FAULT TOLERANCE IN REVERSIBLE LOGIC

SHAMRIA SABATINA LATIF
Bachelor of Science
Ahsanullah University of Science & Technology, Bangladesh, 2015

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

FAULT TOLERANCE IN REVERSIBLE LOGIC

SHAMRIA SABATINA LATIF

Date of Defence: May 23, 2018

Dr. J. E. Rice Supervisor	Professor	Ph.D.
Dr. W. Osborn Committee Member	Associate Professor	Ph.D.
Dr. J. Morris Committee Member	Professor	Ph.D.
Dr. H. Cheng Chair, Thesis Examination Committee	Associate Professor	Ph.D.

Dedication

I dedicate this thesis to the researchers in reversible computing.

Abstract

In recent years reversible logic has offered a promising alternative to traditional logic circuits. Reversible logic introduces a mechanism which allows theoretically zero energy dissipation by eliminating the possibility of information loss. However, it is also desirable that all computation should ideally be done in a fault tolerant manner. To address this we propose techniques to achieve fault tolerance in reversible logic based on a passive hardware redundancy technique. We propose two new designs for a reversible majority voter circuit that can be used to implement fault masking. Comparisons to existing designs are presented in terms of cost metrics such as gate count, garbage outputs, constant inputs, and quantum cost. Comparative failure probability analysis of the proposed voter circuits is also provided. Simulation results of the voter circuit failure probabilities over different numbers of trials are also presented. Our approach can be used to determine the circuit failure probability by using the gate failure probabilities. The proposed methodology can provide useful information for future reversible gate fabrication and designing future fault tolerant reversible circuits.

Acknowledgments

Thanks to the Almighty for giving me the opportunity, strength, and ability to learn and complete this research. It has been a period of intense learning both at the academic and professional level. I would like to reflect my gratitude on the people who have helped and supported me so much throughout my graduate studies.

I would like to express my sincere gratitude to my thesis Supervisor, Dr. Jacquline E. Rice for her endless support, encouragement, and guidance throughout my graduate studies. I also admire her for standing beside me during my academic, professional and personal developments and odds. My supervisor deserves more than thanks for her role in my life.

I would also like to thank my committee members, Dr. Wendy Osborn and Dr. Joy Morris, for their constructive feedback and suggestions for improving my research work. My sincere thanks go to Dr. Howard Cheng and Dr. Amir Akbary for their academic and administrative help. I thank the Administrative Support, Ms. Barb Hodgson, for her cordial assistance. I also thank the departmental graduate students especially Asif, Marzia and Nurgul for the academic discussions and valuable suggestions. I am grateful to Dr. Adriana Predoi-Cross for all of her kind supports.

My sincere gratitude and special thanks go to my parents, who have been with me starting from the beginning and provided their unconditional help and support.

Lastly, I am thankful to the School of Graduate Studies (SGS) of the University of Lethbridge for their financial help. Also, a heartfelt thanks to Canada for giving me the opportunity to learn and grow.

Contents

Contents vi

List of Tables viii

List of Figures ix

1 Introduction 1
- 1.1 Objectives . . . 2
- 1.2 Contributions . . . 3
- 1.3 Thesis Organization . . . 4

2 Background 6
- 2.1 Logic Computation . . . 6
  - 2.1.1 Traditional Logic Computation . . . 6
  - 2.1.2 Reversible Logic Computation . . . 8
- 2.2 Logic Gates . . . 9
  - 2.2.1 Reversible Gates . . . 9
  - 2.2.2 NOT Gate . . . 10
  - 2.2.3 CNOT Gate . . . 10
  - 2.2.4 Toffoli Gate . . . 11
  - 2.2.5 Fredkin Gate . . . 12
- 2.3 Cost Metrics in Reversible Logic . . . 14
  - 2.3.1 Gate Count (GC) . . . 14
  - 2.3.2 Garbage Outputs (GO) . . . 14
  - 2.3.3 Constant Inputs (CI) . . . 15
  - 2.3.4 Quantum Cost (QC) . . . 15
- 2.4 Fault, Failures and Errors . . . 17
- 2.5 Fault Tolerance in Reversible Logic . . . 17
  - 2.5.1 Fault Detection . . . 18
  - 2.5.2 Fault Location . . . 18
  - 2.5.3 Fault Recovery . . . 18
  - 2.5.4 Fault Masking . . . 18
- 2.6 Redundancy to Achieve Fault Tolerance . . . 19
- 2.7 Hardware Redundancy . . . 20
  - 2.7.1 Triple Modular Redundancy . . . 21

3 Reversible Majority Voter Circuits **23**
3.1 Fault Tolerant Reversible Circuits . . . 23
3.1.1 Reversible Majority Voter Circuits in the Literature . . . 24
3.2 Proposed Reversible Majority Voter Circuits . . . 27
3.2.1 Fault Tolerant Reversible Circuit Design . . . 29
3.3 Voter Circuit Comparisons . . . 31
3.4 Voter Circuit Application in Reversible Fault Models . . . 32
3.4.1 Fault Masking of a Single Bit Fault . . . 33
3.4.2 Fault Masking of a Single Gate Fault . . . 33
3.4.3 Fault Masking of Crosspoint Faults . . . 34
3.4.4 Fault Masking of the Family of Missing Gate Faults . . . 36
3.5 Fault Masking within the Voter Circuits . . . 37

4 Fault Testing **41**
4.1 Approaches to the Reliability and Failure Probability Analysis . . . 41
4.1.1 Reliability Analysis in Traditional Logic . . . 42
4.1.2 Failure Probability Analysis for Traditional Logic Circuits . . . 43
4.2 Background and Definitions . . . 44
4.2.1 Set Theory . . . 45
4.2.2 Mutually Exclusive and Collectively Exhaustive Events . . . 45
4.2.3 Probabilities of Event Occurrences . . . 46
4.2.4 Probability of Occurrence of At Least One Event . . . 46
4.2.5 Binomial Distribution . . . 47
4.3 Proposed MVC Failure Probability Analysis Techniques . . . 48
4.4 MVC Failure Probability for Gate Faults . . . 53
4.5 Implications of the Analysis . . . 56

5 Conclusions **62**
5.1 Contributions . . . 62
5.2 Future Works . . . 63

Bibliography **66**

A Voter Circuit Behavior **71**

List of Tables

2.1 Truth tables of traditional logic functions. 7
2.2 Truth tables of reversible logic functions. 8
2.3 Truth table of a 3×3 positive-controlled Toffoli gate. 12
2.4 Behavior of a 3×3 positive-controlled Fredkin gate. 13
2.5 Quantum cost of some reversible gates. 16

3.1 Truth table of the majority voter with two reversible gates. 28
3.2 Truth table of the majority voter with three reversible gates. 29
3.3 Comparative results of different majority voter circuits. 31

4.1 Failure probabilities of the proposed MVCs for an input of 100. 54

List of Figures

2.1 Reversible logic gates. 10
2.2 Toffoli gates. 11
2.3 Reversible Fredkin gates. 13
2.4 Different approaches to fault tolerance and how they are categorized [25]. . 19
2.5 Illustration of TMR 21

3.1 Reversible majority voter circuit [7]. 24
3.2 Reversible fault tolerant multiplexing scheme [7]. 25
3.3 Reversible majority voter circuit [43]. 26
3.4 Majority voter with triplicated voter outputs proposed in [61]. 26
3.5 Example of input assignments and subsequent fault masking for the circuit shown previously in Figure 3.4. 27
3.6 MVC with two reversible gates. 27
3.7 MVC with three reversible gates 28
3.8 Fault tolerant reversible circuit design 30
3.9 Reversible fault tolerant full adder circuit design. 31
3.10 Fault masking of a SBF. 34
3.11 Fault masking of a SGF. 35
3.12 Crosspoint faults in a reversible circuit. 35
3.13 Fault masking of a disappearance fault. 36
3.14 Voter circuit operations in the presence of a single fault on line b. 38
3.15 Voter circuit operations in the presence of a single fault on line c. 38
3.16 Voter circuit operations in the presence of a single fault on line b. 39
3.17 Voter circuit operations in the presence of a single fault on line c. 39
3.18 Voter circuit operations in the presence of a single fault on line b. 40
3.19 Voter circuit operations in the presence of a single fault on line c. 40

4.1 Probability of occurrence of at least one event. 47
4.2 Design comparison of the proposed MVCs. 49
4.3 Probability of the occurrence where at least one gate is missing or inactive. 52
4.4 Pseudo code for MVC failure analysis for an input of 100. 55
4.5 MVC failure probabilities for gate fault probabilities, $x = 0.001\%$. 57
4.6 MVC failure probabilities for gate fault probabilities, $x = 0.002\%$. 58
4.7 MVC failure probabilities for gate fault probabilities, $x = 0.003\%$. 59
4.8 MVC failure probabilities for gate fault probabilities, $x = 0.004\%$. 60
4.9 MVC failure probabilities for gate fault probabilities, $x = 0.005\%$. 61

A.1 Circuit outputs for missing or inactive negative-controlled CNOT gate. . . . 71

A.2 Circuit outputs for missing or inactive positive-controlled Fredkin gate. . . 72
A.3 Circuit outputs for missing or inactive gates. 73

Chapter 1

Introduction

Power dissipation and the overheating of traditional circuits are ongoing concerns in digital logic design. Digital electronic devices are now faster and more complex than their predecessors. Such increases in speed and complexity often lead to heat dissipation during logic computation [17]. The ever growing need for faster and more power efficient devices requires a technology that can reduce power consumption during computation. Researchers suspect that the current technology has reached the limit of transistor density [2] and are willing to explore possibilities in reversible logic. In 1961, R. Landauer showed that for every irreversible bit operation the amount of energy loss is $kT\ln 2$ (where k is Boltzmann's constant, $1.38 \times 10^{-23} JK^{-1}$ and T is the operating temperature in Kelvin) [30]. Although this appears to be a small amount of energy, if we think about an entire processor where millions of transistors are operating then the dissipated energy can be significant. According to Moore's observation, in a densely integrated circuit, the number of transistors will double approximately every two years. This will increase the heat dissipation exponentially with time [51]. However, if a system is able to return to its initial state from its final state, no energy would be dissipated and theoretically the system would act as a lossless system [6]. In 1973, C. H. Bennett proved that for a circuit to not dissipate energy the circuit must be logically reversible. Reversible logic computation can decrease or even eliminate the energy dissipation specified by R. Landauer in [30]. Currently, reversible logic has applications in quantum computing [22], adiabatic CMOS fabrication [5], optical information processing [8], DNA computing [50], cryptography [56] and nanotechnology [20].

A fault is a physical or functional imperfection that may occur within the hardware or software modules. Faults may result in incorrect functioning of circuits and introduce errors to the system. Fault tolerance strategies can help to avoid or eliminate errors at the output of a system. Also, fault tolerance features can help to offer more reliable performance and mitigate the risk of system failure. Unfortunately, there are few studies that focus on designing fault tolerant reversible circuits. Therefore, exploring techniques for designing fault tolerant reversible circuits has become a topic of interest to reversible logic researchers [25].

1.1 Objectives

- This thesis explores the existing work on designing fault tolerant reversible circuits and presents a passive hardware redundancy technique for achieving fault tolerance in reversible logic. Previous studies in [16, 18, 36, 57] have introduced several reversible logic gates and proposed synthesis methods incorporating these gates. Fault detection within reversible circuits is presented in [45, 55, 26]. The use of the parity preserving property is a commonly used fault detection technique and is still an active area of research [45, 55, 26]. However, the parity preserving property in reversible circuits can only detect the presence of a fault. Most of the literature on fault tolerance e.g. [19, 23, 39, 45] offers fault detection features which do not correct or mask the faulty output. To be labeled as fault tolerant, a circuit should either correct or mask the fault at the output.

- Our proposed hardware fault tolerance technique uses the concept of traditional triple modular redundancy (TMR). The TMR technique requires a majority voter circuit (MVC) for fault masking purposes. The basic function of a MVC is to mask the faults and provide corrected output. Designing reversible fault tolerant circuits using the concept of TMR is one of the main objectives of this thesis. We propose two new designs for a reversible majority voter circuit for fault masking in reversible logic.

- In reversible logic, a one-to-one mapping between the inputs and outputs must be maintained. This requirement often introduces design complexity to a reversible circuit. Design complexity of a reversible circuit is measured by commonly used cost metrics such as gate count, quantum cost, number of constant inputs and garbage outputs. We analyze the design complexity of our proposed voter circuits and present the macro gate level circuit implementation. We also present a comparative analysis of our proposed designs and the existing works in the literature [7, 43, 61].

- One of the objectives of this thesis is to evaluate the performance of our designed majority voter circuits (MVCs). We demonstrate that the proposed MVCs can correct a single bit fault, a single gate fault, a crosspoint fault and the family of missing gate faults.

- Fault tolerance improves reliability by keeping the circuit operational in the event of a failure. Therefore, it is important to assess the reliability of a design and study a circuit's failure probabilities. In a voting based fault masking technique, achieving fault tolerance largely depends on the voter circuit operation. One of the objectives of this thesis is to determine the voter circuit's failure probabilities based on its components (i.e. the comprising gate failure probabilities). The voter circuit failure probabilities are then expanded over different numbers of trials using the binomial distribution. Our proposed probability analysis and simulations can help to study the robustness of future fault tolerant reversible logic designs.

1.2 Contributions

The contributions of this thesis are listed below:

- In this thesis, we propose an approach to achieve fault tolerance in reversible logic circuits. We offer a majority voting based TMR approach for masking faults that occur in the original circuit.

- TMR in reversible circuits requires the design of a reversible majority voter circuit. In our work, we propose two new designs for a reversible majority voter circuit.
- We present a fault tolerant design for a reversible full adder circuit and demonstrate how our proposed voter circuits can be used to design any fault tolerant reversible circuits.
- We demonstrate the proposed reversible majority voter circuit application for masking a single fault occurring in any of the voter circuit's inputs, i.e. before the voter circuit. A single fault assumption considers a situation where at most one fault is present in a circuit. We present the fault masking capability of our proposed voter circuits in the case of a single bit fault (SBF), a single gate fault (SGF), a crosspoint fault and the family of missing gate faults.
- We also demonstrate that under specific conditions the second of our proposed voter circuits offers better fault masking capability than existing designs in the case of a single fault occurring inside the voter circuit.
- We propose a failure probability analysis of the proposed majority voter circuits. We demonstrate that the methodology proposed in this thesis can be used to determine a voter circuit's failure probabilities based on the gate failure probabilities.

1.3 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 provides a brief introduction to reversible logic and related background. The basic concepts of fault, failures and errors are discussed. Cost metrics in reversible logic are defined. An overview of fault tolerance techniques as well as the concept of hardware redundancy is provided.

Chapter 3 introduces a fault masking technique to achieve hardware fault tolerance in reversible logic. We propose two new designs of a reversible majority voter circuit (MVC).

A comparison of our proposed designs and the existing approaches [7, 43, 61] is presented.

Chapter 4 provides mathematical analysis and simulations to estimate the failure probability of the proposed MVCs. Using the concepts of set theory and binomial distribution we propose failure probability analysis of the proposed voters.

Chapter 5 concludes the thesis by summarizing the contributions of this research and discussing possible future work.

Chapter 2

Background

This chapter introduces the background of reversible logic computation as well as the fundamental concepts of fault, failure and errors. Fault tolerance techniques are discussed as background for our proposed method of desigining fault tolerant reversible circuits.

2.1 Logic Computation

In computing, logic operations are used to model the information flow through digital circuits. A logic operation connects and verifies two or more units of information. Logic that implements a Boolean function is also known as digital or traditional logic. Modern digital devices follow the principle of Boolean logic and define a problem in terms of Boolean functions. A circuit element that performs a logic operation is called a logic gate. Due to the limitations of existing physical components in traditional logic (e.g. heat dissipation) researchers are showing more interest in reversible logic computation [5]. The following subsections explain the concepts of traditional and reversible logic computation.

2.1.1 Traditional Logic Computation

In traditional logic, a function maps one or more inputs to one or more outputs in a Boolean domain B, where $B = \{0, 1\}$. The following mathematical expression represents a traditional logic function with k inputs and n outputs in the Boolean domain:

$$f : B^k \longrightarrow B^n \tag{2.1}$$

where k and n are positive integers.

In most cases, the number of outputs is less than the number of inputs (i.e. $n < k$). The input-output relation of a traditional logic gate can be described by the associated truth table. A truth table shows the mapping from the k input columns to the n output columns over 2^k rows of any Boolean logic function.

Table 2.1: Truth tables of traditional logic functions.

(a) NOT Operation

A	NOT (A)
0	1
1	0

(b) OR Operation

P	Q	$P+Q$
0	0	0
0	1	1
1	0	1
1	1	1

(c) Full Adder Operation

$Carry_{in}$	A	B	$Carry_{out}$	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The truth table describes all possible logic values of the inputs along with the resulting outputs. Table 2.1a shows a NOT gate operation in traditional logic. The input-output relationship of a traditional NOT gate operation is one-to-one and this is a reversible operation. A reversible operation is invertible and the input-output relationship of a reversible operation is one-to-one and onto. However, Table 2.1b and Table 2.1c show functions for which the relationship between inputs and outputs is not one to one. For example, a conventional OR gate has two inputs and yields a single output. Thus, the OR gate is logically irreversible, meaning it is not possible to determine unique inputs for all the outputs. For example in Table 2.1b, the output value is 1 for inputs $PQ = 01$, 10, and 11. Thus the operation is not reversible. A similar irreversible operation is observed for the truth table shown in Table 2.1c and most other traditional logic functions.

Table 2.2: Truth tables of reversible logic functions.

(a) NOT Operation.

A	NOT (A)
0	1
1	0

(b) Controlled NOT Operation.

P	Q	X	Y
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

(c) Full Adder Operation.

