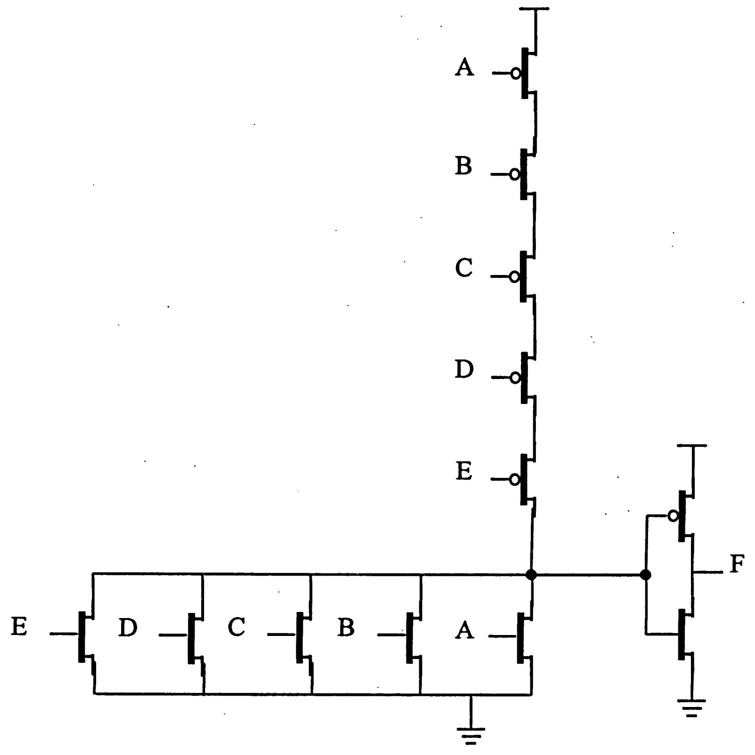

Figure 21. OR5 Gate

f_{nor5} = P(B, P(A, P(C, P(D, P(E, F1))))) ∇ (N(A, F0) ∇ N(B, F0) ∇ N(C, F0) ∇ N(D, F0) ∇ N(E, F0))

f_{or5} = P(f_{nor5}, F1) ∇ N(f_{nor5}, F0)

4.3.18 D-Latch

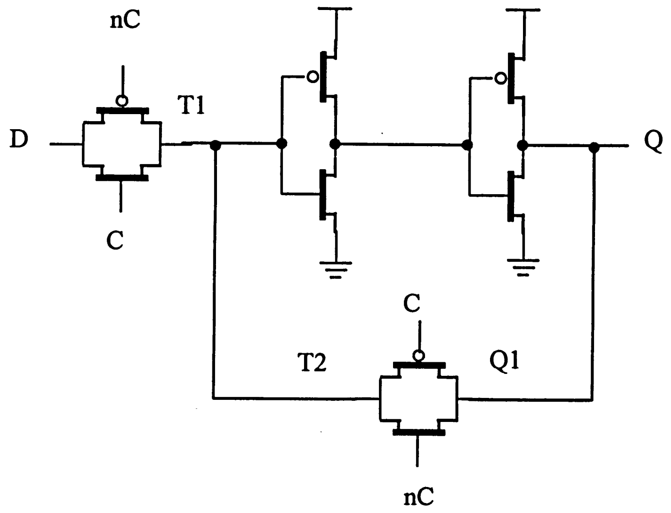

Figure 22. D-Latch

T1 = (P(nC, D) ∇ N(C, D))

Q1 = P(P(T1 , F1) ∇ N(T1, F0), F1) ∇ N(P(T1, F1) ∇ N(T1, F0), F0)

T2 = (P(C, Q1) ∇ N(nC, Q1))

$\boldsymbol{Q_{D\text{-}latch}}$ = P(P((T1 ∇ T2), F1) ∇ N((T1 ∇ T2), F0), F1) ∇ N(P((T1 ∇ T2), F1) ∇ N((T1 ∇ T2), F0), F0)

CHAPTER 5
SINGLE FAULT SIMULATION

The arithmetic equation model can simulate circuit behaviour at switch-level for any input combination. However, if the circuit contains even a single fault the behaviour of the circuit will be affected. We will inject a fault [1][5][6] into the circuit and consequently in the equation to observe circuit behaviour during fault simulation [4][8][10][19][20].

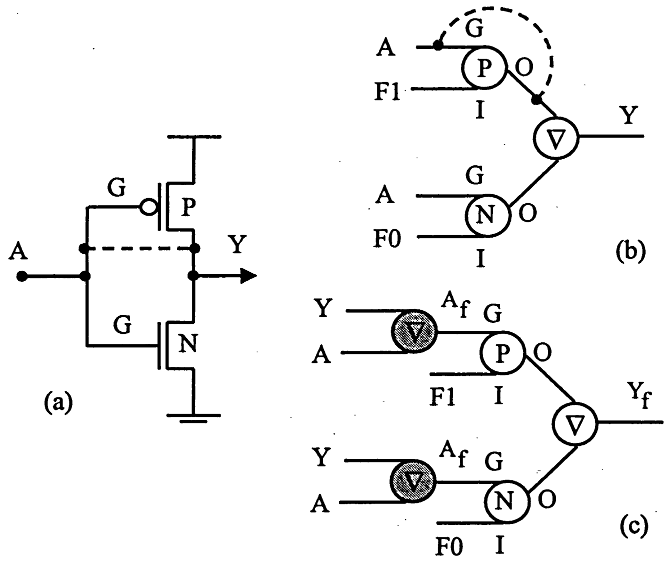

Figure 23. (a) NOT gate circuit with bridging fault (b) Circuit pre-graph with bridging (c) Fault implementation in graph

We will start by considering a single fault, followed by modeling multiple fault simulation in the circuit.

5.1 Bridging fault

Figure 23 (a) shows a NOT gate circuit with a bridging fault [25] between two networks indicated by a dashed line. This bridging fault is displayed in the pre-graph Figure 23 (b) as well. The logical values of all edges that meet in a connection node are equal and have the value of 'Y'. In this algorithm a bridging fault between two networks can be considered as a feedback into the circuit. In Figure 23 (b) the value of connection nodes 'Y' should feed back to the gate of transistor 'P'. To implement this feedback the algorithm must add a connection node to the graph as shown in Figure 23 (c) and thus feed the 'Y' back into the gate of transistor 'P'. Here, 'A' is an input signal for transistor 'N' as well.

According to equation (EQ-4-1), a fault free equation for NOT gate circuit, variable 'G' in function 'P' is connected to input signal 'A'. With this fault the value of signal 'A' will change according to the following equation:

$$A_f = A \nabla Y \qquad \text{(EQ-5-1)}$$

The new value for signal 'A' is calculated with equation (EQ-5-1) and assigned to A_f. Index 'f' refers to a fault in circuit affected input 'A'. To find the final equation for the bridging fault shown in Figure 23, A_f should be replaced with 'A' in equation (EQ-4-2). The result is shown in equation (EQ-5-2):

$$Y_f = P\,(A_f, F1)\ \nabla\ N\,(A_f, F0) \qquad \text{(EQ-5-2)}$$

Table 5 demonstrates that the NOT gate will behave either as a buffer or a wire when there is a bridging fault in the circuit (Figure 23).

A	A_f	P	N	Y_f	Y_f
0	F1	Z	F0	F0	0
1	F0	F1	Z	F1	1

Table 5. Truth table with bridging fault

5.2 Open circuit fault

An open circuit fault results from a metal wire causing a disconnection in a circuit. In some cases, an open circuit wire may have unknown value due to the effects of noise and magnetic fields. However, in most cases, an open circuit is considered as high impedance in a circuit. With this algorithm the whole branch before the open edge is removed and the logical value 'Z' is assigned to that edge. This open circuit fault is represented as stuck-at-z in the equation. In equation (EQ-4-1) the return value for function 'P' is stuck-at-z under all circumstances. Figure 24 shows the graph for an open circuit fault.