$Constant_{in1}$	$Carry_{in1}$	A	B	$Carry_{out}$	Sum	$Garbage_{out1}$	$Garbage_{out2}$
0	0	0	0	0	0	0	0
0	0	0	1	0	1	1	1
0	0	1	0	0	1	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	1	0	1
1	0	0	0	1	0	0	0
1	0	0	1	1	1	1	1
1	0	1	0	1	1	1	0
1	0	1	1	0	0	0	1
1	1	0	0	1	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	1	0	1

2.1.2 Reversible Logic Computation

In reversible logic a one to one mapping exists between the input and output assignment. Therefore, reversible logic gates are bijective requiring an equal number of inputs and outputs. Thus, reversible gates do not erase any information during computation and the computation can be undone to recover the input information [44]. In the scope of this thesis, our research addresses only Boolean reversible logic functions. The truth table for a reversible function of k variables requires 2^k rows and $2k$ columns. The bijective relationship between the inputs and outputs of a NOT, Controlled NOT, and full adder logic functions are shown in Table 2.2.

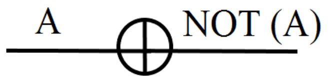
An irreversible logic circuit can be transformed into a reversible circuit by adding inputs and/or outputs. These additional inputs are called constant inputs and the unused outputs are called garbage outputs. For example, the truth table of a reversible full adder presented in Table 2.2c shows additional constant input and garbage outputs as compared to the truth table of the irreversible full adder presented in Table 2.1c. In a full adder, *Carry* and *Sum* are the two output bits of interest. In Table 2.2c, two outputs, $Carry_{out}$ and *Sum*, represent the carry and sum bits of a full adder. The garbage outputs do not contribute to the property of the sum or carry output bits. The additional input $Constant_{in1}$ and the two outputs, $Garbage_{out1}$ and $Garbage_{out2}$ are used to maintain the bijective relation between the inputs and the outputs of this reversible full adder operation. Constant inputs and garbage outputs are often used as cost metrics of reversible circuits. The cost metrics in reversible logic are discussed further in section 2.3.

2.2 Logic Gates

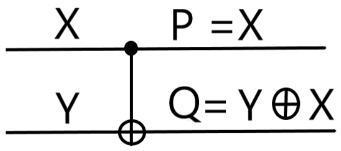
Logic gates are used to perform logic operations. Gates are combined (cascaded) to implement the desired logic functions. Therefore, logic gates are the key components of a circuit. This section introduces the reversible logic gates that are used to design the fault tolerant reversible circuits of this thesis.

2.2.1 Reversible Gates

All reversible gates are bijective and maintain one-to-one and onto relationships between the inputs and outputs. The two most popular families of gates are the NCT (NOT-CNOT-Toffoli) gate family and the SF (SWAP-Fredkin) gate family [59]. This thesis uses Controlled NOT (CNOT) and Toffoli gates from the NCT family and Fredkin gates which belong to the SF gate family. These gates are introduced in the following subsections. In the following figures the $\bullet$ symbol represents the control points and the $\oplus$ symbol indicates the targets of the reversible gates. The targets ($\oplus$) of the reversible gates perform the EXOR operation. The first $n-1$ bits are known as control the control bits, and the last n^{th} bit is the target bit. The reversible gate passes the input values at controls directly to the corresponding outputs without any change and inverts the target bit if and only if all input values at controls are 1. The NOT gate is a special case of a Toffoli gate with no controls. Negative control points are shown as $\circ$ and inverts the target bit if and only if all input values at controls are 0.



(a) A reversible NOT gate.



(b) A CNOT gate.

Figure 2.1: Reversible logic gates.

2.2.2 NOT Gates

One of the simplest reversible logic gates is a NOT gate. A reversible NOT gate operation shown in Table 2.2a is identical to the traditional NOT gate operation in Table 2.1a. A NOT gate in traditional logic is the only logic gate that provides a one-to-one and onto relationship between the inputs and outputs. Thus, the logical NOT operation is reversible. The truth table for a logical NOT operation is presented in Table 2.2a.

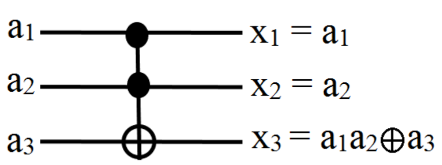
2.2.3 CNOT Gates

The controlled-NOT (CNOT) gate belongs to the NCT gate family. A CNOT gate consists of a control input and a target input. A specified value at the control input inverts the target input value. If the control input value of 1 inverts the target input, the gate is referred to as a positive-controlled gate. For a negative-controlled gate a control input value of 0 inverts the target input. For example, when both the control and target inputs are 1 a positive-controlled CNOT gate inverts the target input value to ($1 \oplus 1$) which is equal to 0.

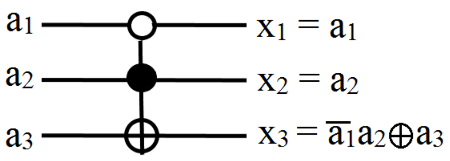
Figure 2.1b shows a positive controlled CNOT gate along with the logic function. The associated truth table is presented in Table 2.2b. A CNOT gate is also known as the Feynman gate [24].

2.2.4 Toffoli Gates

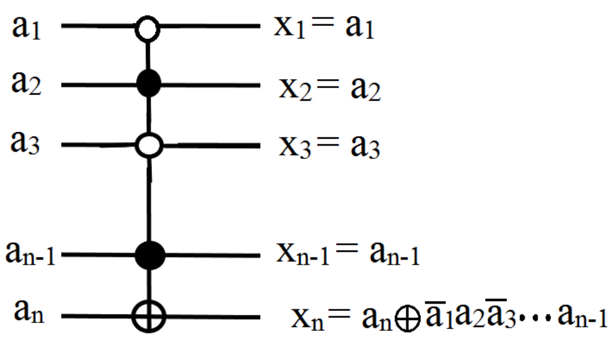
A Toffoli gate is a form of CNOT gate with multiple control points [57]. The reversible NOT gate is a special case of a Toffoli gate with $n = 1$ and no control values. When $n = 2$, the Toffoli gate is also known as the CNOT or Feynman gate. It is possible to implement the three basic logic operations (i.e. the NOT, AND and OR operations) using only Toffoli gates [57]. Thus, a Toffoli gate is considered to be a universal gate.



(a) A 3×3 positive-controlled Toffoli gate.



(b) A 3×3 negative-controlled Toffoli gate.



(c) A $n \times n$ MCT gate with positive and negative control points.

Figure 2.2: Toffoli gates.

Figure 2.2a shows a 3×3 positive-control Toffoli gate with two control points and one target point. The two input lines a_1 and a_2 are connected to the two control points and the input line a_3 is connected to the target. If the values of the two control points are 1 then the target input is inverted. Otherwise, the target input remains unchanged. For

Table 2.3: Truth table of a 3×3 positive-controlled Toffoli gate.

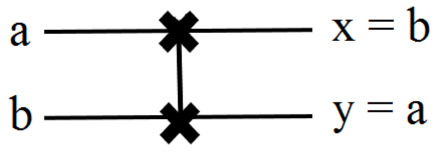
input			output		
a_1	a_2	a_3	x_1	x_2	x_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

example in Figure 2.2a, for the inputs $(a_1,a_2,a_3) \equiv (1,1,1)$, the output of the Toffoli gate is $(x_1,x_2,x_3) \equiv (1,1,0)$. The truth table of a 3×3 positive-control Toffoli gate is shown in Table 2.3.

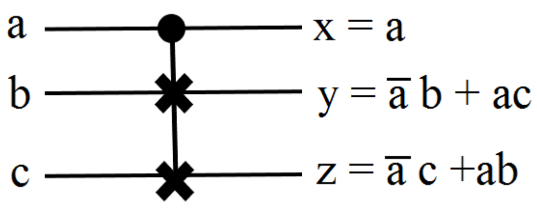
Based on the value of the control points, a Toffoli gate can also be positive or negative controlled (shown in Figure 2.2a and Figure 2.2b). The difference between a positive-controlled and a negative-controlled Toffoli gate lies in the gate operation. A positive-controlled Toffoli gate inverts the target when the control value is 1 while the negative-controlled Toffoli gate inverts the target when the control value is 0. A negative-controlled Toffoli gate may have one or more negative controls. In that case, this gate inverts the target when all the negative control values are 0. A Toffoli gate can also have multiple control points in which case it is referred to as an n-bit Toffoli gate (where $n-1$ is the number of controls and the n^{th} bit is the target) or a multiple-controlled Toffoli (MCT) gate. A MCT gate with different control points inverts the value of the target point when all of the positive controls are at 1 and the negative controls are at 0. An example of this is shown in Figure 2.2c, where the output function is $x_n = a_n \oplus \overline{a_1} a_2\, \overline{a_3} .....a_{n-1}$.

2.2.5 Fredkin Gates

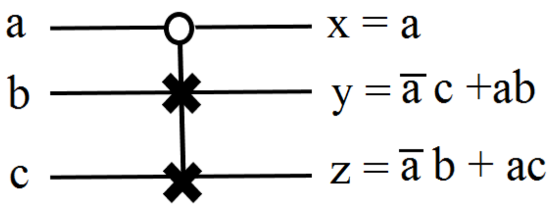
A Fredkin gate is also a universal reversible gate that performs a logic operation controlled by one or more control points [18]. Unlike the gates in the NCT family, the target lines of a Fredkin gate interchange their values depending on the value at the control points. A positive-controlled Fredkin gate interchanges the target lines if the control point value is 1, otherwise the target lines remain unchanged. Similarly, a negative-controlled Fredkin gate performs the logic operation when the control value is 0. Figure 2.3 shows different Fredkin gates. The truth table for a 3×3 positive-controlled Fredkin gate is presented in Table 2.4.



(a) A SWAP gate.



(b) A 3×3 positive-controlled Fredkin gate.



(c) A 3×3 negative-controlled Fredkin gate.

Figure 2.3: Reversible Fredkin gates.

Table 2.4: Behavior of a 3×3 positive-controlled Fredkin gate.

input			output		
a	*b*	*c*	*x*	*y*	*z*
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

A 2-bit Fredkin gate is known as a SWAP gate and is shown in Figure 2.3(a). A SWAP gate always interchanges the values of the target lines. Figure 2.3(b) and Figure 2.3(c) show a positive and a negative control Fredkin gate. If a Fredkin gate has multiple control points then the gate is called an n-bit Fredkin gate or Multiple Control Fredkin (MCF) gate. In a MCF, the gate passes the control bits to the outputs unchanged and interchanges the target bit values if all the positive control point values are 1 and all the negative control points are 0.

2.3 Cost Metrics in Reversible Logic

A number of metrics are used to evaluate the efficiency of reversible circuits. Some of the commonly used cost metrics of reversible circuits include gate count, garbage output, constant input and quantum cost which are described below:

2.3.1 Gate Count (GC)

Gate count is one of the measures used to compare and evaluate different logic circuits. Gate count refers to the number of logic gates used to implement a traditional or reversible circuit. However, gate count does not evaluate the complexity [37] of logic circuits. For example, let us consider two reversible circuits where the first circuit consists of three 3-bit Toffoli gates and the second circuit consists of two 6-bit Toffoli gates. In this case, the gate count measurement will indicate that the second circuit uses fewer gates and therefore, is preferable compared to the first circuit. However, as discussed in Section 2.3.4 a 6-bit Toffoli gate is more complex than a 2-bit Toffoli gate. Since the gates have a different number of input bits, a simple gate count measure fails to fully evaluate the complexity of the circuits. Gate count can be useful to compare different circuits consisting of similar types of gates.

2.3.2 Garbage Output (GO)

Garbage output is another measure for comparing and evaluating reversible circuits. As demonstrated in Section 2.1, it is sometimes necessary to add extra outputs in order to maintain reversibility. The values of these extra outputs are not important to realize the function of a circuit and are known as garbage outputs. For example, in Table 2.1c $Garbage_{out1}$ and $Garbage_{out2}$ are the garbage outputs. However, adding garbage outputs adds extra lines and, therefore increases the circuit width. According to Maslov [37], reducing the garbage outputs is more important than reducing the gate count.

2.3.3 Constant Input (CI)

The number of constant inputs is another metric used to evaluate and compare implementations. Constant inputs are also known as ancilla inputs [37]. Constant inputs are the input lines that are added to a function to make it reversible. The relationship between garbage outputs and constant inputs is given by the following equation:

$$inputs + ancilla\ inputs = outputs + garbage$$

2.3.4 Quantum Cost (QC)

Quantum cost is an important measure to evaluate and compare the cost of reversible circuits. Quantum cost refers to the number of 1×1 or 2×2 (1 input$-$1 output, 2 input$-$2 output) quantum gates required to design a circuit. Quantum gates are the reversible equivalent of traditional transistors and resistors. They are the micro-level building blocks of reversible gates. It is assumed that all reversible gates will be implemented at the quantum level by basic 1×1 or 2×2 quantum gates [37]. The quantum costs of the reversible gates used for the examples and in designing the fault tolerant reversible circuits in this thesis are shown in Table 2.5.

As the number of control inputs increases the quantum cost for the reversible gates also increases. For example, a 3-bit positive control Toffoli gate has a quantum cost of 5, whereas a 6-bit Toffoli gate has a cost of 61 [35]. The quantum cost calculation of an n-bit Toffoli gate is presented in [4]. According to Arabzadeh et al. [4]:

- A multiple control Toffoli gate with $n-1$ controls and the n^{th} bit as the target can be written as $C^{n-1}NOT(c;t)$, where $c = \{x_1, ..., x_{n-1}\} \subset X$ is the set of control lines

Table 2.5: Quantum cost of some reversible gates.

Size (n)	Garbage	Name	Quantum Cost
1	0	NOT	1
2	0	positive-controlled $CNOT$	1
2	0	negative-controlled $CNOT$	3
3	0	Toffoli, C^2NOT	5
4	0	Toffoli, C^3NOT	13
5	0	Toffoli, C^4NOT	29
5	2	Toffoli, C^4NOT	26
6	0	Toffoli, C^5NOT	61
3	0	positive-controlled Fredkin	5
3	0	negative-controlled Fredkin	5

and $t = x_n$ is the target line. For $n = 1$, $n = 2$, and $n = 3$ the gates are called NOT, $CNOT$, and C^2NOT or Toffoli, respectively. A $C^{n-1}NOT$ gate has an exponential cost of $2^n - 3$ only if the gate has zero garbage lines with all positive controls.

- The quantum cost of an $n-1$ negative-control Toffoli gate with at least one positive-control is the same as the cost of an $n-1$ positive-control Toffoli gate.

- If all the controls of a Toffoli gate are negative an extra cost of 2 is added if zero or $(n-3)$ garbage lines are used.

According to [35], a size n Fredkin gate can be simulated efficiently by a size n Toffoli gate and 2 CNOT gates. Thus, the cost of a size n Fredkin gate is calculated by the QC of a size n Toffoli gate + 2. The quantum cost for different sizes of Toffoli and Fredkin gates is determined from the gate implementation technique. In our calculations, the quantum cost of the generalized Toffoli and Fredkin gates is taken from [35]. However, the quantum cost for any of the gates, as given in [35], may change in the future as new technologies are developed.

2.4 Fault, Failures and Errors

Every system or module is designed for a specific output behaviour. When the system is in service its behaviour can be observed and compared with a reference system. A system failure denotes that the system is unable to perform its intended function because of an error. According to Laprie [31], “a system failure occurs when the delivered service no longer complies with the specifications”. A fault can be defined as a physical or functional flaw that occurs within some hardware or software components. A fault may occur due to the design errors, mistakes in system specification or implementation, manufacturing problems or other external disturbances. In general, faults can be classified as a) hard (permanent) or b) soft (transient). Hard faults result from the physical changes of components. A determinate fault (‘stuck at zero’or ‘stuck at one’) within a digital logic circuit can be considered to be a hard fault. Soft faults can occur due to temporary changes in the properties of logic circuits. External influences such as electromagnetic interference, noise in the power supply, or improper functioning of a circuit may result in soft faults within a system.

An error is a manifestation of a fault. In a digital system, if the logical state of an element differs from the intended value this indicates an error. It can be stated that a failure is the result of an error, caused by a fault. This could be illustrated as:

$$\textit{Fault} \rightarrow \textit{Error} \rightarrow \textit{Failure}$$

2.5 Fault Tolerance in Reversible Logic

In traditional logic, fault tolerance has become increasingly important due to the widespread use of computers. Fault tolerance is a design attribute that enables a system to perform the intended operation even in the presence of a fault. The incorporation of fault tolerance into a design is an important approach to achieve reliable performance. Approaches to fault tolerance techniques in traditional logic are described in [25]. This thesis focuses on fault tolerance in reversible logic.

One approach for achieving fault tolerance both in traditional and reversible logic is to introduce redundancy. For designing fault tolerant reversible circuits this thesis work uses the concept of redundancy. Readers are referred to section 2.6 for details on redundancy and the basic concept of our approach. The concepts of fault detection, fault location, fault recovery, and fault masking are important in the design of a fault tolerant system. These concepts are described in the following subsections.

2.5.1 Fault Detection

A fault detection process identifies the occurrences of faults within a circuit. To ensure proper functioning of the circuit, faults must be detected as early as possible [25].

2.5.2 Fault Location

The process of fault location or fault diagnosis occurs after fault detection. Fault location is the process of determining where a fault has occured (i.e. the physical location of the fault within the circuit) [25].

2.5.3 Fault Recovery

Fault recovery is the process of keeping the circuit operational or restoring circuit operation after a fault has occured. A fault recovery process may require some reconfiguration of the circuit [25].

2.5.4 Fault Masking

Fault masking is the process of preventing faults from introducing errors into the system. Common approaches to fault masking include majority voting techniques for structural redundancy or error correcting memories for correcting memory before a circuit uses the data [25].

2.6 Redundancy to Achieve Fault Tolerance

Redundancy is the duplication of information, resources, or time to increase the reliability of a system [25]. Figure 2.4 shows different forms of redundancies used in traditional logic design. The introduction of any type of redundancy in reversible circuits must maintain the one-to-one and onto relationship between the inputs and outputs. In order for a system to be fault tolerant an active hardware redundancy technique requires detection and correction of faults within the system. Alternatively, a passive hardware redundancy technique achieves fault tolerance by masking the faults at the output [25]. For details on hardware redundancy readers are referred to section 2.7.

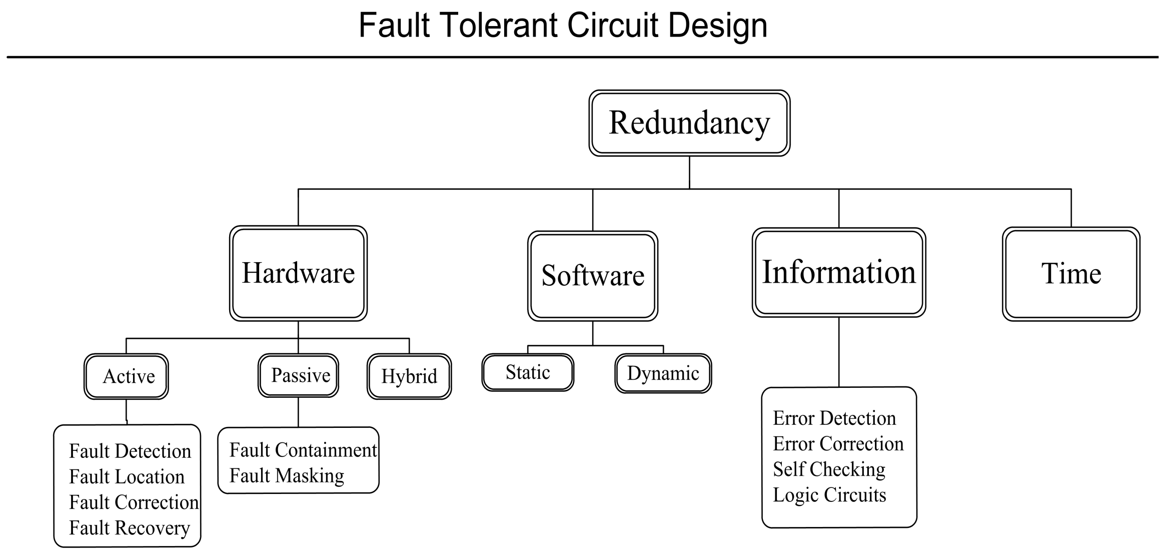


Figure 2.4: Different approaches to fault tolerance and how they are categorized [25].

Software redundancy requires the addition of extra code, which could be small routines or complete programs beyond the module requirement. Techniques used in software redundancy check correctness or the consistency of the results produced by a specific program. Sometimes the use of redundant programs eliminates the need for redundant hardware components which can also reduce the overall cost of a system [25].

The use of additional error detecting or correcting codes can be categorized as information redundancy. Error detecting codes are formed by introducing some structured code words. If the bit combinations are defined in such a way that only some of the bit combinations represent useful information then the errors can be detected. Codes can also be generated for error correction. One possible way to achieve real-time correction is to encode data using error-correcting codes (ECC) [25].

Time redundancy requires several hardware or software modules performing the same operation in parallel. In this technique the same computation is performed several times to compare the results. If the module results disagree with each other this indicates an error. Fault indication requires the error computations to be performed again. Time redundancy implementations for fault tolerance reduce hardware cost at the expense of additional time [25].

2.7 Hardware Redundancy

Hardware redundancy has become more practical due to the decreasing cost of hardware components. The most common approaches used in hardware redundancy techniques are: module replication, use of extra circuits, and majority voting mechanisms [25]. According to Johnson [25], there are three basic forms of hardware redundancy: passive, active and hybrid. Active hardware redundancy is also referred to as dynamic redundancy. An active approach attempts to detect faults and requires additional features to remove the faulty hardware components. In this technique fault tolerance is achieved using fault detection, fault location, and fault recovery. An active hardware redundancy technique often requires that the entire system be reconfigured. Reconfiguring a system may require the faulty modules to be removed and taken out of operation [25]. Thus active hardware redundancy techniques are offline fault tolerance approaches.

Passive hardware redundancy techniques use the concept of fault masking. Fault masking approaches prevent the faults from resulting in errors. The most commonly used form of fault masking is majority voting [25]. Passive hardware redundancy techniques bypass the fault and do not require any extra function for fault detection. This thesis focuses on a majority voting based triple modular redundancy (TMR) arrangement to achieve fault tolerance in reversible logic. The basic concept of TMR is described in subsection 2.7.1.

Hybrid forms of fault tolerance combine the attributes of both passive and active approaches. In hybrid fault tolerance techniques fault masking is used to prevent the errors and improved performance is achieved by incorporating fault detection, fault location, and fault recovery features.

2.7.1 Triple Modular Redundancy

Most passive hardware redundancy techniques rely on voting mechanisms for fault masking purposes. TMR is easy to realize both in traditional and reversible logic design. The basic concept of TMR is illustrated in Figure 2.5.

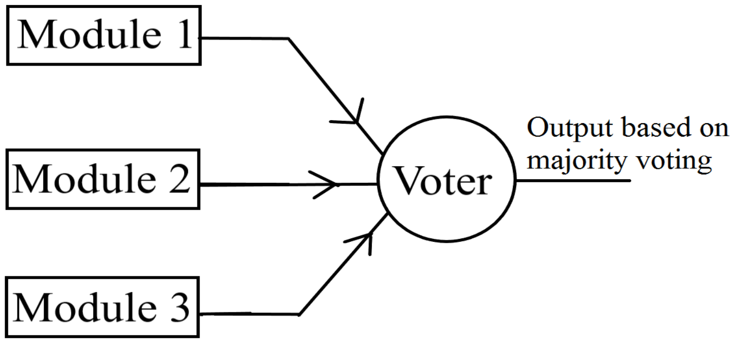


Figure 2.5: Illustration of TMR

In TMR a hardware module is triplicated and a majority vote is performed to determine the correct output. In the case of a fault within one module, it is assumed that the remaining two fault free modules provide a correct output and thus mask the fault. The concept of TMR can also be applied to mask software failures where different versions of programs execute the same function. A generalization of the TMR approach is known as *N* modular redundancy (NMR). NMR uses the same basic approach as TMR but uses N versions of a given module. To use the majority voting arrangement N is selected to be an odd integer [25]. One major difficulty with a TMR approach is the dependence on the majority voter circuit. If the voter circuit fails to execute its operation correctly, there is the risk of complete system failure.

Because TMR relies so heavily on the reversible majority voter circuit, part of this thesis focuses on the design of these circuits. We have proposed two new designs of a reversible majority voter circuit. Chapter 3 presents the existing reversible majority voters [7, 43, 61] as well as our designs to achieve fault tolerance in reversible logic.

Chapter 3

Reversible Majority Voter Circuits

This chapter proposes a passive hardware fault tolerance technique and two new designs for a majority voter circuit (MVC). A comparison of the proposed and existing designs is presented. In addition, the performance of the proposed designs is presented as well as the overall costs in terms of the cost metrics used in reversible logic.

3.1 Fault Tolerant Reversible Circuits

Most of the literature that propose fault tolerant reversible circuits provide only fault detection mechanisms [19, 23, 45, 52]. However, for a circuit to be fault tolerant the detected faults must also be corrected or masked. In this thesis we use a passive hardware redundancy technique to achieve fault tolerance in reversible circuits. Passive hardware redundancy techniques tolerate faults without the need of fault detection or system reconfiguration. Unlike active hardware redundancy, passive hardware redundancy techniques do not need to locate and correct a fault. Passive hardware redundancy techniques generally use the concept of fault masking. Fault masking in a passive hardware fault tolerant system saves the additional cost of fault location and correction, while still resulting in an online fault tolerant approach. Details on passive hardware redundancy techniques are provided in Chapter 2 of this thesis. All of the passive fault masking techniques proposed for reversible logic [7, 43, 61] require replication of modules. The most commonly used form of passive hardware redundancy is a triple modular redundancy (TMR) technique [25]. Readers are referred to Chapter 2 for details on TMR. The basic concept of TMR is to use three identical hardware modules and a voter circuit. The voter circuit provides an output with the value of the majority of the input bits. Therefore, in the case of a fault resulting in an error in the value of one of the inputs to the voter, the majority voter masks the fault and produces an output based on the remaining fault free modules. Since fault masking is done by voting rather than comparison or fault correction the entire functionality of a TMR-based design depends on the majority voter circuit (MVC) [25].

3.1.1 Reversible Majority Voter Circuits in the Literature

There are only three works in the literature that propose reversible majority voter circuits. Boykin and Roychowdhury [7] describe a reversible fault tolerant design using the concept of a traditional 3-bit repetition code. Their technique introduces a multiplexing scheme and a voting mechanism. A reversible MVC is used to generate the majority value of the input bits. The proposed voter consists of two CNOT gates and one Toffoli gate as shown in Figure 3.1. This voter provides the majority bit at the output line *x*. For example, if the input bits *abc* have the values 100 then the value at line *x* is 0, which is the value of the majority of the input bits.

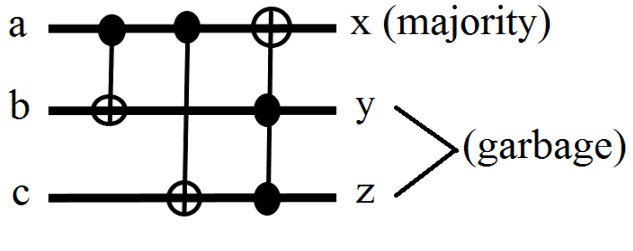


Figure 3.1: Reversible majority voter circuit [7].

The reversible multiplexing scheme proposed in [7] introduces an error recovery circuit as shown in Figure 3.2. The error recovery circuit uses six blocks of the majority voter circuit of Figure 3.1 for a total of eight operations. Functionality of the error recovery circuit is described using two phases: encoding and decoding. The first three MVC^{-1} blocks are used to encode the input bits, while the last three MVC blocks are used to decode the bits.

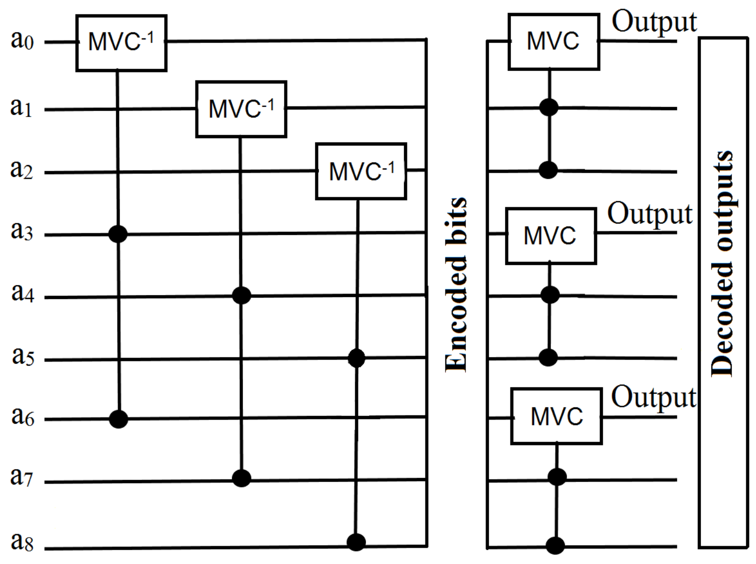


Figure 3.2: Reversible fault tolerant multiplexing scheme [7].

If an input bit has a logical value of 0 or 1, the encoding blocks map the bit to an expanded value of 000 or 111. In other words, the MVC^{-1} blocks triplicate each input bit. The MVC blocks in Figure 3.2 perform decoding and generate the majority value at each of the three designated output lines. If there are no errors, the outputs of Figure 3.2 should match the inputs. In case of a fault, at most one bit in each of the MVC outputs is altered and indicates an error. Since the MVCs in Figure 3.2 return the majority bit values of the encoded bits, a single error will not affect the majority bit values at the outputs. In practical use, this error recovery cycle would be placed strategically throughout the circuit interspersed with computations.

The MVC in Figure 3.1 uses three reversible gates and therefore, the gate count for this circuit is 3. Since the design does not require any additional inputs the number of constant inputs for this voter is zero. The design of this MVC produces 2 garbage outputs. The MVC consists of two positive-controlled CNOT gates and one positive-controlled 3-bit Toffoli gate. Thus, the total QC of this design is $(1+1+5) = 7$.

A 3-bit reversible majority voter circuit is proposed in [43]. Researchers in this study designate a specific output line to carry the majority value. The proposed design is shown

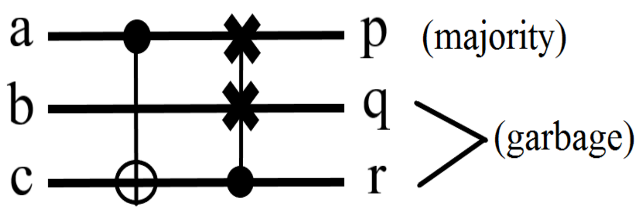


Figure 3.3: Reversible majority voter circuit [43].

in Figure 3.3. For example, let us consider an input bit combination *abc* = 011. For this combination the control bit value of the first gate is 0. This passes the bits to the second gate without any changes. The second gate has its control bit at 1 and swaps the first and second input bits. This provides the majority value of 1 at the output (on line *p*). The MVC presented in Figure 3.3 consists of a 2-CNOT and a 3- bit Fredkin gate and does not require any constant input. The gate count of this MVC is 2 with a QC of $(1+5) = 6$.

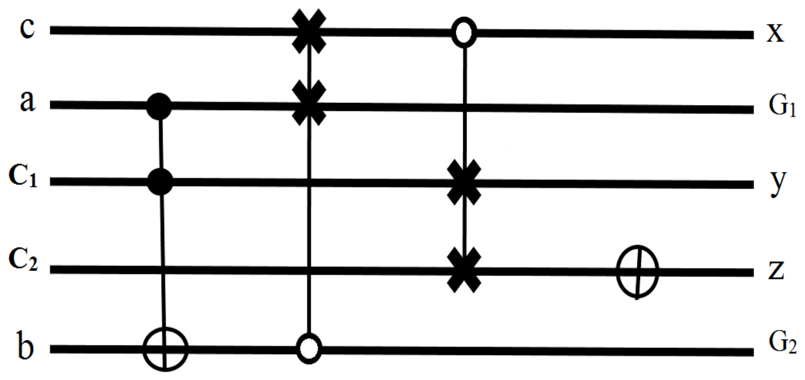


Figure 3.4: Majority voter with triplicated voter outputs proposed in [61].

Another MVC proposed in [61] takes three input bits from three identical modules and provides the majority line on all three outputs. In other words, the majority value is replicated on three output lines. In order to maintain reversibility, two control lines ($C_1 = 1$, $C_2 = 0$) are added at the voter input and two garbage lines (G_1, G_2) are added with the voter outputs. Voting is done on the data inputs (a, b and c), and the goal is to produce all 0*s* or all 1*s* at the data outputs (x, y and z). For example, if the input bits *abc* have the values

101 and the control lines are set to $C_1 = 1$, $C_2 = 0$, this generates the value 111 at the data outputs xyz as shown in Figure 3.5. The number of constant input and garbage output for

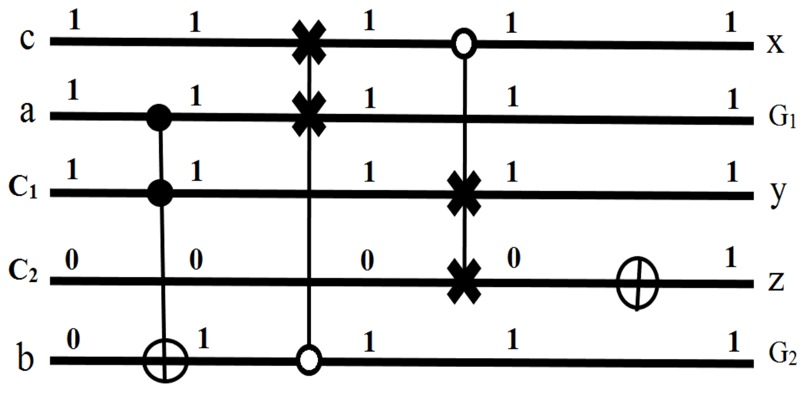


Figure 3.5: Example of input assignments and subsequent fault masking for the circuit shown previously in Figure 3.4.

the voter presented in Figure 3.4 is 2 and 2, respectively. The circuit in Figure 3.4 uses 4 reversible gates which results in a gate count of 4. The QC of the circuit in Figure 3.4 is $(5+5+5+1) = 16$.

3.2 Proposed Reversible Majority Voter Circuits

In this section we propose two designs for a 3-bit reversible majority voter. For a $n \times n$ reversible circuit we can designate any of the output lines as an output line of interest [43, 7] and consider the rest of the outputs to be garbage. The voter circuit behaviour is characterized by the truth table shown in Table 3.1.

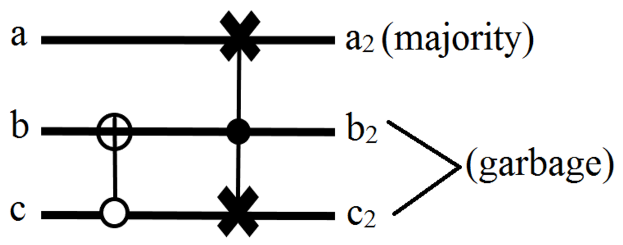


Figure 3.6: MVC with two reversible gates.

Table 3.1: Truth table of the majority voter with two reversible gates.

	after 1st gate	after 2nd gate
abc	$a_1b_1c_1$	$a_2b_2c_2$
000	010	**0**10
001	001	**0**01
010	000	**0**00
011	011	**1**10
100	110	**0**11
101	101	**1**01
110	101	**1**01
111	111	**1**11

The reversible MVC shown in Figure 3.6 consists of a negative control CNOT and a positive control Fredkin gate. The output of interest is a_2 which gives the majority of input bits (shown in Table 3.1). The other two output lines are not used and are considered to be garbage lines. To demonstrate the behaviour of this circuit, let us consider an input combination of $abc = 100$. As we can see from Figure 3.6, the control of the Fredkin gate is $(c \oplus b)$. When $c = b$, $(c \oplus b) = 0$ and the Fredkin gate does not swap the values of a and c at the output. However, when $c \neq b$ then $(c \oplus b) = 1$, and the Fredkin gate operates and swaps the values of a and c at the output. Thus, the gate count for this circuit is 2 and the design does not require any constant input. The QC of a negative-controlled CNOT gate and a positive-controlled Fredkin gate is 3 and 5 respectively [35]. Thus, the QC of the circuit in Figure 3.6 is $(3+5) = 8$.