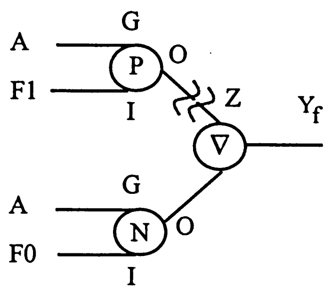

Figure 24. P-type transistor is stuck-at-z

The equation and values are shown in the following equations:

$$Y_f = Z \nabla N(G, I) = Z \nabla N(A, F0)$$

The output value for all the input connections (A=0 or 1) will be calculated as follows:

$$A = 0, \quad Y_f = Z \nabla N(0, F0) = Z \nabla Z = Z$$

$$A = 1, \quad Y_f = Z \nabla N(1, F0) = Z \nabla F0 = F0$$

Table 6 displays all outputs for the above equation. When the input signal for the gate is equal to ‘0’ the output is ‘Z’ and when the input is ‘1’ the circuit functions as a normal inverter.

A	P_f	N	Y_f	Y_f
0	Z	Z	Z	Z
1	Z	F0	F0	0

Table 6. Truth table for open circuit fault

5.3 Stuck-at Fault

Stuck-at-fault simulation is a unique aspect of this algorithm. No changes in the graph or equation are required in order to inject a stuck-at fault. With a stuck-at fault the transistor remains in a stable state in all situations and the return value for 'P' and 'N' functions are independent of input values. If there is a stuck-at fault transistor in the circuit, the return value for its function is taken from Table 7 rather than Table 1.

Type	P – Switch	N – Switch
Stuck-at-0	O = 0	O = 0
Stuck-at-1	O = 1	O = 1
Stuck-at-off	O = Z, G = 1	O = Z, G = 0
Stuck-at-on	O = I, G = 0	O = I, G = 1

Table 7. Stuck-at fault lookup table, O = Output, I = Input and G = Gate

It is assumed that for stuck-at-off the gate for the P-type transistor is stuck-at-1 and for N-type is stuck-at-0. Also assume that for stuck-at-on the gate for the P-type transistor is stuck-at-0 and for N-type is stuck-at-1 [17]. These values are shown in Table 7. The following example explains how to use Table 7 for stuck-at-1 fault.

According to Table 7, a P-Type transistor (Figure 25) is stuck-at-1 and the return value for function P (G, I) in equation (EQ-4-1) is always equal to 1.

$$Y = P(A, F1) \nabla N(A, F0) = 1 \nabla N(A, F0)$$

For each value of 'A' the output 'Y' is calculated as:

$$A = 0, \quad Y = 1 \nabla Z = 1 \qquad A = 1, \quad Y = 1 \nabla F0 = F0$$

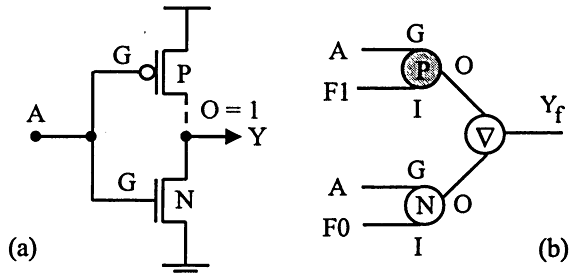

Figure 25. P-type transistor is stuck-at-1

To better demonstrate the functionality of this algorithm further examples with larger circuits follow. Figure 26 illustrates a two input OR gate circuit. For this circuit we will find the circuit equation and then observe the result of fault injection into the circuit.

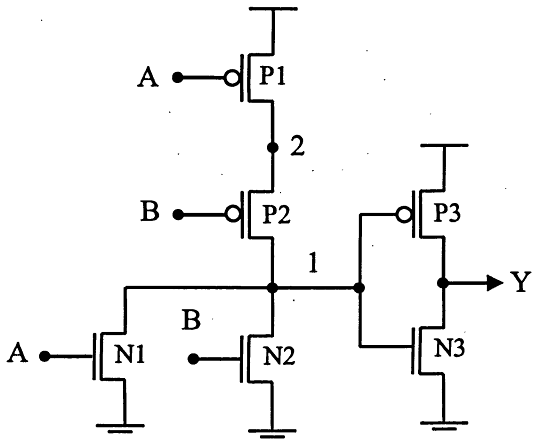

Figure 26. Transistor model of OR gate

In CMOS technology [24] an OR gate circuit is a combination of a NOR and a NOT gate. In Figure 27 below, the output of NOR gate is assigned to $\overline{Y}$ and Y is the primary output for the entire circuit. The circuit is represented in the following equation (EQ-5-3)

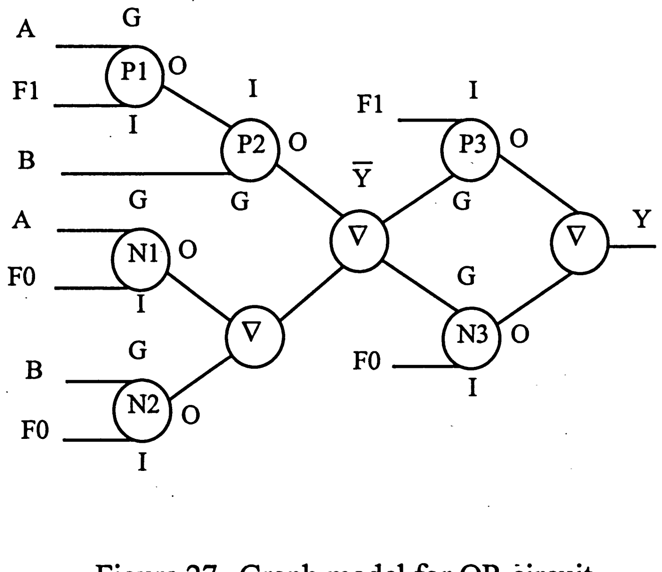

Figure 27. Graph model for OR circuit

$$\overline{Y} = P2\ (B, P1(A, F1)) \nabla [N1(A, F0) \nabla N2(B, F0)]$$

$$Y = P3\ (\overline{Y}, F1) \nabla N3\ (\overline{Y}, F0) \qquad \text{(EQ-5-3)}$$

Figure 26 indicates the behaviour of the above equation for various inputs.

A	B	P1	P2	N1	N2	$\overline{Y}$	P3	N3	Y	Y
0	0	F1	F1	Z	Z	F1	Z	F0	F0	0
0	1	F1	Z	Z	F0	F0	F1	Z	F1	1
1	0	Z	Z	F0	Z	F0	F1	Z	F1	1
1	1	Z	Z	F0	F0	F0	F1	Z	F1	1

Table 8. Truth table for OR circuit equation

We have used the arithmetic model to illustrate the behaviour of the circuit (Figure 26). We will now inject faults [5][6] into the circuit and analyze the equation for all possible inputs. Some single bridging fault examples are shown as follows.

5.4 Bridging between A and 1

This fault causes a connection between two edges in the related pre-graph (Figure 28) indicated by a dashed line. To remove this problem from the graph the algorithm must add a connection point in the graph and feed the signal back from the point with the value $\overline{Y}$ to the gate of transistor 'P1'. The equation is then regenerated with the added nodes.