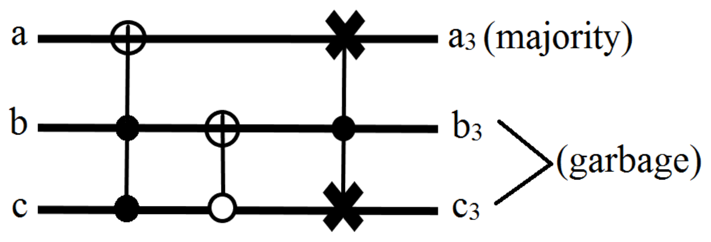


Figure 3.7: MVC with three reversible gates

Table 3.2: Truth table of the majority voter with three reversible gates.

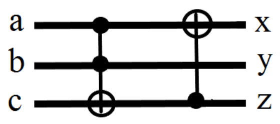
	after 1st gate	after 2nd gate	after 3rd gate
abc	$a_1b_1c_1$	$a_2b_2c_2$	$a_3b_3c_3$
000	000	010	**0**10
001	001	001	**0**01
010	010	000	**0**00
011	111	111	**1**11
100	100	110	**0**11
101	101	101	**1**01
110	110	100	**1**01
111	011	011	**1**10

The second MVC we propose consists of three reversible gates: a positive control Toffoli, a negative control CNOT and a positive control Fredkin gate. For this circuit the output of interest is a_3 and the rest are garbage outputs. The truth table shown in Table 3.2 demonstrates the bit combinations at each gate output. This 3×3 reversible majority voter circuit provides the same output as our design proposed in Figure 3.6. However, the MVCs presented in Figure 3.6 and Figure 3.7 differ in their fault tolerance mechanism. The functional difference of the proposed voters is described in Section 3.5.

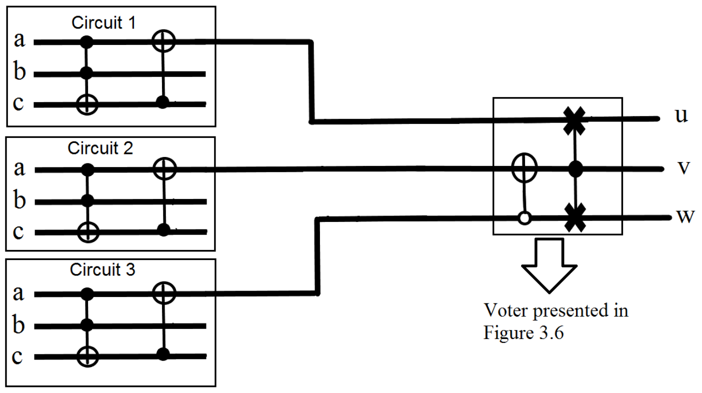
The gate count for this circuit is 3 and the design does not require any constant input. The QC of a positive-controlled Toffoli gate, a negative-controlled CNOT gate and a positive-controlled Fredkin gate is 3, 5, and 5 respectively [35]. Thus, the QC of the circuit in Figure 3.7 is $(5+3+5) = 13$.

3.2.1 Fault Tolerant Reversible Circuit Design

Our proposed voter circuits can be used with TMR to achieve fault tolerance for any reversible circuit. For example, let us consider a 3×3 reversible logic circuit consisting of a 2-bit Toffoli and a CNOT gate. This circuit is shown in Figure 3.8a where x is the output of interest and y and z are garbage outputs. In accordance with the basic principles of TMR we need three versions of the circuit in Figure 3.8a for designing a fault tolerant reversible circuit. The three output lines of the three copied circuits are then connected to



(a) A reversible circuit.



(b) Fault tolerant design of the circuit using TMR.

Figure 3.8: Fault tolerant reversible circuit design

any of our proposed MVCs. In this case, let us consider the two-gate voter circuit shown in Figure 3.6. If any one of the modules is faulty, the majority voter masks the fault and sends the corrected output on line *u*. A fault tolerant design of the circuit in Figure 3.8a is shown in Figure 3.8b. A full adder circuit is considered to be a fundamental building block of many computational logic units. Figure 3.9 shows a fault tolerant full adder design using two majority voter circuits. For a full adder circuit we have two functional outputs: *Sum* and *Carry*. Output lines carrying the *Sum* from the triplicated modules are connected to the majority voter 1 and the *Carry* output lines are connected to the majority voter 2. To assure fault tolerance, the circuit needs to mask the fault at both of these two output lines. Corrected *Sum* and *Carry* bits are generated at *Corrected Sum* and *Corrected Carry* output lines as shown in Figure 3.9. Figure 3.9 also demonstrates the use of both of our proposed voter circuits in this example, although we note that any voter circuit could be used in either place.

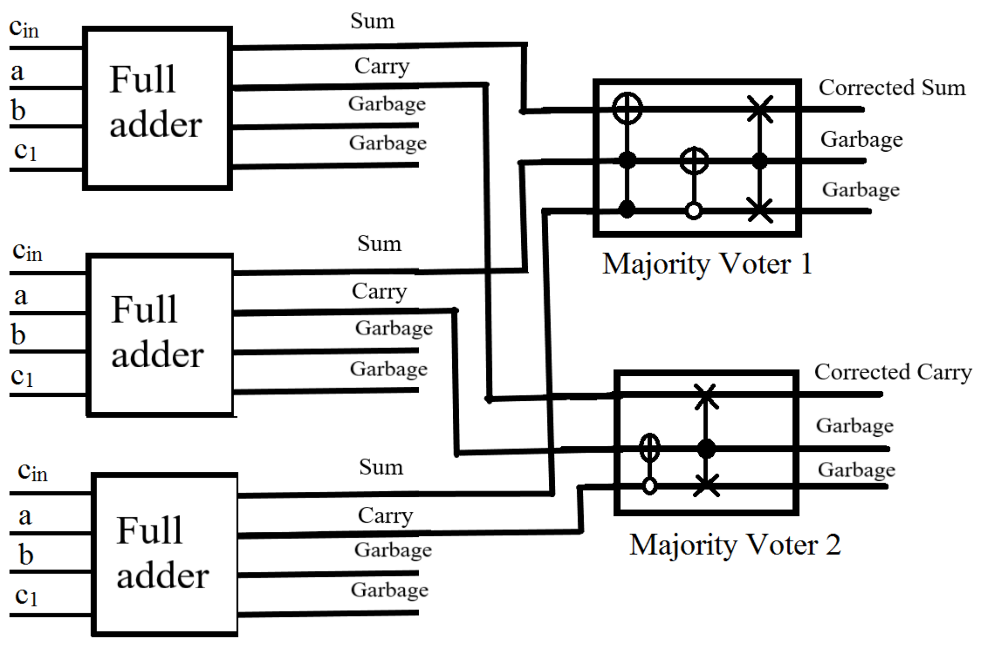


Figure 3.9: Reversible fault tolerant full adder circuit design.

3.3 Voter Circuit Comparisons

Reversible logic design evaluation is based on parameters such as the number of constant inputs (CI), the number of garbage outputs (GO), quantum cost (QC), and gate count (GC) [24]. These are often referred to as cost metrics. Readers are referred to Chapter 2 for details on cost metrics for reversible circuits. In this section, we compare our proposed majority voter circuits to the existing designs presented in [7, 43, 61] respectively. Our comparisons are shown in Table 3.3.

Table 3.3: Comparative results of different majority voter circuits.

	GC	GO	CI	QC
Proposed design in Figure 3.1 [7]	3	2	0	7
Proposed design in Figure 3.3 [43]	2	2	0	6
Proposed design in Figure 3.4 [61]	4	4	2	16
Proposed design in Figure 3.6	2	2	0	8
Proposed design in Figure 3.7	3	2	0	13

One of the goals in reversible logic design is to minimize the number of reversible gates used in a circuit. Our first design proposes a majority voter circuit requiring two reversible gates while the second design requires three reversible gates. This is comparable with the existing approaches, all of which require equal or more gates for designing a reversible majority voter circuit. Although the majority voter circuit in [7] is more cost efficient, achieving fault tolerance using the approach in [7] is expensive, as detailed in subsection 3.1.1. However, it would also be feasible to use the voter circuit in [7] in a TMR-based arrangement to achieve a fault tolerant design.

The comparative results presented in Table 3.3 show that the design in Figure 3.4 [61] has higher cost metrics (i.e. GC, GO, CI, and QC) compared to our proposed designs and the other existing approaches. The MVC in Figure 3.3 [43] is similar to our first design which is presented in Figure 3.6. Both of the designs have gate counts and garbage outputs equal to 2 and require 0 constant inputs. However, the quantum costs of our proposed design in Figure 3.6 is higher than the design in Figure 3.3 [43].

Our second approach for designing a reversible MVC is presented in Figure 3.7. The number of garbage outputs and constant inputs of this circuit is equal to our first design presented in Figure 3.6 and the designs proposed in Figure 3.1 [7] and Figure 3.3 [43]. The gate count for the circuit in Figure 3.7 is 3 with a quantum cost of 13, which is higher than our first design and the designs in [7, 43]. However, as we describe in Section 3.4, under certain circumstances the voter circuit presented in 3.7 offers better fault masking capability than the lower cost designs.

3.4 Voter Circuit Application in Reversible Fault Models

Faults can lead circuits' to a system failure and therefore, must be detected and corrected at an early stage. A fault can be categorized under one of the following assumptions: a single fault assumption or a multiple fault assumption. A single fault assumption is a situation where at most one fault is present in a circuit. If there are two or more faults at the same time then the multiple fault assumption is used.

When using TMR, the proposed voters in Figure 3.6 and Figure 3.7 can mask any single fault that may occur within the triplicated circuits. Moreover, we demonstrate that the design proposed in Figure 3.7 can mask any single fault if the fault occurs in particular locations inside the voter circuit. For details of these fault masking mechanisms readers are referred to Section 3.5. The following subsections define different single faults available in reversible logic. We demonstrate that our proposed circuits can generate the corrected output in the event of a single bit fault, a single gate fault, a cross point fault and the family of missing gate faults.

3.4.1 Fault Masking of a Single Bit Fault

In a single bit fault (SBF) exactly one output value of a circuit is altered from the fault-free value [49]. A SBF assumes that the fault is anywhere in any circuit line and alters the value of that line from 0 to 1 or vice versa. Figure 3.10 shows a TMR arrangement of a reversible circuit where the outputs of the triplicated circuits are connected to the MVC from Figure 3.6. The schematic of this design and example values on each line are shown in Figure 3.10. To demonstrate the fault masking mechanism, let us consider 110 as inputs to the triplicated reversible circuits. As shown in Figure 3.10, 110 as the inputs should result in a value of 0 at the output. If a SBF occurs at any of the circuits' outputs the fault alters the value and produces an error. The SBF is then masked by the voter circuit and the majority value from the other two fault-free circuits is sent as the voter's output.

3.4.2 Fault Masking of a Single Gate Fault

A single gate fault (SGF) occurs when an entire gate fails completely or becomes inactive [49]. As a result, the input value of the affected gate remains unchanged at the output. Figure 3.11 shows another example of TMR similar to that in Figure 3.10. In this example we use the proposed MVC from Figure 3.7. We again assume that the values 110 are the inputs to the triplicated circuits, which should produce a 0 at the circuits' output lines. If the second gate of Circuit 1 in Figure 3.11 fails then the values 111 are passed to the voter

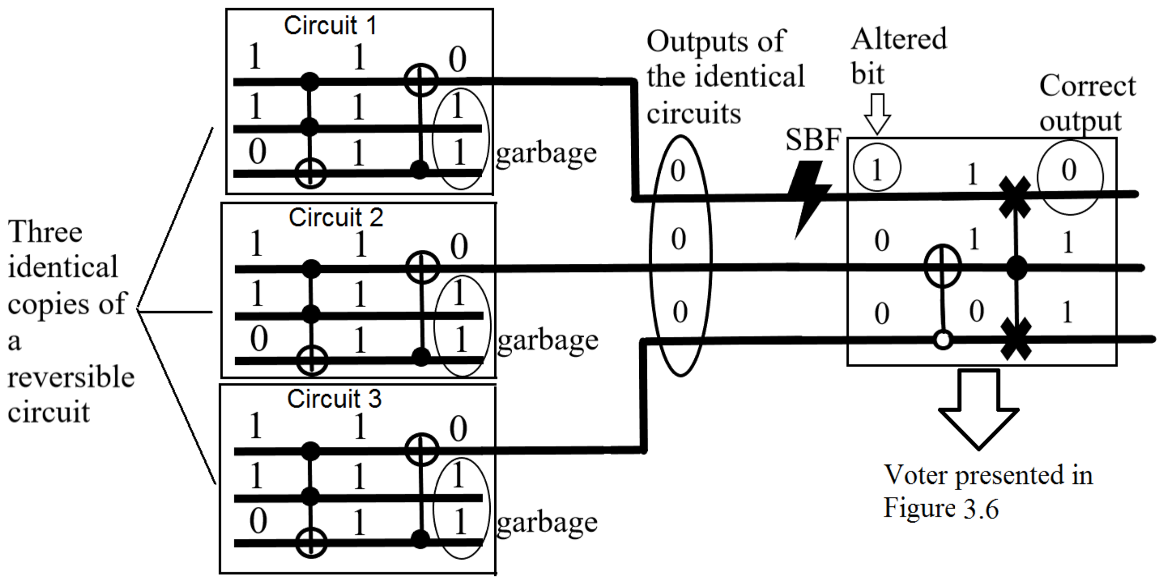

Circuit 1
Circuit 2
Circuit 3
Three identical copies of a reversible circuit
garbage
Outputs of the identical circuits
SBF
Altered bit
Correct output
Voter presented in Figure 3.6

Figure 3.10: Fault masking of a SBF.

circuit. This results in a value of 1 at the output line of Circuit 1 where the error-free versions of the same reversible circuits produce a value of 0. However the MVC masks the error, and the correct value is given on the output of interest.

3.4.3 Fault Masking of Crosspoint Faults

A crosspoint fault occurs if the existing control points of a reversible gate are missing or any extra control point is added to the gate [49]. Crosspoint faults are classified as appearance faults and disappearance faults. An appearance fault occurs when extra control points are added and a disappearance fault occurs when one or more control points of a gate become inactive. Figure 3.12 shows an example of appearance and disappearance faults in a reversible circuit. Control points of a reversible gate determine whether a gate performs its intended operation on the target inputs. For example, a 3-bit positive control Fredkin gate interchanges the values of the target inputs when the control value is 1. If the control value is missing then the interchange of the target values will take place even when it should not (i.e. interchange in the target values when the value of the control line is 0, assuming a positive control Fredkin gate). Thus, a missing control point results in a disappearance (crosspoint)

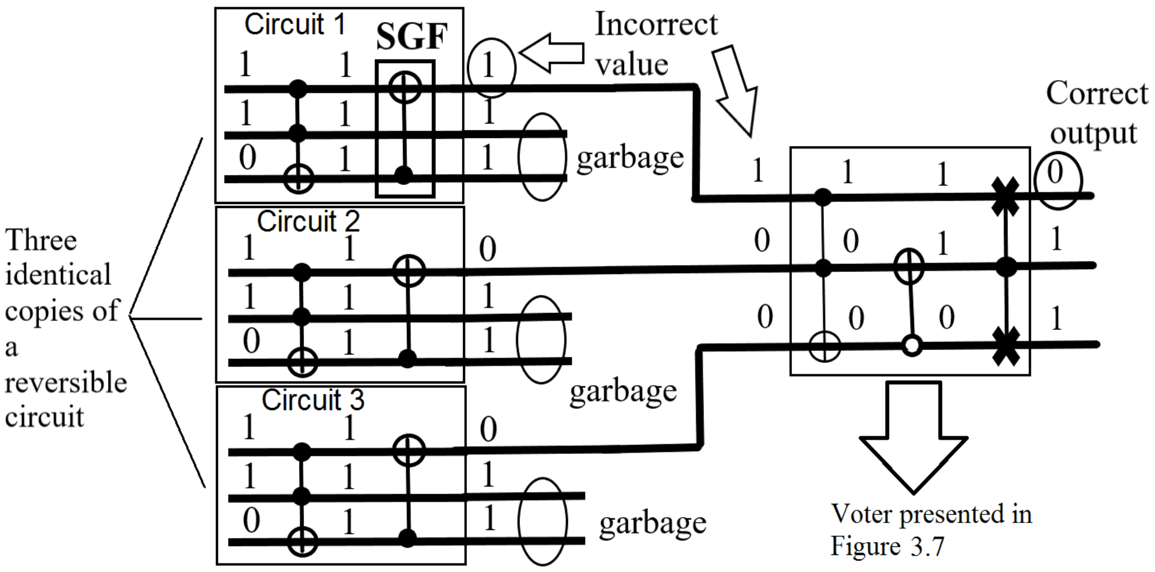


Figure 3.11: Fault masking of a SGF.

fault and provides an incorrect output. Figure 3.13 illustrates a TMR arrangement where the Fredkin gate of Circuit 1 is affected by the crosspoint (disappearance) fault. In this example, the input values of 101 should produce an output of 100. However, the disappearance fault in Circuit 1 causes it to produce 001. The faulty bit is then carried over to the MVC where the faulty value is masked.

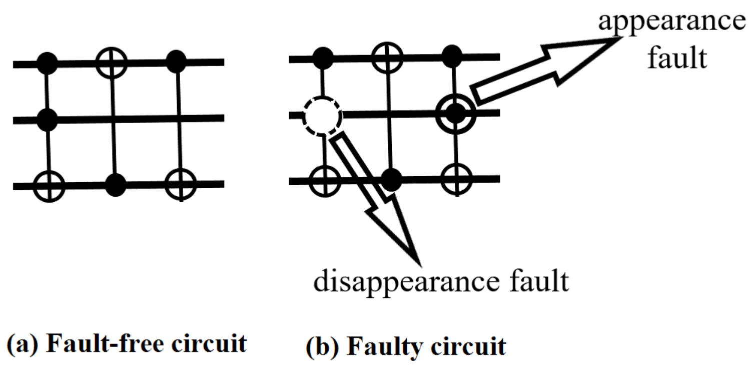


Figure 3.12: Crosspoint faults in a reversible circuit.