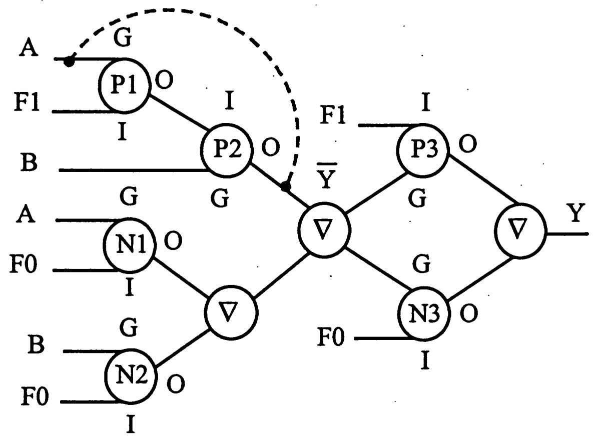

Figure 28. OR circuit pre-graph with bridging fault

Figure 29 displays the changes in the graph with a bridging fault implemented as a feedback.

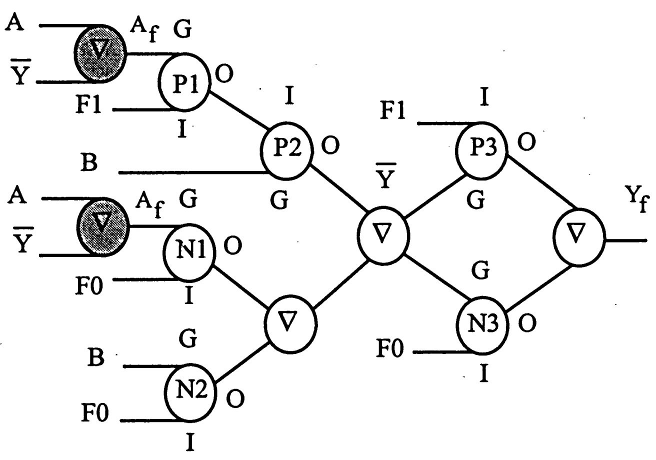

Figure 29. OR graph with bridging fault

In equation (EQ-5-3) value 'A' should be replaced with $A_f = A \nabla \overline{Y}$. The result is reflected in equation (EQ-5-4).

$\overline{Y}_f$ = P2 (B, P1 (A_f, F1)) ∇ [N1 (A_f, F0) ∇ N2 (B, F0)]

Y_f = P3 ($\overline{Y}_f$, F1) ∇ N3 ($\overline{Y}_f$, F0) (EQ-5-4)

Table 9 shows all the primary and intermediate logical values for the circuit graph (Figure 29) with a bridging fault between points 'A' and '1'.

A	B	A_f	P1	P2	N1	N2	$\overline{Y}_f$	P3	N3	Y_f	Y_f
0	0	F1	Z	Z	F0	Z	F0	F1	Z	F1	1
0	1	F0	F1	Z	Z	F0	F0	F1	Z	F1	1
1	0	F0	F1	F1	Z	Z	F1	Z	F0	F0	0
1	1	F0	F1	Z	Z	F0	F0	F1	Z	F1	1

Table 9. Truth table for OR gate circuit with bridging fault between points 'A' and '1'

5.5 Short between A and Y

Another fault represented in the circuit is a short between points 'A' and 'Y'. This fault can also be considered in gate level with a connection between primary output 'Y' and primary input 'A' and can be explained by equation (EQ-5-6).

Y_f = A ∇ Y (EQ-5-6)

A	B	Y	Y_f
0	0	0	0
0	1	1	U
1	0	1	1
1	1	1	1

Table 10. Truth table for OR gate circuit with bridging fault between two points 'A' and 'Y'

The result for all input combinations is shown in Table 10.

5.6 Short between 1 and 2

Any bridging between source and drain in a transistor can be considered a stuck-at-on fault for the transistor. As illustrated in Table 7, the gate could be considered as stuck-at-0 for a P-type transistor. Figure 30 shows the pre-graph for a short between point '1' and '2'. The graph for the fault injected circuit above is shown in Figure 31.

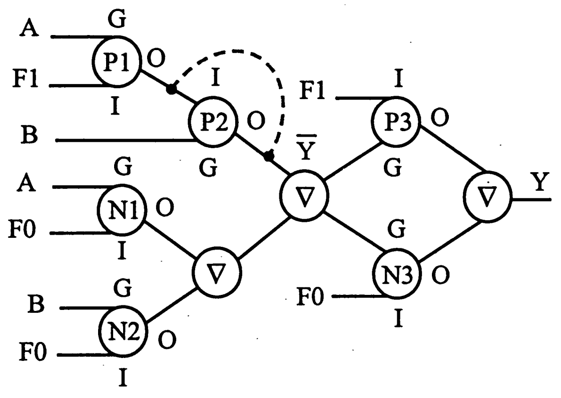

Figure 30. Pre-graph for OR circuit with bridging fault

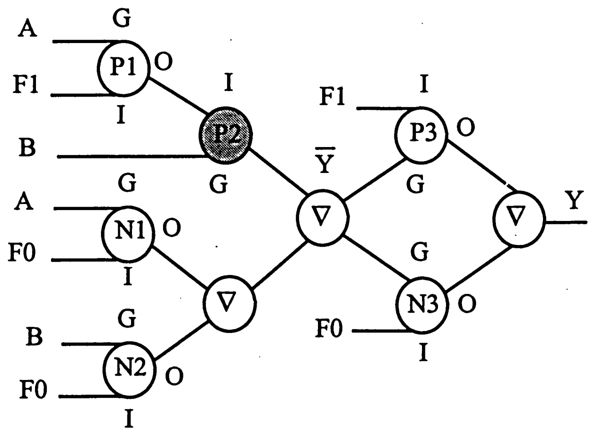

Figure 31. OR circuit graph with bridging fault

Equation (EQ-5-7) shows the bridging fault between points '1' and '2'.

$$\overline{Y}_f = \boldsymbol{P2}_f(\mathrm{B}, \mathrm{P1}(\mathrm{A}, \mathrm{F1})) \nabla [\mathrm{N1}(\mathrm{A},\mathrm{F0}) \nabla \mathrm{N2}(\mathrm{B}, \mathrm{F0})]$$

$$Y_f = \mathrm{P3}(\overline{Y}_f, \mathrm{F1}) \nabla \mathrm{N3}(\overline{Y}_f, \mathrm{F1}) \qquad \text{(EQ-5-7)}$$

All primary and intermediate logical values for the equation (Eq-5-7) with bridging fault between two points '1' and '2' are shown in Table 11.

A	B	P1	$P2_f$	N1	N2	$\overline{Y}_f$	P3	N3	Y_f	Y_f
0	0	F1	F1	Z	Z	F1	Z	F0	F0	0
1	0	Z	Z	F0	Z	F0	F1	Z	F1	1

Table 11. Truth table for OR gate with bridging fault between points '1' and '2'

CHAPTER 6
MULTIPLE FAULT SIMULATION

The simultaneous injection of two faults into a CMOS circuit is represented in Figure 32. Transistor P3 is stuck-at-on and a bridging fault is present between points ‘A’ and ‘1’.

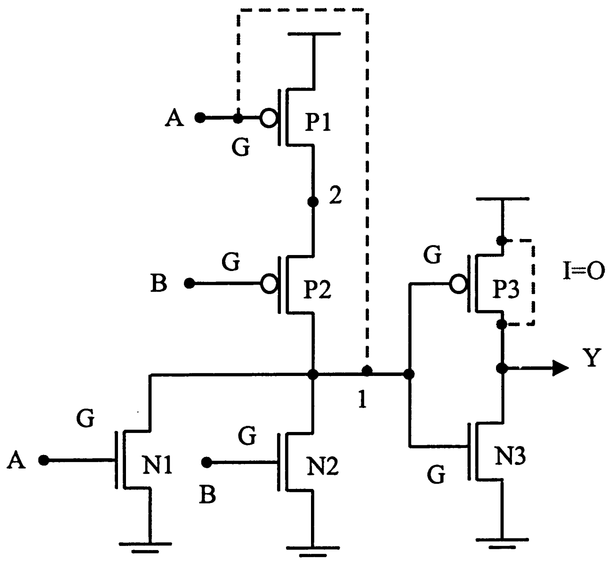

Figure 32. CMOS circuit with stuck-at-on for P3 and bridging between points 'A' and '1'

The pre-graph for the circuit (Figure 32) with two injected faults is displayed in Figure 33 and its graph is shown in Figure 34

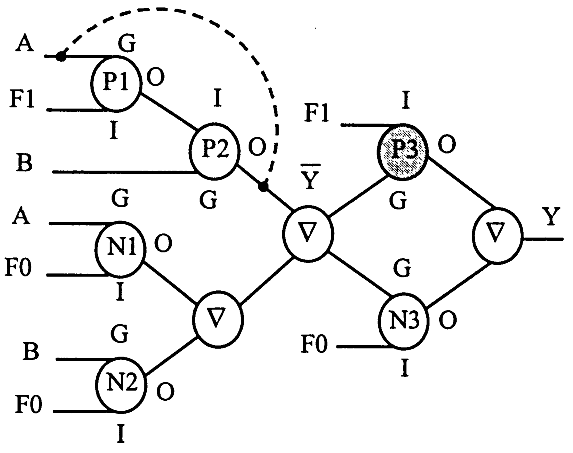

Figure 33. Circuit pre-graph with stuck-at-on for P3 and bridging between points 'A' and '1'

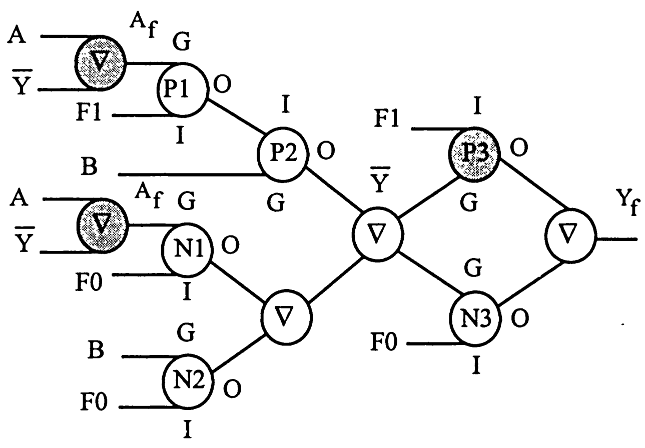

Figure 34. Circuit graph with stuck-at-on for 'P3' and bridging between points 'A' and '1'

Equation (EQ-7-1) is the fault free equation for the same circuit.

$$\overline{Y} = P2\ (B, P1(A, F1))\ \nabla\ [N1(A, F0)\ \nabla\ N2(B, F0)]$$

$$Y = P3\ (\overline{Y}, F1)\ \nabla\ N3\ (\overline{Y}, F0) \qquad \text{(EQ-7-1)}$$

By replacing $A_f = A\ \nabla\ \overline{Y}$ and $P3_f = F1$ in equation (EQ-7-1) we derive the results in equation (EQ-7-2).

$$\overline{Y}_f = P2\ (B, P1\ (A_f, F1))\ \nabla\ [N1\ (A_f, F0)\ \nabla\ N2\ (B, F0)]$$

$$Y_f = P3_f\ \nabla\ N3\ (\overline{Y}_f, F0) \qquad \text{(EQ-7-2)}$$

Table 12 shows all the primary and intermediate logical values for the circuit in Figure 32 with multiple faults between two points 'A' and '1' and stuck-at-on for P3 transistor. The result of two different types of faults simultaneously injected into the circuit is shown in this table. The grey columns represent gate behaviour for the fault injected at switch level.

A	B	A_f	P1	P2	N1	N2	$\overline{Y}_f$	$P3_f$	N3	Y_f	Y_f
0	0	F1	Z	Z	F0	Z	F0	F1	Z	F1	1
0	1	F0	F1	Z	Z	F0	F0	F1	Z	F1	1
1	0	F0	F1	F1	Z	Z	F1	F1	F0	U	U
1	1	F0	F1	Z	Z	F0	F0	F1	Z	F1	1

Table 12. Truth table for OR gate with multiple faults

CHAPTER 7
EXPERIMENTAL RESULTS

Experimental results were obtained in two phases. In first phase we checked the functionality of the algorithm with the Cadence Spectra simulator. The experiment was completed for all CMOS gates at transistor level. In order to explain the algorithm, just a few circuits were mentioned in this paper. In the second phase, Multiple Fault Simulation (MFS) software was developed and run on several ISCAS85 and ISCAS89 benchmark circuits.

7.1 Cadence Spectra results for a CMOS circuit

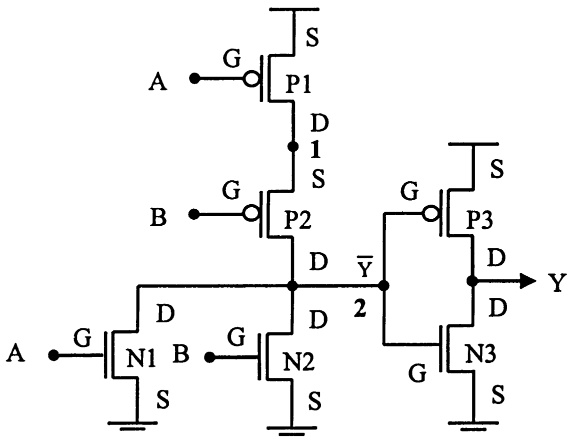

Figure 35. CMOS OR gate circuit

The experimental results from the Cadence Spectra for a CMOS circuit (Figure 35) with several faults are shown in Figure 36. These results are identical to those attained from the algorithm. As shown in the timing diagram (Figure 36), input signals 'A' and 'B', fault free output, several short faults and finally multiple fault outputs for circuit are discussed.