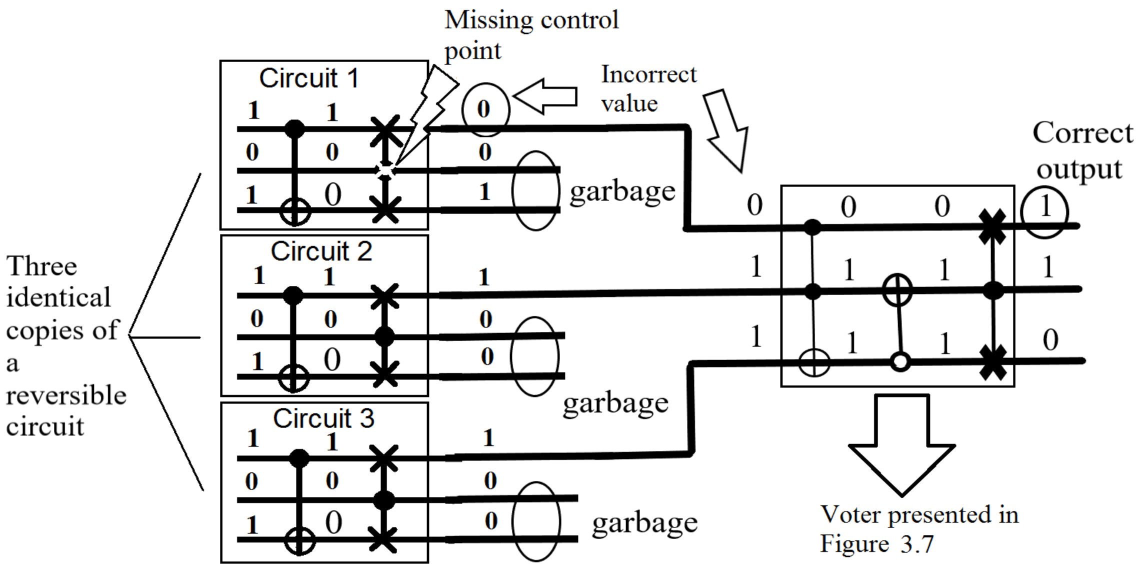


Figure 3.13: Fault masking of a disappearance fault.

3.4.4 Fault Masking of the Family of Missing Gate Faults

The family of missing gate faults includes four different faults [49]:

- the single missing gate fault (SMGF): a fault that is caused by the disappearance of an entire gate;
- the repeated gate fault (RGF): a fault resulting from the unwanted repetition of a gate within a circuit;
- the multiple missing gate fault (MMGF): a fault that occurs when a number of gates are missing from a circuit and
- the partial missing gate fault (PMGF): a fault caused by missing control points of a gate.

As described in Sections 3.4.1-3.4.4 the proposed design will mask any type of fault including a single bit fault, a single gate fault, a crosspoint fault and the family of missing gate faults, as long as it is a single fault resulting in only one of the input lines to the voter having an incorrect value. Either of our proposed voters may be used. We discuss faults within the voter itself in Section 3.5.

3.5 Fault Masking within the Voter Circuits

This section presents the design and functional differences of our proposed majority voter circuits presented in Figures 3.6 and 3.7. In Section 3.4 we have demonstrated that when using TMR, our proposed voters can correct/mask a single fault. We assume that the single fault occurs in one of the triplicated circuits and affects a single input to the voter circuit. However, a fault may also occur inside the MVCs. We can demonstrate that for a specific input bit combination of 011 and two particular locations inside the voter circuits, the design presented in Figure 3.7 offers better fault masking capability than the existing designs. In Figures 3.14−3.19 voter circuits are partitioned into two parts labeled Stage A and Stage B. We introduce these stages for design comparison and demonstrate how a fault between the stages impacts the functional behaviour of the different MVC designs.

First, let us consider the examples shown in Figures 3.14 and 3.15. In these examples we assume that the inputs to the MVCs are 011. If any single fault occurs between the stages, the design presented in Figure 3.6 produces a faulty output. However, the design in Figure 3.7 includes an additional Toffoli gate before Stage A. This results in the fault being masked at the voter's output. Although the circuit presented in Figure 3.7 has higher cost metrics than the design in Figure 3.6, the additional cost could be viewed as being offset by the better fault masking capability. This can be noted for the designs shown in Figure 3.1 [7] and Figure 3.3 [43], as illustrated in Figures 3.16 - 3.19.

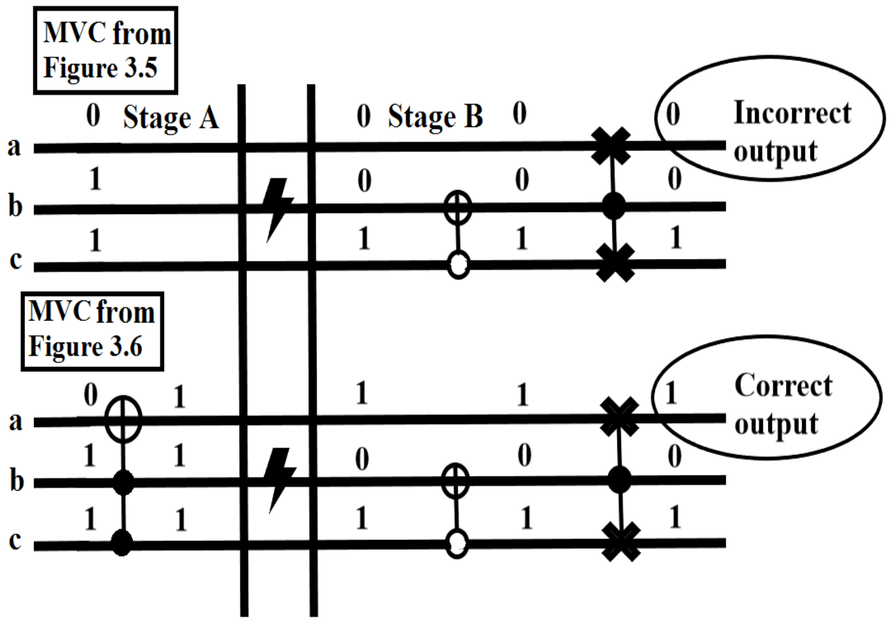


Figure 3.14: Voter circuit operations in the presence of a single fault on line b.

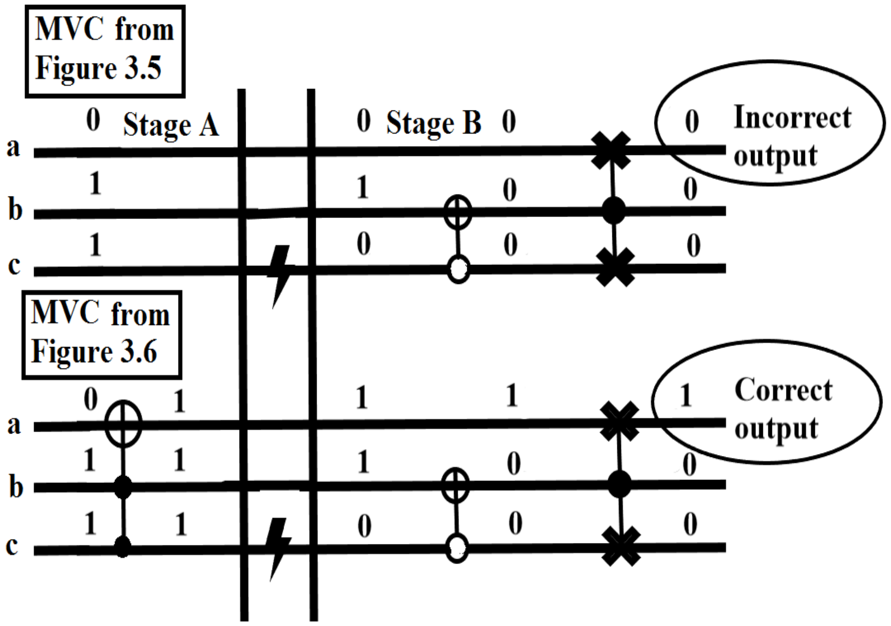


Figure 3.15: Voter circuit operations in the presence of a single fault on line c.

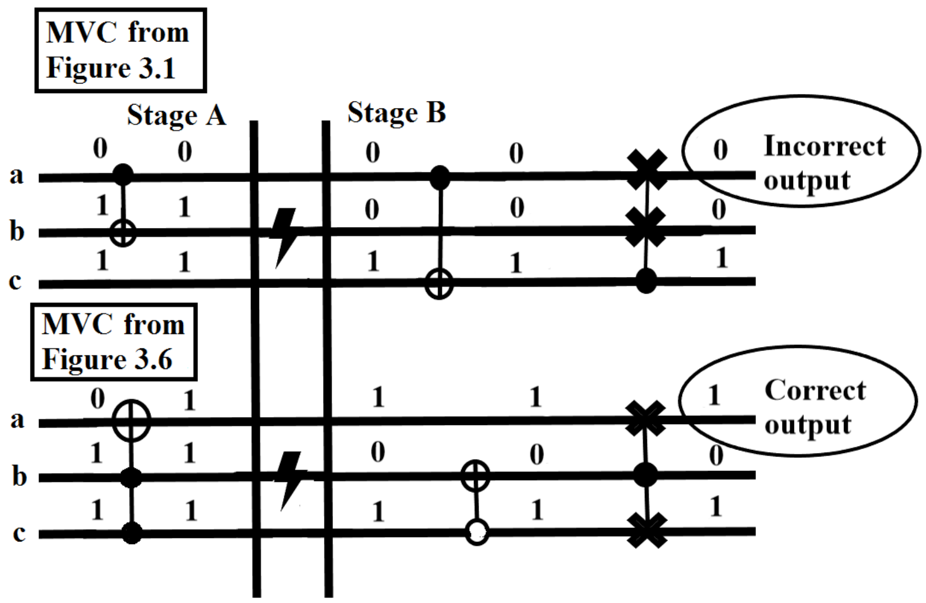


Figure 3.16: Voter circuit operations in the presence of a single fault on line b.

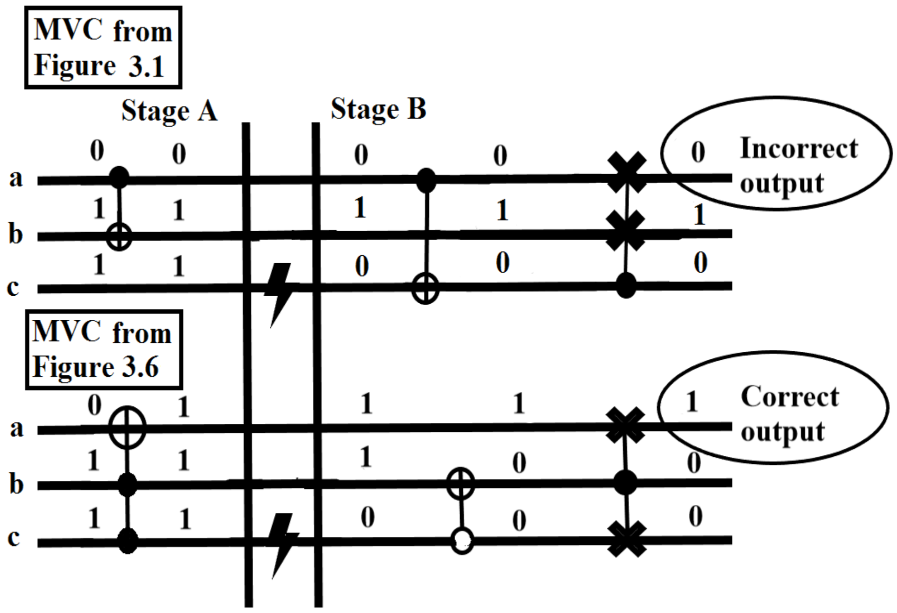


Figure 3.17: Voter circuit operations in the presence of a single fault on line c.

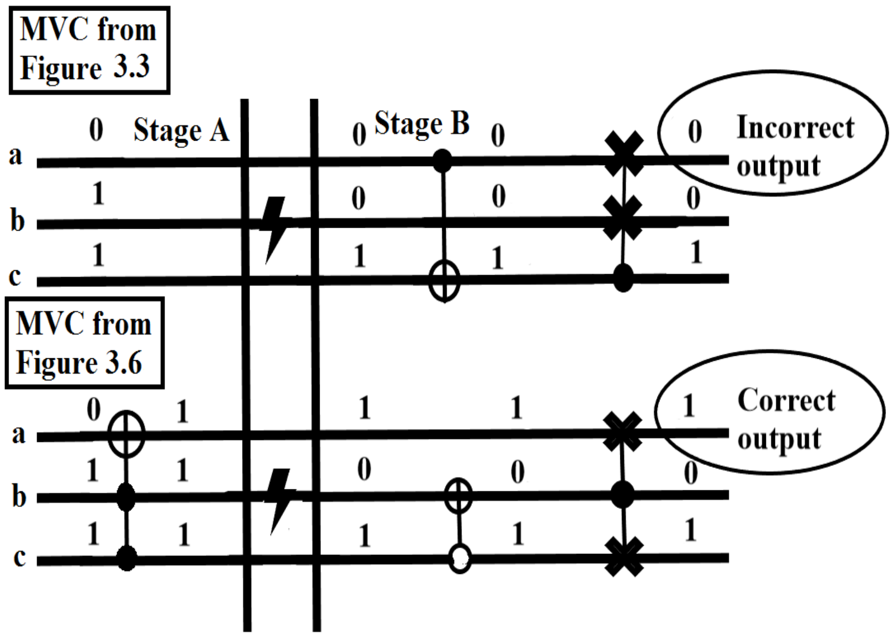


Figure 3.18: Voter circuit operations in the presence of a single fault on line b.

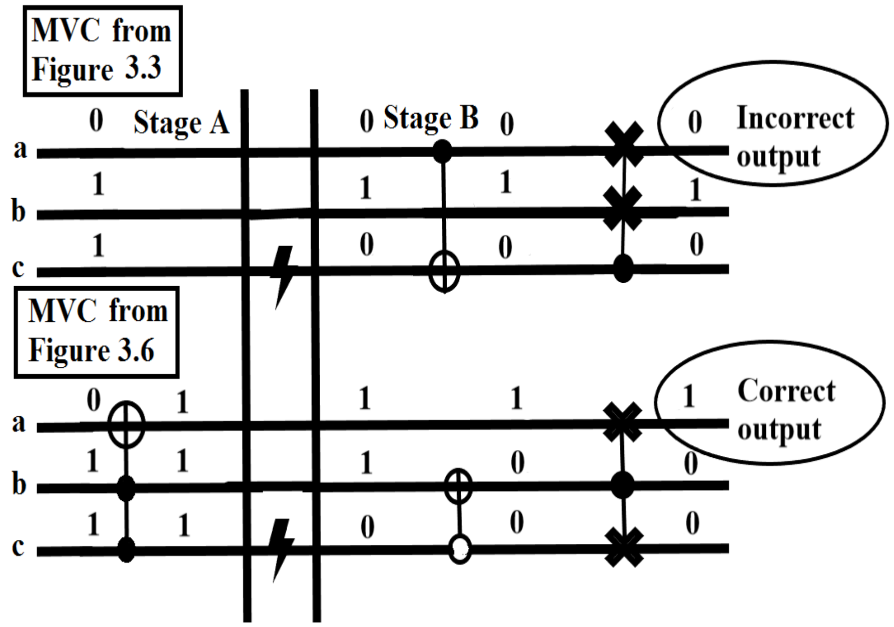


Figure 3.19: Voter circuit operations in the presence of a single fault on line c.

Chapter 4

Failure Probability Analysis

This chapter describes the mathematical analysis for determining failure probabilities of the majority voter circuits proposed in Chapter 3. We review the literature on traditional circuit reliability and failure probability. We also determine the failure probabilities of our designs over different numbers of trials using the basic concepts of set theory and binomial distribution.

4.1 Approaches to the Reliability and Failure Probability Analysis

Reliability and failure probability analysis are not new concepts in traditional logic design. Reliability is defined as the probability that a component or system experiences no failures during a specific time interval. Conversely, the failure probability of a component or system is defined as the probability that the component or system experiences failure one or more times during a specific time interval. Failure probability is often referred to as unreliability. Once the reliability is defined, the failure probability or unreliability can be determined using the following equation:

$$Reliability + Unreliability = 1 \tag{4.1}$$

Reliability and failure probability analysis in traditional logic is carried out to assure reliable performance of the logic circuits. However, reliability and failure probability analysis has not yet been considered for reversible logic circuits. In this thesis, we propose a macro gate level circuit failure probability analysis for reversible logic circuits. To our best knowledge, this is the first work for a circuit failure probability analysis based on the comprising gate failure probabilities in reversible logic designs.

4.1.1 Reliability Analysis in Traditional Logic

As device dimensions are shrinking, component failure probabilities are increasing. Issues affecting digital circuit reliability include manufacturing imprecision, environmental factors and process variations (e.g. power supply variations, device mismatch) [10]. In CMOS technology, designers face challenges in designing nano-scaled devices due to quantum effects and power dissipation. Other non-conventional circuit technologies include single electron tunneling technology, carbon nanotubes and quantum dot automata. Electronic devices based on these non-conventional technologies are more sensitive to external noises and require different operating conditions (e.g. low temperature). This has led researchers to focus on improving circuit reliabilities and determining the failure probability of these circuits [9].

Traditional approaches to improve circuit reliability include triple modular redundancy (TMR) [32] and concurrent error detection (CED) [53] techniques. However, recent advancements in the field of reliability studies use probabilistic transfer matrices (PTM) [29] and Bayesian networks [48]. In [13], Choudhury and Mohanram propose probabilistic theorems for reliability analysis of traditional logic circuits. The theorems are based on logic synthesis and testing approaches. Proof of correctness and applications of the proposed reliability analysis are verified by simulation results for several benchmark circuits.