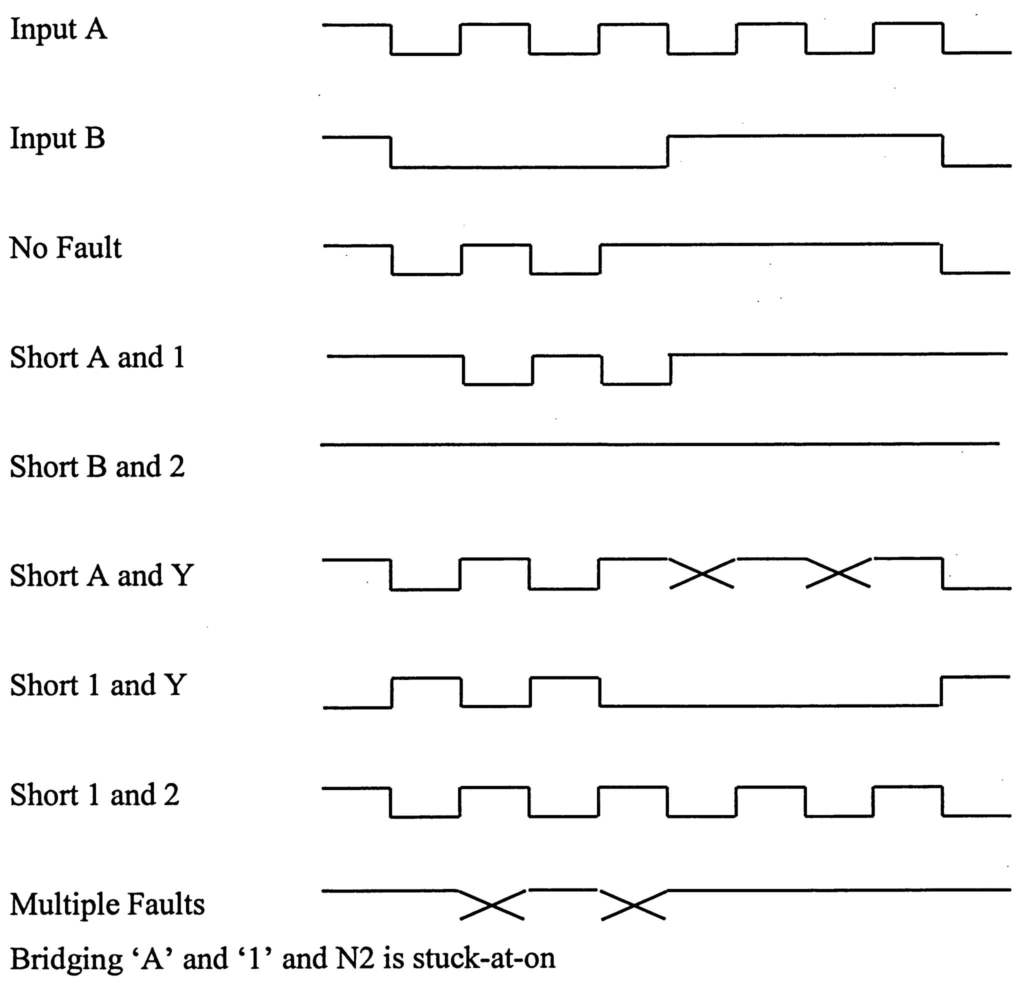

Figure 36. Fault free and faulty output timing diagram for circuit in Figure 35

In the timing diagram (Figure 36) with a short between points 'A' and '1' the OR gate behaves like NOR when B=0. In the next diagram with a short between 'B' and '2' the gate output or 'Y' is stuck-at-1. The next timing diagram represents a short between 'A' and 'Y'. This fault is modeled as bridging unknown. In the case of a short between '1' and 'Y' this defect changes the function of the OR to NOR. In the case of short between '1' and '2' the output signal is identical to the input signal 'A'. This short is modeled as 'B' stuck-at-0. Finally, the last timing diagram depicts two injected faults, a bridging between 'A' and '1', and at the same time, transistor 'P3' is stuck-at-on. This fault is also modeled as bridging unknown and stuck-at fault unknown.

The exact images from the Cadence Spectra simulator for a NOT gate circuit with a single bridging fault is shown in Figure 37.

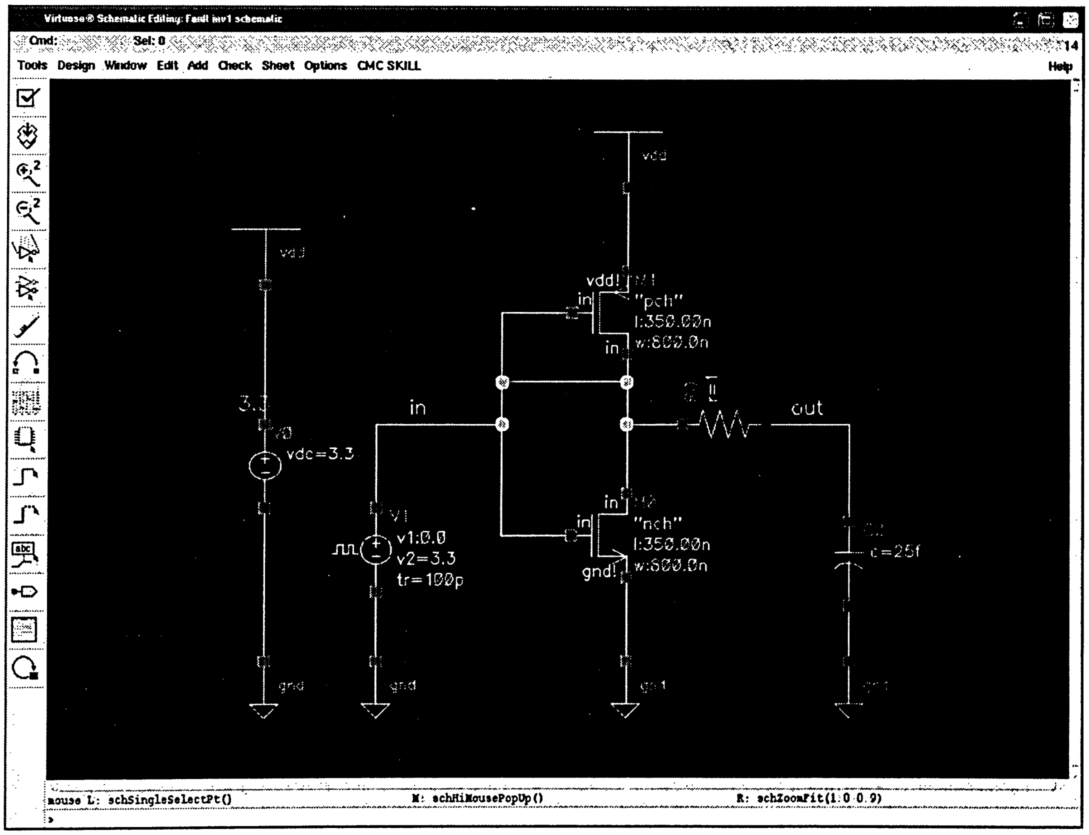

Figure 37. NOT gate schematic with bridging fault in Cadence Spectra

Input/output signals for circuit in Figure 37 are shown in Figure 38. It is evident that a NOT gate with a single bridging fault will behave like a buffer or a wire. It was proofed from the signals that the behaviour of actual circuit is exactly similar to algorithm behaviour.