Reliability analysis of unreliable devices is reported by Chen and Xiao [10]. Their analysis is based on the input probabilities and gate reliabilities of traditional logic circuits. The statistical approach in [10] determines a circuit's output reliability by adding together the reliability of each unreliable logic gate that makes up the circuit. The output reliability of each gate is expressed as a function of input lines and the gate's reliability. The accuracy of this approach is estimated through a simulation study based on traditional benchmark circuits.

4.1.2 Failure Probability Analysis for Traditional Logic Circuits

Probability distribution of the time-to-failure data of a device can be defined in terms of a failure rate function. Reliability studies for traditional logic circuits define the failure rate function of mechanical and electrical components by using the ‘bathtub shaped curve’ [58]. A bathtub shaped curve is used to describe different failure modes of a component as described in [42]. To explain the bathtub shaped failure rate several models have been proposed such as [58, 60]. In [58], Wang showed that by using a simple probability plot technique we can estimate different parameters to determine mean time to failure (MTTF), burn-in time, and the replacement time of a component. Studies on a few practical models for the bathtub shaped failure rate function are presented in [60].

It is recognized that for most circuits, failure distribution can be determined based on the component’s degradation data [11]. For example, degradation data for 100 CG3A transistor units are studied in [11]. Researchers in [11] assume that the degradation process is a continuous function which describes the degradation mechanism over time. When a component’s degradation level increases to a fixed critical threshold, the elapsed time is considered to be the lifetime of the component. During the test, degradation for the 100 transistor units was observed to increase over the increasing operational hours. Similar observations in lifetime distribution based (LDB) analysis are noted in [33], where the researchers have used a least-squares-based two stage method for the analysis. However, researchers in [11, 21, 60, 42] concluded that the Weibull distribution is appropriate for the lifetime distribution analysis of the CG3A transistor series.

The Weibull distribution analysis for predicting transistor reliability as a function of nuclear radiation exposure is reported in [46]. In the presence of integrated neutron exposure, ac gain degradation of seven different types of transistors is plotted on a Weibull graph. The Weibull graph shows that the probability of transistor failure rates ranges from

0.1 to 0.01%. During the radiation exposure experimentations most of the transistor failure probabilities increased to more than 50%. The purpose of this experimentation was to justify the suitability of using different series of transistors in electronic devices for particular application areas. This study concluded that due to the changes in a transistor's material property (e.g. resistivity) radiation exposure can increase the failure probability and wear out the operating transistors.

Researchers in [1, 41, 42] have modelled and analysed the failure probability of traditional nanoscale SRAM cells by varying different parameters such as the access time failure, read/write stability failure and hold stability failure. Their analysis shows that a SRAM cell can fail due to any of the failure mechanisms (e.g. access time, read, and write failures) described above [1].

Failure probability analysis of traditional logic circuits is carried out based on the semiconductor degradation data, or the failure rate data over time [11, 21, 60, 42]. Depending on different parameters, [34, 12, 13] show that the traditional gate fault probabilities range from 0.001 to 0.1%. Due to the unavailability of the failure probability data of reversible logic gates, we had to review the data available for traditional logic gates. In this thesis, we propose approaches to calculate the voter circuit failure probabilities based on the reversible gate fault probabilities. In our analysis, we assume that the reversible gate fault probability data varies within the range of 0.001 to 0.005%, which is comparable to the gate fault probabilities in traditional logic.

4.2 Background and Definitions

The following subsections describe the basic concepts required for the failure probability analysis of our proposed MVC designs.

4.2.1 Set Theory

Set theory is a branch of mathematical logic that studies collections of objects. A collection of distinct objects or events is defined as a *set*. Set theory introduces a fundamental binary relation between an object and a set. The binary relation identifies whether the object belongs to or does not belong to a specified set. If an object a is a member or element of set U, this is expressed as $a \in U$. The laws of set theory define the operations of union and intersection and relations of set equality and set inclusion. The set operations provide a systematic approach for evaluating mathematical expressions and performing calculations. For details on the fundamental rules of set theory readers are referred to [28]. A common approach to visualize the occurrences of multiple events is to use a Venn diagram. A Venn diagram shows all possible relations among a collection of sets [3].

4.2.2 Mutually Exclusive and Collectively Exhaustive Events

Probability is a measure of how likely an event is to occur under a given set of conditions. In probability theory, two events are said to be mutually exclusive if and only if the events cannot occur at the same time. Two mutually exclusive events have no shared outcomes. The most common example of two mutually exclusive events can be illustrated through the outcome of a single coin toss, which results in the occurrence of either heads or tails. The events are called mutually exclusive since the two potential outcomes of heads or tails cannot occur at the same time (i.e. from the same coin toss). If we consider the events to be the members of a set, then the intersection of the events will result in an empty set [47]. For example, two mutually exclusive events, A and B can be denoted by the formula $A \cap B = \varnothing$.

A set of events is collectively exhaustive if the conditions require at least one of the events to occur. In the case of collectively exhaustive events, the union ($\cup$) of the events must cover all possible occurrences within the entire sample space [47]. In the coin toss example for mutually exclusive events, both outcomes are collectively exhaustive in theory.

This is because at least one of the events must occur (head or tail).

4.2.3 Probabilities of Event Occurrences

In a sample space, the probability that an event occurs can be determined by the following equation:

$$P(event) = \frac{A}{T} \tag{4.2}$$

where

• P(event) is the probability of a favourable outcome,

• A is the number of favourable outcomes, and

• T is the total number of events.

In probability, a favorable outcome is the outcome of interest. In a coin toss event, the possible outcomes are either heads or tails. Suppose the coin is flipped twice and we want to determine the probability of getting heads both times. In this case, getting heads both times is the favorable outcome. This can be illustrated as:

• favorable outcomes: HH, and

• possible outcomes: HH, HT, TH, TT

Thus, the probability that tossing the coin twice will result in heads both times is $\frac{1}{4} = 0.25$.

4.2.4 Probability of Occurrence of At Least One Event

If two events are not mutually exclusive, then simply adding the probability of individual events does not accurately calculate the probability that either event occurs. For events that are not mutually exclusive, we must add the probabilities of the events together and subtract the probability of the intersection of the events [14]. The intersection of the events is subtracted to avoid double counting of some elements. For example, if event A and event B are not mutually exclusive, then the probability that at least one event occurs can be determined by the equation $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$. This can also be written as:

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) = P(A) + P(B) - P(A) \cdot P(B) \tag{4.3}$$

where the union of the sets is denoted by $\cup$ and the intersection of two sets is denoted by $\cap$. The Venn diagram showing the probability of occurrence of at least one event is presented in Figure 4.1.

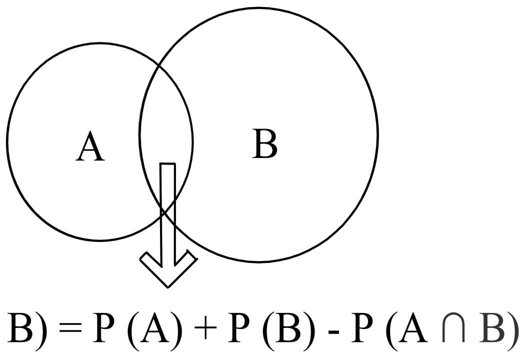


Figure 4.1: Probability of occurrence of at least one event.

In Case III of Section 4.3, we have used this concept where at least one gate being mising or inactive causes the entire voter circuit to fail. In this case, the voter circuit failure probability is calculated using equation 4.3.

4.2.5 Binomial Distribution

In probability theory and statistics, the binomial distribution is used when there are exactly two independent outcomes of each experimental trial. The outcomes of such a trial are generally labelled as *success* and *failure*. The occurrence of a single success (or failure) event is called a Bernoulli trial, and for a single trial (i.e. $N = 1$), the binomial distribution is the probability distribution for a single Bernoulli trial. In general, the binomial distribution gives the discrete probability distribution for obtaining exactly k successes out of N Bernoulli trials [14]. Equation 4.4 is the formula used to find the binomial distribution. Thus, the probability of obtaining k successes in N trials is:

$$P_p(k|N) = \binom{N}{k} p^k (1-p)^{N-k} \tag{4.4}$$

where

• k is the number of successes,

• N is the number of independent experiments,

• p is the probability that the result of a Bernoulli trial is true, and

• *(1-p)* is the probability that the result of a Bernoulli trial is false.

In equation 4.4, $\binom{N}{k}$ is a binomial coefficient. The binomial coefficient $\binom{N}{k}$ is the number of ways of selecting k outcomes from N possibilities. Using the concept of binomial distribution, we have analysed the voter circuit failure probabilities over varying numbers of trials in Section 4.4.

4.3 Proposed MVC Failure Probability Analysis Techniques

This section analyzes the voter circuit failure probability for all possible voter inputs. Figure 4.2 shows our proposed MVC designs divided into sections labeled Stage A and Stage B. The stages are introduced for design comparison purposes. Readers will notice similar gate orientation at Stage B for both of the designs presented in Figure 4.2. In Chapter 3, we have demonstrated that the Toffoli gate in our second design offers fault masking capability for input 011. If the Toffoli gate in Stage A is missing then the circuit is identical to our first MVC design and continues to perform the intended operation.

However, if any of the gates in Stage B are missing or inactive the voter fails to perform the intended operation for different input combinations. For all possible inputs, the voter circuit behaviour for missing or inactive gates is shown in Appendix A. For both of our proposed designs (presented in Figure 3.6 and Figure 3.7) only the missing or inactive occurrences of the two gates at Stage B determine the circuit failure probability for all possible inputs. This allows us to use the same mathematical analysis for both of our proposed MVCs.

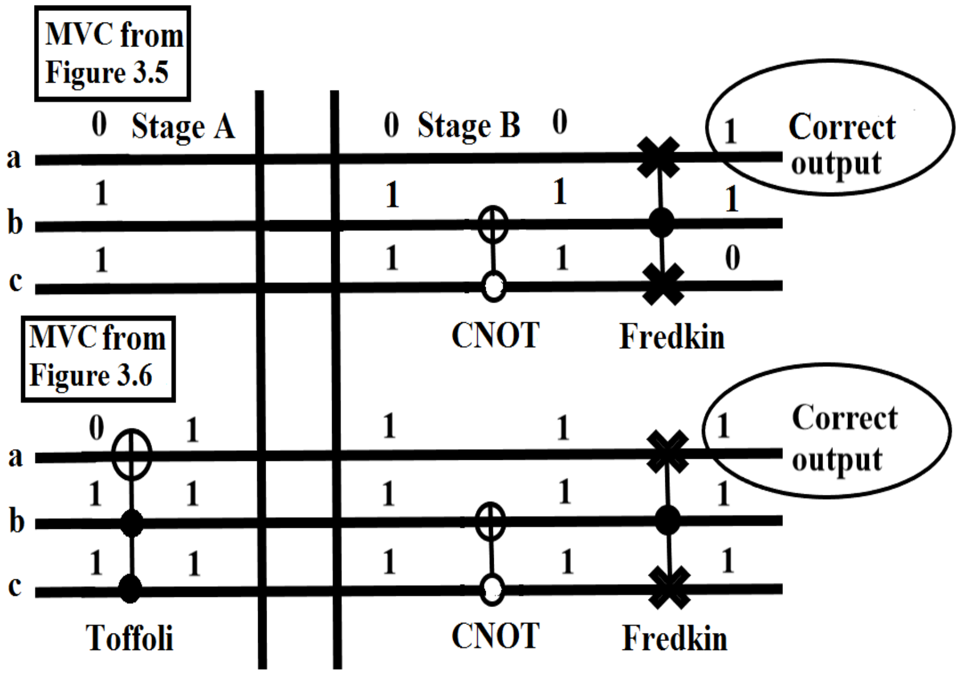


Figure 4.2: Design comparison of the proposed MVCs.

Researchers in [34, 12, 13] show that the overall circuit failure probability directly depends on the gate failure probabilities in a digital circuit. We extend this concept to reversible circuits and determine our proposed MVC failure probability from the probabilities of missing or inactive gate occurrences. For the purpose of analysis, we assume that the probabilities of missing or inactive gate occurrences are independent (i.e. the occurrences do not depend on each other). Our investigation identifies that there are exactly three ways in which the outcome can become incorrect: if the input is 110 and the CNOT gate fails but the Fredkin gate does not fail; if the input is 011 and the Fredkin gate fails (regardless of whether or not the CNOT gate fails); or if the input is 100 and at least one of the two gates fails. Cases I, II, and III below show the voter circuit failure probability analysis under these three conditions.

Case I: Missing or inactive negative-controlled CNOT gate

The MVCs shown in Figure 4.2 are 3-bit reversible circuits. Inputs to any of the circuits can be combined in $2^3 = 8$ possible ways. The 8 possible inputs can be shown as the elements of set I, where $I =$ (000, 001, 010, 011, 100, 101, 110, 111). If we select each of the inputs at random (by simple random sampling), then each input has the same probability (equal chance) of being an input to the voter circuit [15]. Under this assumption of equal probability, the probability of a particular input being selected is $\frac{1}{N}$, where $N =$ the number of all possible inputs. For example, out of the 8 possible inputs from I, the probability of occurrence of a particular input abc is $P(abc) = \frac{1}{8} = 0.125$.

We observe that for an input of $abc = 110$ to the MVCs shown in Figure 4.2, the MVCs will fail if exactly one CNOT gate is missing or inactive. Thus, for an input of $abc = 110$, the failure probabilities of the proposed MVCs are equal to the probability of occurrence of exactly one gate being missing or inactive. Resulting outputs for each different input to the circuits in Figure 4.2 are shown in Figure A.1.

To demonstrate, let us assume that:

• $P(CNOT_{exact})$ is the probability of occurrence of exactly one CNOT gate being missing or inactive,

• $P(CNOT)$ is the probability of the negative-controlled CNOT gate being missing or inactive, and

• $P(Fredkin)$ is the probability of the positive-controlled Fredkin gate being missing or inactive.

Thus, the probability of occurrence of exactly one CNOT gate being missing or inactive can be determined by:

$$P(CNOT_{exact}) = [P(CNOT) \times (1 - P(Fredkin))] \tag{4.5}$$

If the probability of the negative-controlled CNOT gate being missing or inactive is 0.001% and the probability of the positive-controlled Fredkin gate being missing or inactive is 0.002% then the voter failure probability can be calculated as:
$P(CNOT_{exact})$ or, $P(MVC_{failure}) = [0.001 \times (1-0.002)]\% = 0.000998\%$
However, under the assumption of equal probability for the inputs, the probability of MVC failure occurrence for an input $abc = 110$, can be determined by:
$P(abc) \times P(MVC_{failure}) = (\frac{1}{8} \times 0.000998)\% = 0.000124\%$.

Case II: Missing or inactive positive-controlled Fredkin gate

In case II, for an input of $abc = 011$ an inactive or missing Fredkin gate causes the MVCs to fail. Resulting outputs for each different input to the circuits in Figure 4.2 are shown in Figure A.2. The probability that the Fredkin gate fails (regardless of the other gate), will just be $P(Fredkin)$ since it doesn't matter what happens to the other gate. Thus, for input $abc = 011$ the probability of exactly one Fredkin gate being missing or inactive is equal to the failure probability of the proposed MVC and can be calculated by $P(abc) \times P(MVC_{failure})$.

Case III: At least one missing or inactive gate

In case III we see that for input $abc = 100$ the MVCs fail where at least one of the gates at Stage B in Figure 4.2 is missing or inactive. Figures A.1-A.3 in Appendix A show the outputs of the circuit where at least one of the gates being missing or inactive causes the circuit to fail. If the gate fault occurrences are not mutually exclusive then the circuit failure probability can be determined using the probability that at least one event occurs, as described in Section 4.2.4. Figure 4.3 shows a Venn diagram for determining voter circuit failure probability under the following assumptions:
- P(CNOT) is the probability that the CNOT gate is missing,
- P(Fredkin) is the probability that the Fredkin gate is missing,
- P(CNOT $\cup$ Fredkin) is the probability that at least one gate is missing, and

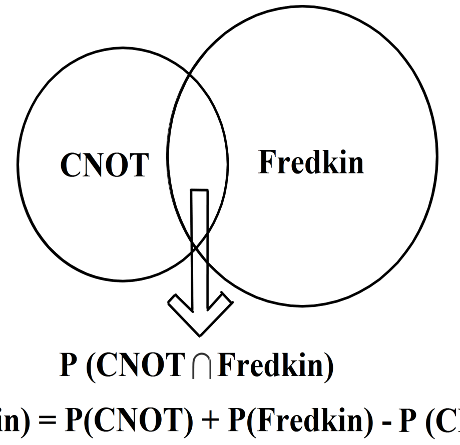

CNOT
Fredkin
P (CNOT ∩ Fredkin)
P (CNOT ∪ Fredkin) = P(CNOT) + P(Fredkin) - P (CNOT ∩ Fredkin)

Figure 4.3: Probability of the occurrence where at least one gate is missing or inactive.

• *P*(CNOT ∩ Fredkin) is the probability that both of the gates are missing.

Thus, the MVC failure probability can be determined by the following equation:

$$P(CNOT \cup Fredkin) = P(CNOT) + P(Fredkin) - P(CNOT \cap Fredkin) \qquad (4.6)$$

This can also be written as: *P*(CNOT ∪ Fredkin) = *P*(CNOT) + *P*(Fredkin) - *P*(CNOT ∩ Fredkin)

or,

$$P(CNOT \cup Fredkin) = P(CNOT) + P(Fredkin) - P(CNOT) \cdot P(Fredkin) \qquad (4.7)$$

Equation 4.7 can be used to determine the MVC failure probability where at least one inactive or missing gate fault occurs at Stage B in Figure 4.2 causing the MVCs to fail. However, for input $abc = 100$ the failure probability of the MVC is equal to the probability of at least one gate being missing or inactive, and therefore can be calculated by $P(abc)$ $\times$ $P(MVC_{failure})$. Readers are reminded that this analysis is just for an input of 100 and $P(abc = 100) = \frac{1}{8}$.

4.4 MVC Failure Probability for Gate Faults

This section introduces a technique for analysing inactive gate fault occurrences over varying numbers of trials. In the case of inactive gate faults, one or more gates become inactive and therefore, cause the entire circuit to fail. Studies on traditional circuit failure probabilities show that depending on different parameters, probabilities of gate fault occurrences may vary within the range of 0.001 to 0.1% [46, 42, 40, 1, 41, 11, 21, 60]. For analysis purposes, let us consider that we want to determine the circuit failure probabilities in the case where at least one gate becomes inactive. For an input of 100 our proposed MVCs fail when at least one of the reversible gates at Stage B in Figure 4.2 becomes inactive. This particular instance can be analysed using our proposed technique in Case III. Due to the unavailability of reversible gate failure data, we have assumed that the probabilities of gate fault occurrences vary from 0.001 to 0.005%. We consider that for each operation the reversible gates at Stage B in Figure 4.2 have the same gate fault probabilities. Simulated results are analysed to determine circuit failure probabilities over varying numbers of trials using binomial distribution.

In the first step of our analysis, we have determined the voter circuit failure probability for a single trial. We then expand the voter circuit failure probabilities over 100, 1000 and $10,000$ trials. The results of our analysis are presented in Table 4.1.

In Table 4.1,

• x is the probability of the gate being inactive in one trial, where $x \in [0,1]$

• P_{voter1} is the voter circuit failure probability in one trial, where $P_{voter1} \in [0,1]$

• P_{voter2} is the voter circuit failure probability in N trials, where $P_{voter2} \in [0,1]$

As shown in Table 4.1, if the probability of the CNOT or the Fredkin gate becoming inactive is 0.001% then the probability that the voter circuit will fail at least once in one trial is $1.99999 \times 10^{-3}\%$. The voter circuit failure probability is calculated using the equation (4.7). We expand the voter circuit failure probability over varying numbers of trials using a binomial distribution. The binomial distribution is calculated and plotted against the

Table 4.1: Failure probabilities of the proposed MVCs for an input of 100.

x	P_{voter1}	N	P_{voter2}
0.00001	1.99999E-05	100	0.002
		1000	0.018
		10000	0.18
0.00002	3.99996E-05	100	0.004
		1000	0.038
		10000	0.34
0.00003	5.99991E-05	100	0.006
		1000	0.058
		10000	0.45
0.00004	7.99984E-05	100	0.008
		1000	0.078
		10000	0.55
0.00005	9.99975E-05	100	0.010
		1000	0.09
		10000	0.62

number of trials using the ***numpy.random.binomial*** function in Python. The computation was performed using a computer with a 2.6 GHz processor, 4 GB of RAM and the 64 bit Windows operating system. The pseudo code for running the MVC simulations is shown in Figure 4.4.

Studies in [40, 42, 46, 41, 11, 21, 60] show that the best way to represent probability distribution data of circuit failure distribution is probability plotting. The probability plotting technique enables us to determine whether the proposed mathematical model is suitable for detailed analysis and can provide directions for future studies. Figures 4.5-4.9 show the proposed voter circuit failure probability plots over varying numbers of trials. These plots demonstrate that the failure probabilities increase with the numbers of trials and with the increasing gate fault probabilities. Our analysis demonstrates how the probabilities of gate fault occurrences can affect the voter circuit failure probabilities. The probability distribution data shows that if at least one gate at Stage B of Figure 4.2 fails then the voter circuit failure probability exceeds 50% with increasing gate fault probabilities and circuit trials. For example, if the probability of a gate of becoming inactive is 0.003% then the probabil-

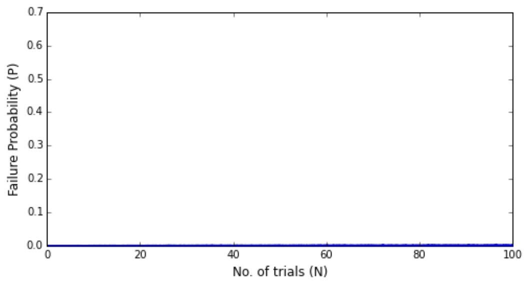
```
State Input:  gate fault probabilities
State Output: circuit failure probability

  State function {Main}
   a = gate failure probabilities
       for {each values in a}
          COMPUTE the circuit failure probabilities
   P_{voter1} = circuit failure probabilities
       return P_{voter1}
       end {Main}

  State function {binomial distribution}
   IDENTIFY P_{voter1}
   SPECIFY the no. of circuit operations (N)
       for {k = 1, k++, while k < N}
         COMPUTE P_{voter2} (k,N)
         LIST values
       end {binomial distribution}

plot (P_{voter2}, N)
       Return
```

Figure 4.4: Pseudo code for MVC failure analysis for an input of 100.

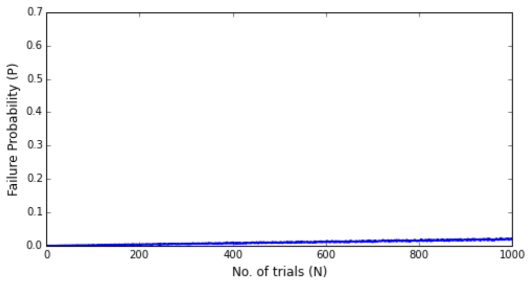
ity of the voter circuit failure is $5.99999 \times 10^{-3}\%$ in one trial. With an increasing number of trials, e.g. for $N = 100$, 1000 and 10000 the voter circuit failure probability increases to 0.6%, 5.8% and 45%, respectively. If the gate fault probability increases to 0.004%, then over 10000 trials the voter failure probability increases to 55%. In other words, the voter circuit failure probability will exceed 50% over the numbers of trials assuming equal gate fault probabilities of 0.004%. Thus, the simulation studies proposed in this thesis can be used to analyse both the gate fault probabilities and the resulting voter circuit failure probabilities over varying numbers of trials.

4.5 Implications of the Analysis

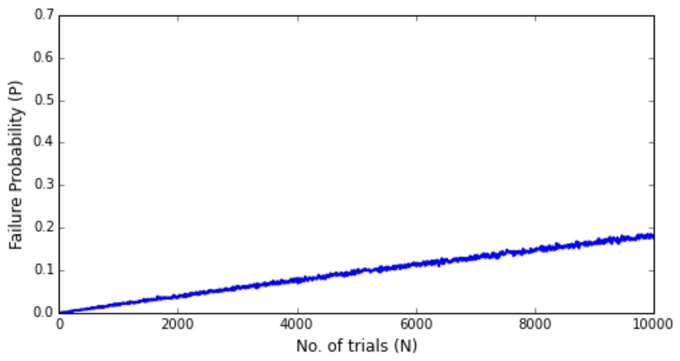
A review of the literature on reliability and failure probability analysis indicates that the traditional analysis approaches can be computationally expensive due to the increasing size of digital logic circuits [10]. Circuit failure probability analysis or determining the logic gate failure rate are common forms of reliability analysis [10, 34]. In reversible logic, technologies are still being developed and are still mostly theoretical in nature. The data on failure rate information for reversible logic gates is unavailable because the technologies to implement reversible gates are still being studied and developed. Also, the technologies have not been used in the quantities and over the time required to generate accurate statistical data on their failure rates. This motivated us to conduct our research on finding a potential approach for failure probability analysis of the proposed reversible majority voter circuits. Instead of determining the failure probabilities as a function of time, our proposed approach considers failure probabilities of the circuit components for determining the overall circuit failure probability. Analysis and the probability plots shown in this thesis can be used to assess the theoretical distribution of real sets of data. Future research on designing reliable reversible logic circuits may depend highly on predicting the failure probabilities of the circuit components. Predictions on circuit failure probability can also be used to evaluate design feasibility of any reversible logic circuit. However, in this thesis we provide a methodology and structure, set of equations and process that can be used when researchers have more information about the probabilities of failures under different circumstances for reversible logic gates and circuits.



(a) MVC failure probabilities over 100 trials

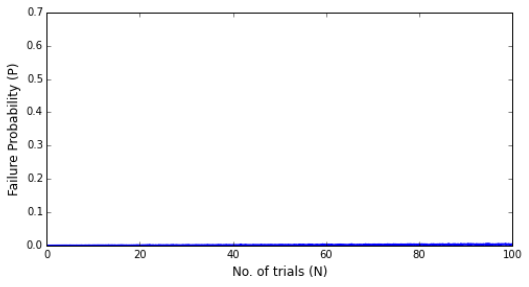


(b) MVC failure probabilities over 1000 trials

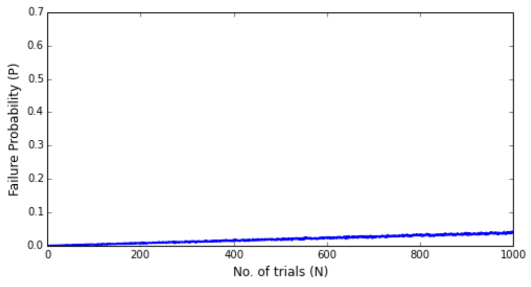


(c) MVC failure probabilities over 10000 trials

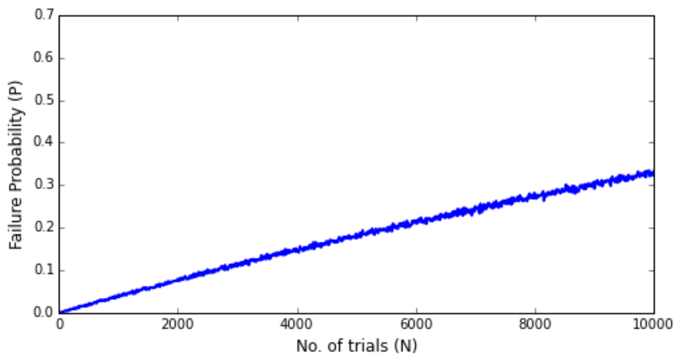
Figure 4.5: MVC failure probabilities for gate fault probabilities, $x = 0.001\%$.



(a) MVC failure probabilities over 100 trials

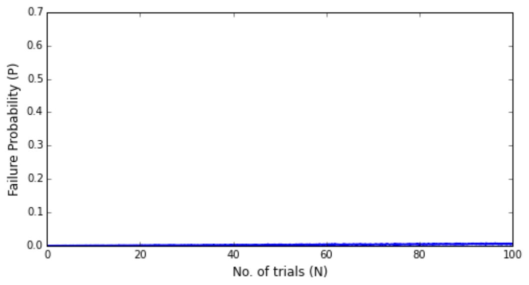


(b) MVC failure probabilities over 1000 trials

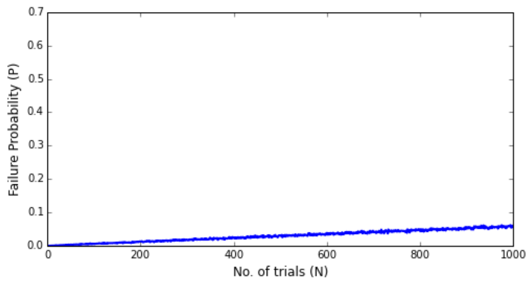


(c) MVC failure probabilities over 10000 trials

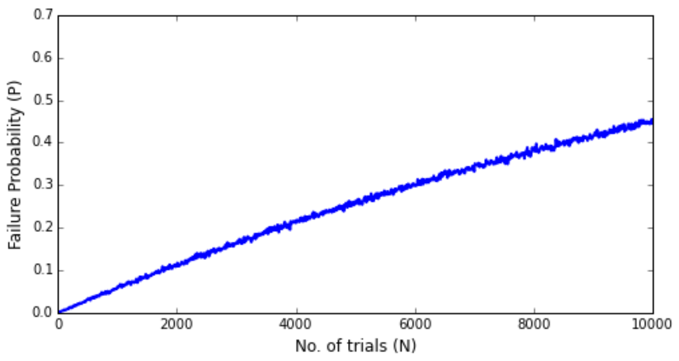
Figure 4.6: MVC failure probabilities for gate fault probabilities, $x = 0.002\%$.



(a) MVC failure probabilities over 100 trials

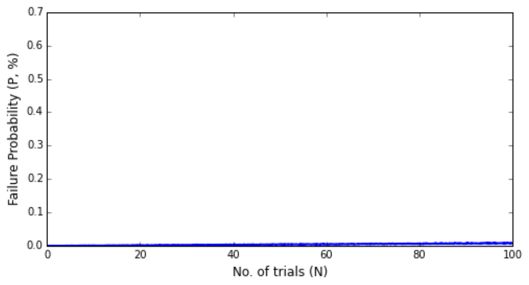


(b) MVC failure probabilities over 1000 trials

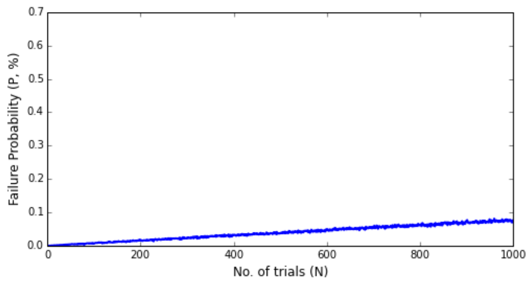


(c) MVC failure probabilities over 10000 trials

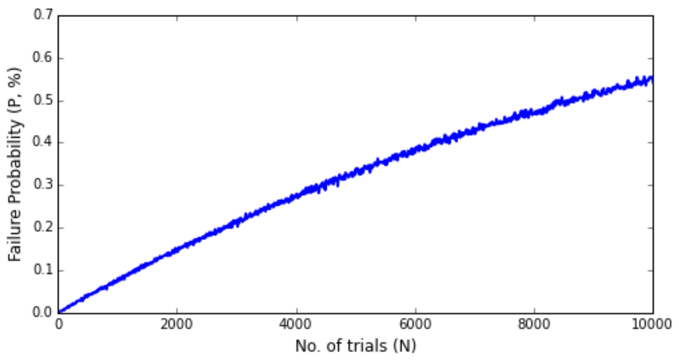
Figure 4.7: MVC failure probabilities for gate fault probabilities, $x = 0.003\%$.



(a) MVC failure probabilities over 100 trials

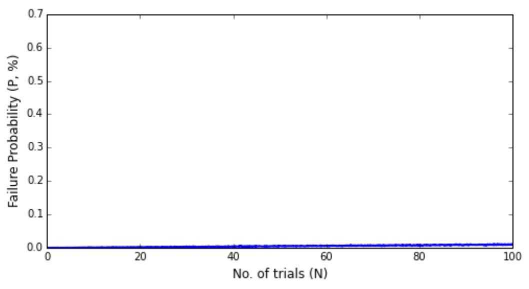


(b) MVC failure probabilities over 1000 trials

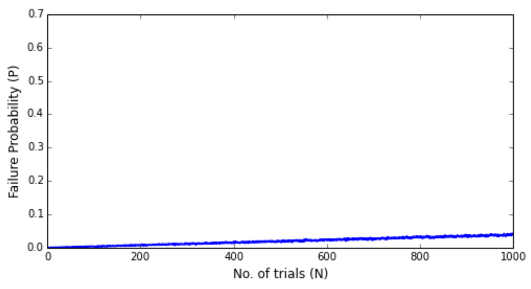


(c) MVC failure probabilities over 10000 trials

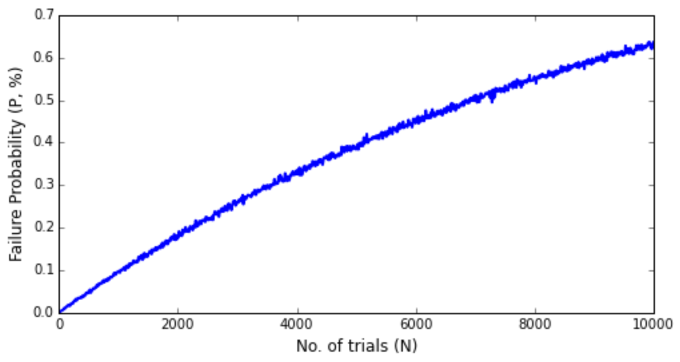
Figure 4.8: MVC failure probabilities for gate fault probabilities, $x = 0.004\%$.



(a) MVC failure probabilities over 100 trials



(b) MVC failure probabilities over 1000 trials



(c) MVC failure probabilities over 10000 trials

Figure 4.9: MVC failure probabilities for gate fault probabilities, $x = 0.005\%$.

Chapter 5

Conclusions

5.1 Contributions

The contributions of this thesis span the field of fault tolerance in reversible circuit designs. The majority of the existing work in reversible fault tolerance only focuses on fault testing. In this thesis we have described the requirements and techniques to achieve fault tolerance in reversible logic. We offer a passive hardware redundancy technique to achieve fault masking by majority voting. Our first contribution, described in Chapter 3, is the design of two reversible majority voter circuits for fault masking purposes. The majority voter circuit generates the majority of the input bits on a specified output line. We have demonstrated the use of this circuit in a full adder circuit design. The proposed voter circuits can be used to make any reversible circuit fault tolerant, which is also demonstrated.

We have shown that the proposed reversible majority voter circuits can correct any single fault (i.e. a single bit fault, a single gate fault, a crosspoint fault and the family of missing gate faults) occuring in any of the triplicated modules. We have also compared our proposed voter circuit design with the designs available in the literature [7, 43, 61]. Our analysis shows that the proposed designs are simpler and have lower costs in terms of the gate count, quantum cost, and the number of constant inputs and garbage outputs.

This thesis also evaluates the performance of the proposed designs for a majority voter circuit. We have presented a comparison of the proposed and available reversible voter circuit designs. We have also analyzed our voter circuit's fault masking capability assuming faults at different locations inside the voter. We have demonstrated that one of our proposed voter circuits has fault masking capability unlike any of the existing designs in the literature.

Another major contribution of this thesis, described in Chapter 4, determines failure probabilities of our proposed voter circuits over varying numbers of trials under specific conditions. The voter circuit failure probability is determined based on the probabilities of missing or inactive gate faults. We have developed a set of equations that can be used to calculate the circuit failure probability based on the probabilities of gate fault occurrences. We have demonstrated that if the gate fault probabilities increase to 0.003% or more then for a certain number of trials the voter circuit failure probability exceeds 50%. The simulation results can be useful to reversible logic designers during the fabrication process of future reversible logic gates. We expect that this thesis will have a significant impact on designing fault tolerant reversible circuits as well as determining failure probabilities of the designs.