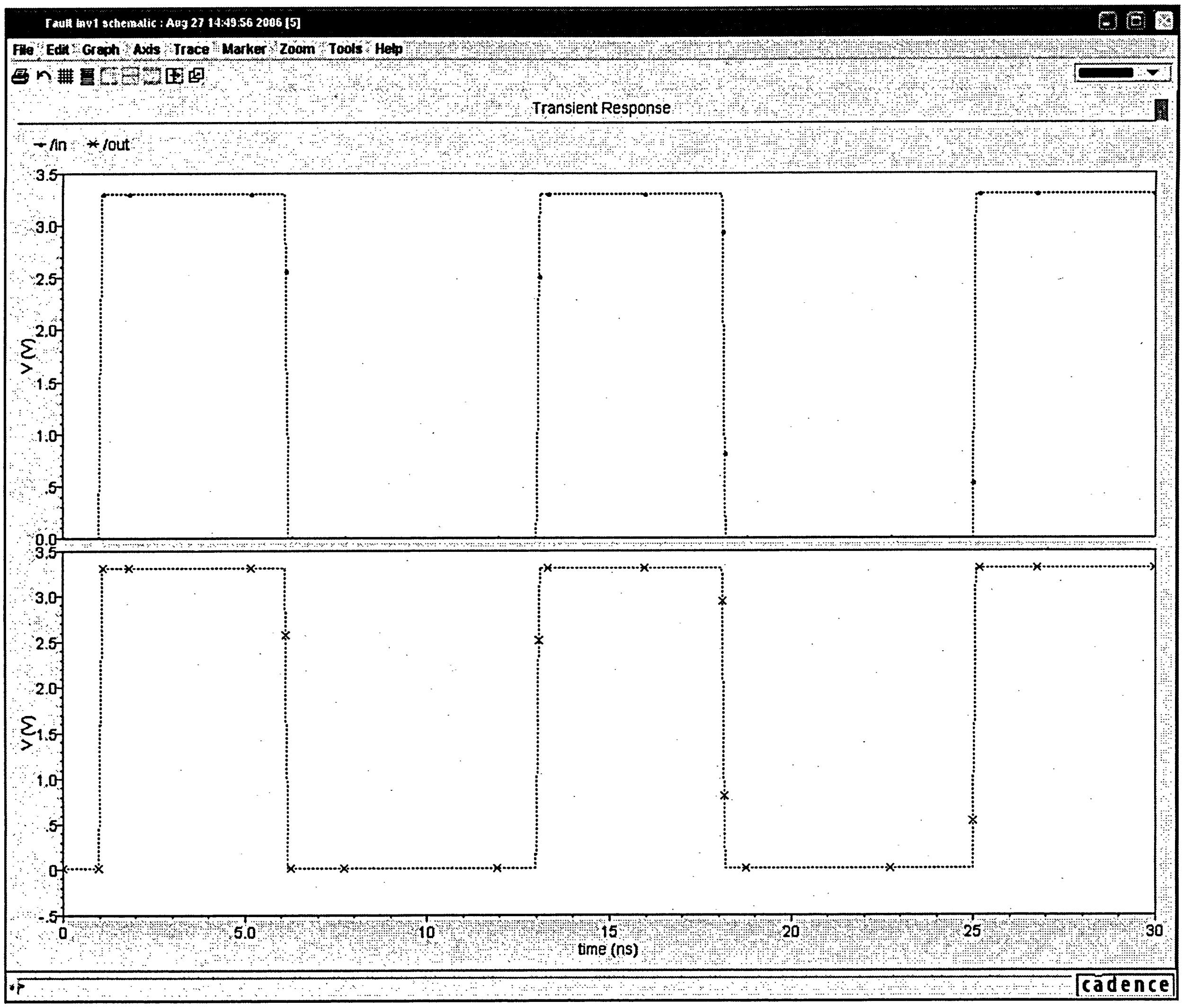

Figure 38. In/out timing diagram for circuit in Figure 37

Another experiment is shown in Figure 39 using an OR gate circuit. Multiple faults are injected into the circuit with the Cadence Spectra simulator and the input/output signals for all input combinations as shown. According to Figure 35 there is one bridging fault between points 'A' and '1' and a stuck-at-on fault for transistor 'N2'

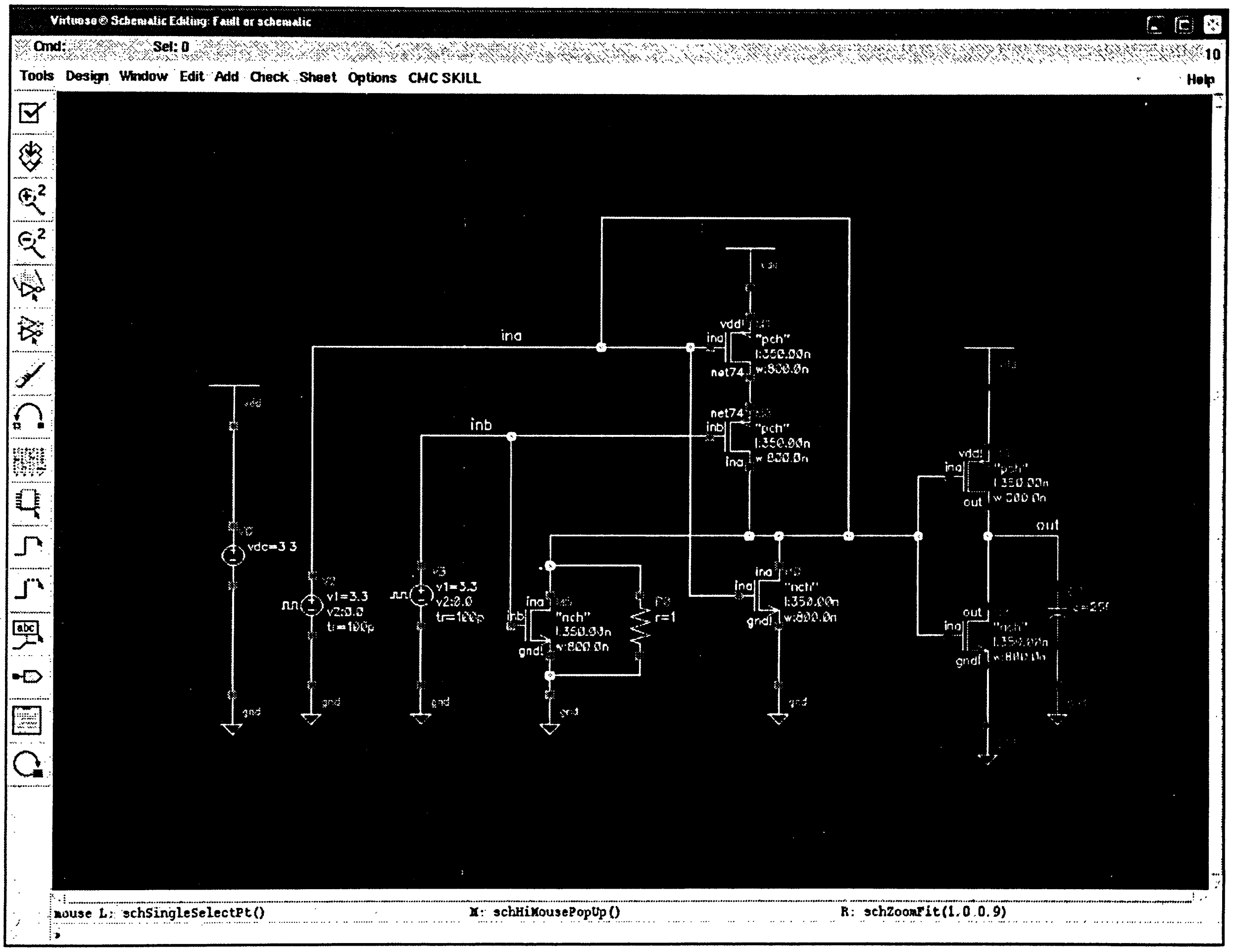

Figure 39. OR gate circuit with multiple fault injection in Cadence Spectra

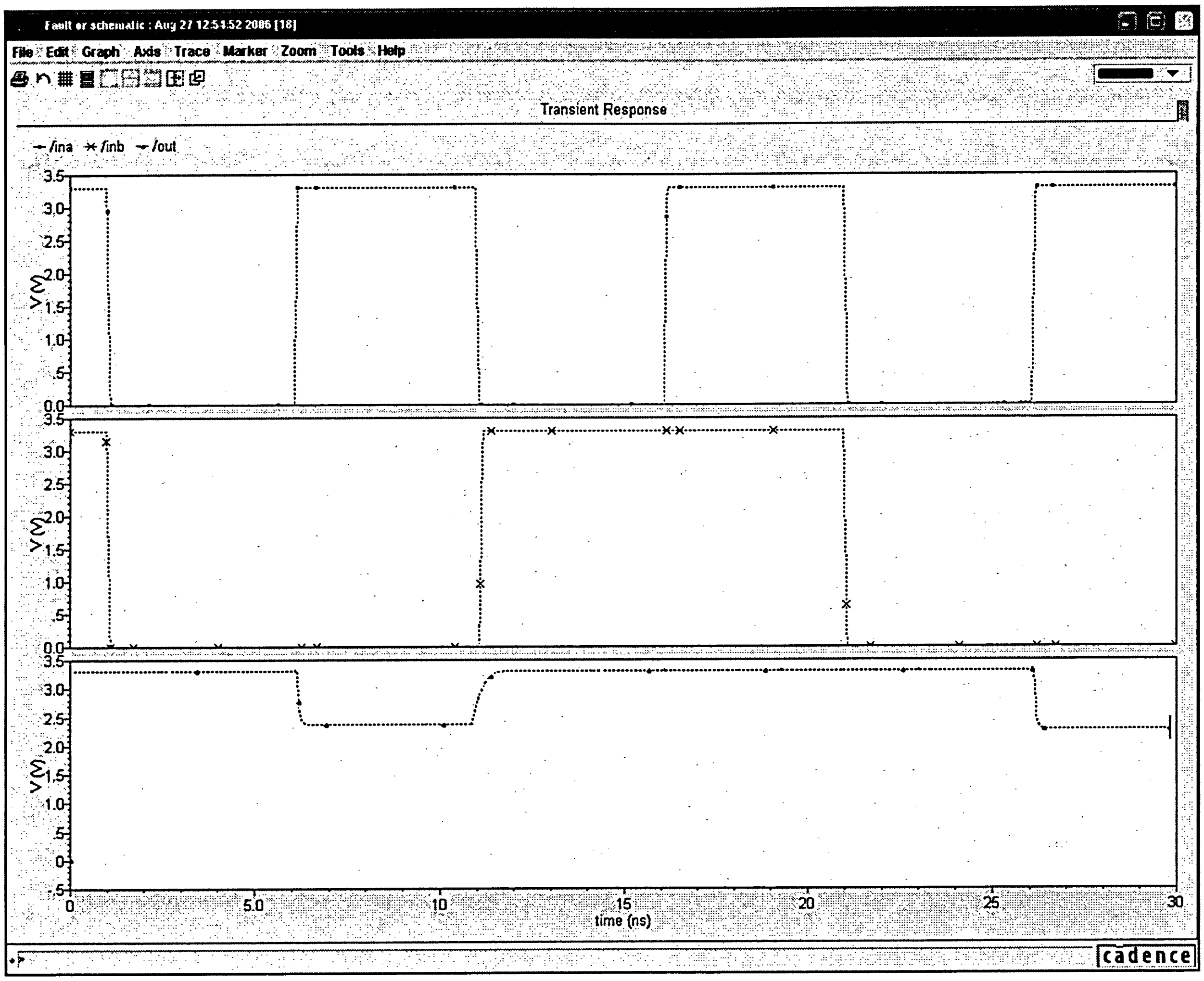

Figure 40. In/out timing diagram for circuit in Figure 39

Depicted in Figure 39 is a transistor model for OR gate with two different faults injected simultaneously into the circuit. Stuck-at-on for transistor 'N2' is modeled with a connection between source and drain. The result is show in Figure 40.

7.2 Multiple Fault Simulation (MFS) software

The pseudo-random input test sets are obtained from gate level test generation tools. Fault Coverage (FC) simulation results for single faults are shown in Table 13 and for multiple faults in Table 14.

As expected, results indicate that gate level test vector sets detect fewer switch level faults. In this case, switch level fault coverage was less than the gate level fault coverage. This result, while not a surprise, does confirms the fact that switch fault simulation can be a better design verification tool since a larger test set would be required to achieve switch level fault coverage similar to the gate level fault coverage. Any possible switch level faults that are undetected for the gate level test set could be detected by the larger switch level test set.

Circuit	# of Switches	# of Faults	Single FC %
C17	24	60	100
C2670	6212	1547	67.79
C7552	18802	4667	59.54
S27	66	130	99.72
S13207	30984	9705	57.67
S38417	85912	23816	52.88

Table 13. Percentage of Single Fault Coverage for Benchmark Circuits

In Table 14 the number of faults for each benchmark is multiplied by two indicating that two different faults were applied per iteration in the circuit. If the results of this table are compared with single fault coverage in the previous table it is clear that although the number of faults are doubled, the fault coverage is still near the previous result. This result for multiple fault simulation is obtained while the CPU time and the number of test sets are the same for the simulation.

Circuit	# of Switches	# of Faults	Multiple FC %
C17	24	60 x 2	91.94
C2670	6212	1547 x 2	56.54
C7552	18802	4667 x 2	55.81
S27	66	130 x 2	86.78
S13207	30984	9705 x 2	51.35
S38417	85912	23816 x 2	48.15

Table 14. Percentage of Multiple Fault Coverage for Benchmark Circuits

CPU time and memory requirements depend on both the number of nodes and the number and order of the Boolean variables in the circuit function. Therefore, the analysis of very large circuits requires embedding variable ordering and partitioning strategies into the analyzer. The simulation was run on a Pentium 4 system (3.0 GHz, RAM =1 GB, OS = Windows XP). Table 15 shows the CPU time based on this system. The unit of measurement is second.

Circuit	# Switches	CPU time(s)
C17	24	<1
C2670	6212	840
C7552	18802	1440
S27	66	<1
S13207	30984	3120
S38417	85912	6480

Table 15. Benchmarks Synthesis CPU Time

In comparison with related previous research [26][27][28] involving many different factors such as benchmark circuits and testing tools, the method presented here has better fault coverage and CPU time in terms of multiple fault injection, making it 2-3 times faster. In [26] which used a DP32 RISC processor for evaluation, the fault coverage varied between 51 to 56 percent. In other papers [27][28] using smaller benchmark circuits, this approach still results in better performance.

CHAPTER 8
CONCLUSION

Multiple fault injection and test synthesis for CMOS VLSI design has the highest priority in current technology. In general, fault injection at switch level is more accurate than at gate level. Many regions at gate level are not accessible and most of the physical defects occur inside the gates. In the approach presented here, we endeavoured to address all possible physical problems at switch level for CMOS technology-based circuits of any size. However, the focus was mainly on analyzing inside digital gates using simulation in addition to the algorithm introduced here capable of mapping faults from a transistor level model to a gate level model. This algorithm can also be used for circuit synthesis and verification. The equations of functions can analyze CMOS circuits whether or not faults are present in the circuit. Thus, if a fault is not injected into the circuit, the algorithm could be considered for circuit verification and synthesis.

Another advantage of this algorithm is its ability to analyze different types of multiple faults. A multiple fault refers to any combination of bridging, stuck-at-fault and open-wire in the circuit occurring simultaneously. Experimental results using Cadence tools and MFS software confirm that the algorithm is reliable for CMOS technology. This algorithm can reduce the time needed for fault simulation. Although switch level fault simulation is more time consuming than gate level, it is essential for advanced technology of the future.

CHAPTER 9
FUTURE WORK

Due to various deep submicron effects, a circuit may fail to operate at the desired clock frequency. Timing failure analysis is a procedure used to locate the source of timing failures. The resolution and the hit rate of the candidates, which are reported by the delay-fault diagnosis process, will determine the efficiency of timing failure analysis. The resolution is defined as the ratio of the number of real fault sites to the total number of the reported candidates. Unfortunately, the existing delay-fault diagnosis methodologies suffer from poor resolution or low scalability.

When a physical defect leads to excessive delays on signals instead of altering the logic function of the circuit, it is no longer a static defect. Such unknown and unpredictable defect behaviours make it very difficult to analyze a fault. The dynamic nature of such faults disturbs the timing of the logic propagation. In our approach, the timing failures caused by short and open resistive faults inside the gates, the level of the voltage presenting at the gate output will be affected. The relation between the logic propagation delay and its eventual effect on the circuit output voltage is determined by performing a switch-level analysis on CMOS primitive gates.

Consequently the defected voltage, which is carrying timing disturbance, should be propagated to primary output. As the delay size is relatively small, the voltage changes are not causing logical problem at output of the gate and tracing these kinds of signals to the output is very difficult. To resolve this problem we are going to using a nine-valued voltage model on top of a five-valued voltage model to propagate faulty signals. The main concept is that those faults that cause logical errors in the nine-valued voltage model are still delay faults in five-valued voltage model. Dynamic behaviours of resistive defects tend to delay the correct logic state propagation at the gate output. Generally, the delay is caused by certain fault locations with respect to the input vector the gate is subjected to, defect resistance of the fault and the technology variation. These faults in the five-valued voltage model only disturb the logic propagation time without adversely changing the functional output in most cases. As a result the output voltage fluctuates between ranges of intermediate voltage value. However the disturbance of the propagation time can change the functional output in nine-valued voltage model. By reducing feature sizes, resistive fault occurrences are expected to be on a rise. Hence, their effects on the logic voltage-levels in static CMOS primitive gates subject to technologies like 65nm, 45nm and 32 nm are determined.

PUBLICATIONS

1. M. Reza Javaheri, R. Sedaghat, Leo Kant, Jason Zalev, *"Verification and Fault Synthesis Algorithm at Switch-Level"*, Journal of Microprocessors and Microsystems, Science Direct, Elsevier, Volume 30, Issue 4 , 6 June 2006, Pages 199-208

2. R. Sedaghat, R. Javaheri, *"Bi-directional Switch-Level Verification and Multiple Fault Synthesis Algorithm"*, WSEAS TRANSACTIONS on CIRCUITS AND SYSTEMS, Accepted, 2006

3. R. Sedaghat, M. Kunchwar, R. Abedi, R. Javaheri, *"Transistor-level to Gate-level Comprehensive Fault Synthesis for n Input Primitive Gates"* International Journal of Microelectronics Reliability, Science Direct, Elsevier, Accepted, 2006

4. M. Reza Javaheri, Reza Sedaghat, Mayuri Kunchwar, *"Delay Propagation for Resistive Faults in Deep Sub-Micron Technologies"* International Journal of Microelectronics Reliability, Science Direct, Elsevier, submitted, 2006

REFERENCES

[1] Al-Khalili, D.; N-Rozon, C.; B. Show, D.; Fault security analysis of CMOS VLSI circuits using defect-injectable VHDL models ElSEVIER, INTEGRATION, the VLSI journal 32 (2002), pp. 77–97.

[2] Jolly, S.; Parashkevov, A.; and McDougall, T.; Automated equivalence checking of switch level circuits, in proceedings of the IEEE/ACM International conference on DAC, (2002), pp. 299-304.

[3] McDonald, C.B.; and Bryant, R.E.; CMOS Circuit verification with symbolic switch level timing simulation, IEEE Transaction on Computer-Aided Design of integrated Circuits and Systems, vol. 20 Issue: 3 (March 2001), pp. 458-474.

[4] Leveugle, R.; A low-cost hardware approach to dependability validation of IPs. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, (2001) pp. 242–249.

[5] Antoni, L.; Leveugle, R.; Feher, B.; Using run-time reconfiguration for fault injection in hardware prototypes. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, (2000) pp. 405–413.

[6] Leveugle, R.; Fault Injection in VHDL descriptions and emulation. IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (2000), pp. 414–419.

[7] Krishnaswamy, V.; Casas, J.; Tetzlaff, T.; A Switch Level Fault Simulation Environment: Design Automation Conference, 2000. Proceedings 2000. 37th (2000), pp. 780-785.

[8] Cheng, K.T.; Huang S.Y.; and Dai, W.J.; Fault emulation: a new methodology for fault grading. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 18 - 10 (1999), pp. 1487–1495.

[9] Casas, J.; Yang, H.; Khaira, M.; Joshi, M.; Tetzlaff, T.; Otto, S.; and Seligman, E.; Logic verification of very large circuits using Sharkin: Proceedings of the Twelfth International Conference on VLSI Design (1999) pp. 310–317.

[10] Benso, A.; Rebaudengo, M.; Impagliazzo, L.; Marmo, P.; Fault-list collapsing for fault injection experiments, RAMS'98. Annual Reliability and Maintainability Symposium (1998), pp. 383–388.

[11] Hungse Cha; Rudnick, E.M.; Patel, J.H.; Iyer, R.K.; Choi, G.S.; A gate-level simulation environment for alpha-particle-induced transient faults Computers, IEEE Transactions on , Volume: 45 , Issue: 11 (1996), pp. 1248–1256.

[12] Abramovici M.; Breuer, M.A.; and Friedman, A.D.; Digital System Testing and Testable Design, Revised edition IEEE Press, (1995).

[13] Verilog Hardware Descriptor Language Reference Manual (LRM) DRAFT, IEEE 1364, (April 1995).

[14] Kuehlmann, A.; Cheng, D.I.; Srinivasan A.; and Lapotin, D.P.; Error diagnosis for transistor-level verification: Proceedings of the 31th IEEE/ACM Design Automation Conference (1994) pp. 218–224.

[15] Alt, J.; Mahlstedt, U.; Simulation of non-classical faults on the gate level – fault modeling – Institute fur Theoretische Elektrotechnik Universitat Hannover, Germany, 11th VLSI Test Symposium, (April 1993), pp. 351 – 354.

[16] Dahlgren, P.; Liden, P.; Efficient modeling of switch-level networks containing undetermined logic node states. Proceedings IEEE/ACM International Conference on CAD (1993) pp. 746–752.

[17] N.H.E. Weste and K. Eshraghian, Principles of CMOS VLSI design. Addison-Wesley (1993) Chapter 7.

[18] Blaauw, D.T.; Saab, D.G.; Banerjee, S.P.; Abraham, J.A.; Functional abstraction of logic gates for switch-level simulation: Proceedings of the European Conference on Design Automation (1991) pp. 329–333.

[19] Johnson, B.W.; Design and analysis of fault-tolerant digital systems. Addison-Wesley (1989) Chapter 2.

[20] Wadsack, R.L., "Fault Modeling and Logic Simulation of CMOS and MOS Integrated Circuits" BELL System Technical Journal, vol. 57, no.5, (May-June 1987), pp. 1449-1474.

[21] Bryant, R.E.; A switch-level model and simulator for MOS digital systems. IEEE Transactions Computers C-33 2 (1984), pp. 160–177.

[22] Chandramaouli, R.; "On testing stuck-open faults," FTCS Fault Tolerant Computing, (1983), pp. 258-265.

[23] Ditlow, G.; Donath, W.; and Ruehli, A.; Logic equations for MOSFET circuits: Proceedings of the IEEE International Symposium on Circuits and Systems (1983) pp. 752–755.

[24] Kohyama, S.; and Sate, T.; "CMOS technologies for VLSI circuits," IEEE Conf. on VLSI, (1981), pp. 24-25.

[25] Mei, K.C.Y., "Bridging and Stuck-At-Fault", IEEE Trans. Computers, vol. C-23, no.7, (July 1974), pp. 720-727.

[26] Zarandi H.R.; Miremadi, S.G.; Ejlali, A.; Dependability analysis using a fault injection tool based on synthesizability of HDL models Defect and Fault Tolerance in VLSI Systems, 2003.

[27] Dahlgren P.; Switch-level bridging fault simulation in the presence of feedbacks Test Conference, 1998. Proceedings International 18-23 Oct. 1998 Page(s):363 – 371

[28] Vandris, F.; Sobelman, G.; Algorithms for fast, memory efficient switch-level fault simulation Design Automation Conference, 1991. 28th ACM/IEEE June 17-21, 1991 Page(s):138 – 143

[29] Mahlstedt. U., Heinik, M., Alt, J., "Test Generation for I, Testing and Leakage Fault Detection in CMOS Circuits", EURO-DAC 92, pp.486-491

[30] Eldred, R.D., "Test Routines Based on Symbolic Logical Statements", Joumal ACM, vo1.6, no. 1, 1959

[illegible] 10-5-68