5.2 Future Work

This thesis opens up a number of possible research areas for fault tolerance in reversible logic. Further research may pursue several avenues related to this work, such as:

1. In this thesis, we propose a passive hardware redundancy technique for fault masking purposes. The proposed voter circuits can mask a single fault using triple modular redundancy (TMR) technique, however they cannot detect or locate the fault. It may be feasible and useful to detect a fault within the voter or any of the TMR modules. A commonly used technique for fault detection is to add a parity line within the reversible circuits. A parity line for the voters or any of the triplicated modules could be used to detect faults within the circuits.

2. Our proposed majority voter circuits cannot correct all possible single faults (e.g. a single gate fault or, a single bit fault) that may occur inside a voter circuit. Although one of our proposed voter circuits is able to mask a single fault in specific locations inside the voter circuit lines, the voter circuit fails to perform the intended operation in case the fault occurs anywhere else inside the circuit. Extension of our proposed voter

circuits should focus on masking any single fault occuring inside the voter circuits and ensure robustness of the designs.

3. The concept of triple modular redundancy (TMR) can be generalized as the *N*-modular redundancy (NMR). NMR technique applies the same approach as TMR but uses *N*-versions of a given module. Our proposed designs can also be used for an extension to a *n*-bit majority voter circuit proposed in [43].

4. In order to increase reliability, our proposed TMR technique can be duplicated in the logic design. In this case, we further triplicate the TMR circuit and obtain a circuit with nine copies of the basic module. A duplicated TMR technique will use two layers of the majority voter circuits. This process can be repeated to obtain a cascaded triple modular redundancy (CTMR) or recursive triple modular redundancy (RTMR). For traditional logic, it is shown that recursive voting offers a double exponential decrease in a circuit's failure probability [54]. It would be interesting to analyse our proposed designs with recursive voting and determine the circuit failure probabilities over different numbers of trials.

5. One of the main design issues of a TMR based system is to design a reliable majority voter circuit. The system fails to provide a correct output if the voter circuit itself is faulty. Therefore, reliability of the fault tolerant design depends largely on the robustness of the voter circuit. In this thesis, we analyse the failure probabilities of the proposed voters based on the gate fault occurrences. However, further scope exists for exploring the reliability of future reversible gates and circuits. Observability based reliability analysis can be used to determine the reliability of the reversible gates and fault tolerant circuit designs proposed in this thesis.

6. The voter circuit failure probability analysis proposed in this thesis only applies when the gates used in the circuit are missing or become inactive. The extension of this thesis work should explore possibilities for circuit failure analysis for other possible gate faults, or bit faults that may cause the voter circuit to fail.

7. In the scope of this thesis, we have only considered the area of Boolean reversible logic which is a binary or two-valued logic. Our proposed majority voter circuits will mask the faults within the logic circuits designed in Boolean domain. However, our work can be extended for designing multiple-valued logic circuits within the scope of TMR. A multiple-valued logic (MVL) is a d-valued logic with $d > 2$. If $d = 3$, it is known as ternary logic. For a detailed description about MVL readers are referred to [38]. As a completely new research area, multiple-valued logic and ternary reversible gates are yet to be explored. Designing a majority voter circuit in MVL will involve ternary reversible gates [27] such as ternary Toffoli gates and Muthukrishnan-Stroud (M-S) gates. However, further investigation is required to design a feasible majority voter circuit in MVL logic design.

Bibliography

[1] A. Agarwal, B. C. Paul, S. Mukhopadhyay, and K. Roy. Process Variation in Embedded Memories: Failure Analysis and Variation Aware Architecture. *IEEE Journal of Solid-State Circuits*, 40(9):1804–1814, 2005.

[2] A. N. Al-Rabadi. *Reversible Logic Synthesis: from Fundamentals to Quantum Computing*. Springer Science & Business Media, 2012.

[3] G. Allwein and J. Barwise. *Logical Reasoning with Diagrams*, volume 6. Oxford University Press, 1996.

[4] M. Arabzadeh, M. Saeedi, and M. S. Zamani. Rule-Based Optimization of Reversible Circuits. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, pages 849–854. IEEE Press, 2010.

[5] W. C. Athas and L. J. Svensson. Reversible Logic Issues in Adiabatic CMOS. In *Proceedings of the Workshop on Physics and Computation*, pages 111–118. IEEE, 1994.

[6] C. H. Bennett. Logical Reversibility of Computation. *IBM Journal of Research and Development*, 17(6):525–532, 1973.

[7] P. O. Boykin and V. P. Roychowdhury. Reversible Fault-Tolerant logic. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 444–453. IEEE, 2005.

[8] T. Chattopadhyay. All-Optical Symmetric Ternary Logic Gate. *Optics & Laser Technology*, 42(6):1014–1021, 2010.

[9] C. Chen and Y. Mao. A Statistical Reliability Model for Single-Electron Threshold Logic. *IEEE Transactions on Electron Devices*, 55(6):1547–1553, 2008.

[10] C. Chen and R. Xiao. A Fast Model for Analysis and Improvement of Gate-Level Circuit Reliability. *INTEGRATION, the VLSI Journal*, 50:107–115, 2015.

[11] Z. Chen and S. Zheng. Lifetime Distribution Based Degradation Analysis. *IEEE Transactions on Reliability*, 54(1):3–10, 2005.

[12] M. R. Choudhury and K. Mohanram. Accurate and Scalable Reliability Analysis of Logic Circuits. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 1454–1459. EDA Consortium, 2007.

[13] M. R. Choudhury and K. Mohanram. Reliability Analysis of Logic Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(3):392–405, 2009.

[14] J. L. Devore. Probability and Statistics for Engineering and the Sciences. 2008.

[15] V. J. Easton and J. H. McColl. Statistics Glossary. 1997.

[16] K. Fazel, M. A. Thornton, and J. E. Rice. ESOP-based Toffoli Gate Cascade Generation. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 206–209. IEEE, 2007.

[17] M. P. Frank. Approaching the Physical Limits of Computing. In *Proceedings of the 35th International Symposium on Multiple-Valued Logic*, pages 168–185. IEEE, 2005.

[18] E. Fredkin and T. Toffoli. *Conservative Logic*. Springer, 2002.

[19] M. Haghparast and K. Navi. Design of a Novel Fault Tolerant Reversible Full Adder for Nanotechnology Based Systems. *World Applied Sciences Journal*, 3(1):114–118, 2008.

[20] J. S. Hall. Nanocomputers and Reversible Logic. *Nanotechnology*, 5(3):157, 1994.

[21] P. L. Hall and J. E. Strutt. Probabilistic Physics-of-Failure Models for Component Reliabilities Using Monte Carlo Simulation and Weibull Analysis: A Parametric Study. *Reliability Engineering & System Safety*, 80(3):233–242, 2003.

[22] T. Hey. Quantum Computing: An Introduction. *Computing & Control Engineering Journal*, 10(3):105–112, 1999.

[23] M. S. Islam, M. M. Rahman, Z. Begum, and M. Z. Hafiz. Fault Tolerant Reversible Logic Synthesis: Carry Look-Ahead and Carry-Skip Adders. In *Proceedings of the International Conference on Advances in Computational Tools for Engineering Applications*, pages 296–401, 2009.

[24] M. S. Islam, M. M. Rahman, Z. Begum, and M. Z. Hafiz. Realization of a Novel Fault Tolerant Reversible Full Adder Circuit in Nanotechnology. *The International Arab Journal of Information Technology*, 7(3):317–323, 2010.

[25] B. W. Johnson. *Design & Analysis of Fault Tolerant Digital Systems*. Addison-Wesley Longman Publishing Incorporated Companies, 1988.

[26] P. Kaur and B. S. Dhaliwal. Design of Fault Tolerant Full Adder/Subtarctor Using Reversible Gates. In *International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5. IEEE, 2012.

[27] M. H. A. Khan and M. A. Perkowski. Quantum Ternary Parallel Adder/Subtractor with Partially-Look-Ahead Carry. *Journal of Systems Architecture*, 53(7):453–464, 2007.

[28] S. G. Krantz. *Handbook of Logic and Proof Techniques for Computer Science*. Springer Science & Business Media, 2002.

[29] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes. Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*, pages 282–287. IEEE Computer Society, 2005.

[30] R. Landauer. Irreversibility and Heat Generation in the Computing Process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[31] J. C. Laprie. Dependability: Basic Concepts and Terminology. In *Dependability: Basic Concepts and Terminology*, pages 3–245. Springer, 1992.

[32] C. A. L. Lisbôa, E. Schüler, and L. Carro. Going Beyond TMR for Protection Against Multiple Faults. In *Proceedings of the 18th Annual Symposium on Integrated Circuits and System Design*, pages 80–85. ACM, 2005.

[33] C. J. Lu and W. O. Meeker. Using Degradation Measures to Estimate a Time-to-Failure Distribution. *Technometrics*, 35(2):161–174, 1993.

[34] D. Manimekalai and P. Dixit. Analysis of Reliability for Fault Tolerant Design in NANO CMOS Logic Circuit. *International Journal of Nanoelectronics & Materials*, 10(2), 2017.

[35] D. Maslov, G. Dueck, and N. Scott. Reversible Logic Synthesis Benchmarks Page. *Online: http://www. cs. uvic. ca/˜ dmaslov*, 2005.

[36] D. Maslov, G. W. Dueck, and D. M. Miller. Simplification of Toffoli Networks via Templates. In *Proceedings of the 16th Symposium on Integrated Circuits and Systems Design*, pages 53–58. IEEE, 2003.

[37] D. A. Maslov. *Reversible Logic Synthesis*. PhD thesis, Citeseer, 2003.

[38] D. M. Miller and M. A. Thornton. Multiple Valued Logic: Concepts and Representations. *Synthesis Lectures on Digital Circuits and Systems*, 2(1):1–127, 2007.

[39] S. K. Mitra and A. R. Chowdhury. Minimum Cost Fault Tolerant Adder Circuits in Reversible Logic Synthesis. In *25th International Conference on VLSI Design (VLSID)*, pages 334–339. IEEE, 2012.

[40] S. Mukhopadhyay, H. Mahmoodi, and K. Roy. Statistical Design and Optimization of SRAM Cell for Yield Enhancement. In *Proceedings of the 2004 IEEE/ACM International Conference on Computer-Aided Design*, pages 10–13. IEEE Computer Society, 2004.

[41] S. Mukhopadhyay, H. Mahmoodi, and K. Roy. Modeling of Failure Probability and Statistical Design of SRAM Array for Yield Enhancement in Nanoscaled CMOS. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(12):1859–1880, 2005.

[42] S. Mukhopadhyay, H. Mahmoodi-Meimand, and K. Roy. Modeling and Estimation of Failure Probability due to Parameter Variations in Nano-scale SRAMs for Yield Enhancement. In *Symposium on Digest of Technical Papers in VLSI Circuits*, pages 64–67. IEEE, 2004.

[43] M. A. Nashiry and J. E. Rice. A Reversible Majority Voter Circuit and Applications. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 1–6. IEEE, 2017.

[44] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[45] B. Parhami. Fault-Tolerant Reversible Circuits. In *Proceedings of the Fortieth Asilomar Conference on Signals, Systems and Computers (ACSSC)*, pages 1726–1729, Oct. 29–Nov. 1 2006.

[46] F. W. Poblenz. Analysis of Transistor Failure in a Nuclear Environment. *IEEE Transactions on Nuclear Science*, 10(1):74–79, 1963.

[47] J. W. Pratt, H. Raiffa, and R. Schlaifer. *Introduction to Statistical Decision Theory*. MIT Press, 1995.

[48] T. Rejimon and S. Bhanja. Scalable Probabilistic Computing Models using Bayesian Networks. In *48th Midwest Symposium on Circuits and Systems*, pages 712–715. IEEE, 2005.

[49] J. E. Rice. An Overview of Fault Models and Testing Approaches for Reversible Logic. In *Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pages 125–130. IEEE, 2013.

[50] A. Sarker, T. Ahmed, S. M. M. Rashid, S. Anwar, L. Jaman, N. Tara, M. M. Alam, and H. M. H. Babu. Realization of Reversible Logic in DNA Computing. In *11th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 261–265. IEEE, 2011.

[51] R. R. Schaller. Moore's Law: Past, Present and Future. *IEEE Spectrum*, 34(6):52–59, 1997.

[52] B. Sen, S. Ganeriwal, and B. K. Sikdar. Reversible Logic-Based Fault-Tolerant Nanocircuits in QCA. *ISRN Electronics*, 2013, 2013.

[53] T. R. Stankovic, M. K. Stojcev, and G. L. Djordjevic. Design of Self-Checking Combinational Circuits. In *6th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service*, volume 2, pages 763–768. IEEE, 2003.

[54] D. D. Thaker, R. Amirtharajah, F. Impens, I. L. Chuang, and F. T. Chong. Recursive TMR: Scaling Fault Tolerance in the Nanoscale Era. *IEEE Design & Test of Computers*, 22(4):298–305, 2005.

[55] H. Thapliyal and N. Ranganathan. Testable Reversible Latches for Molecular QCA. In *8th IEEE Conference on Nanotechnology*, pages 699–702. IEEE, 2008.

[56] H. Thapliyal and M. Zwolinski. Reversible Logic to Cryptographic Hardware: A New Paradigm. In *49th IEEE International Midwest Symposium on Circuits and Systems*, volume 1, pages 342–346. IEEE, 2006.

[57] T. Toffoli. *Reversible Computing*. Springer, 1980.

[58] F. K. Wang. A New Model with Bathtub-Shaped Failure Rate Using an Additive Burr XII Distribution. *Reliability Engineering & System Safety*, 70(3):305–312, 2000.

[59] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler. Revlib: An Online Resource for Reversible Functions and Reversible Circuits. In *38th International Symposium on Multiple Valued Logic*, pages 220–225. IEEE, 2008.

[60] M. Xie and C. D. Lai. Reliability Analysis Using an Additive Weibull Model with Bathtub-Shaped Failure Rate Function. *Reliability Engineering & System Safety*, 52(1):87–93, 1996.

[61] M. Zamani, N. Farazmand, and M. B. Tahoori. Fault Masking and Diagnosis in Reversible Circuits. In *16th IEEE European Test Symposium (ETS)*, pages 69–74. IEEE, 2011.

Appendix A

Voter Circuit Behavior

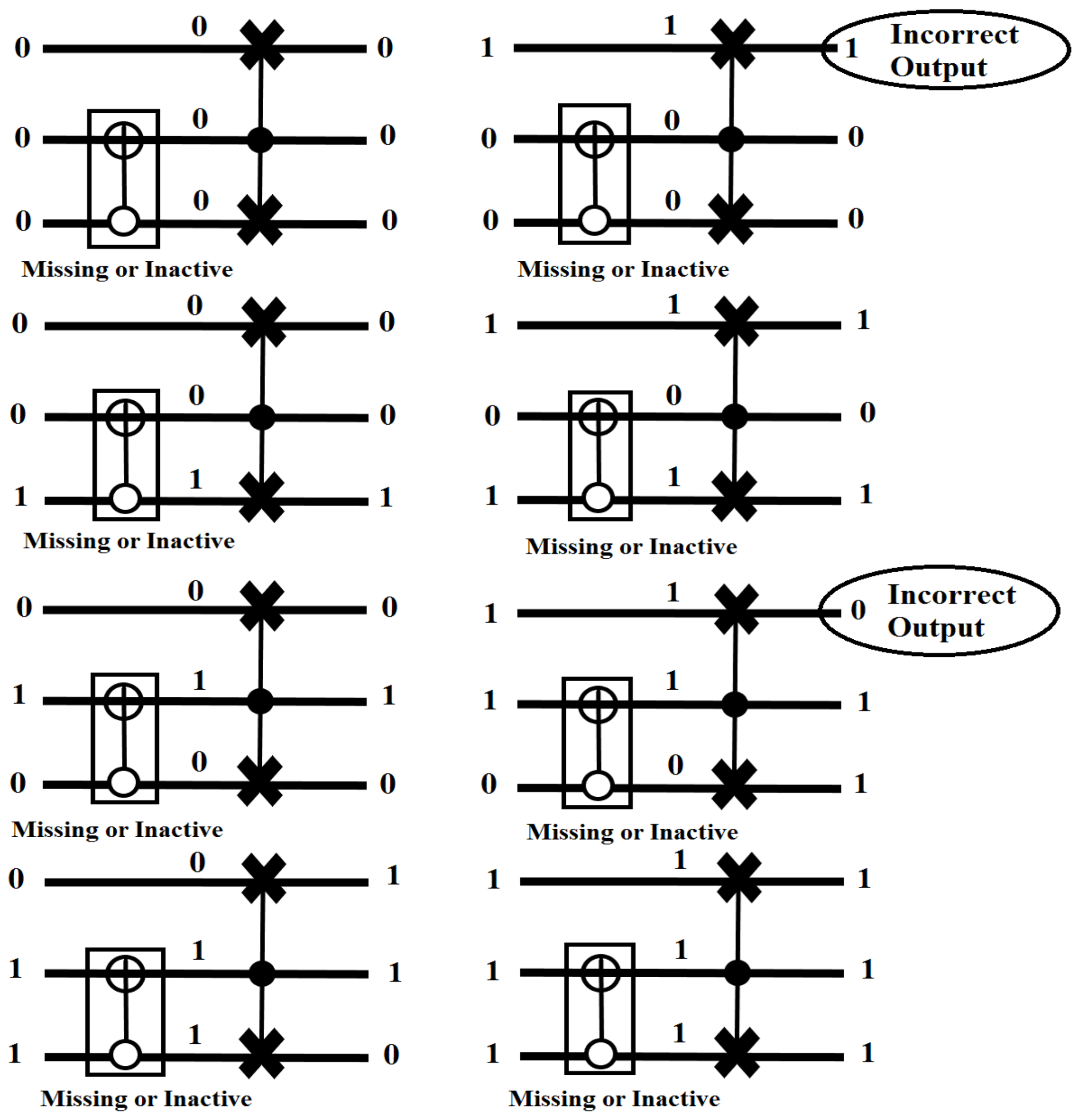


Figure A.1: Circuit outputs for missing or inactive negative-controlled CNOT gate.

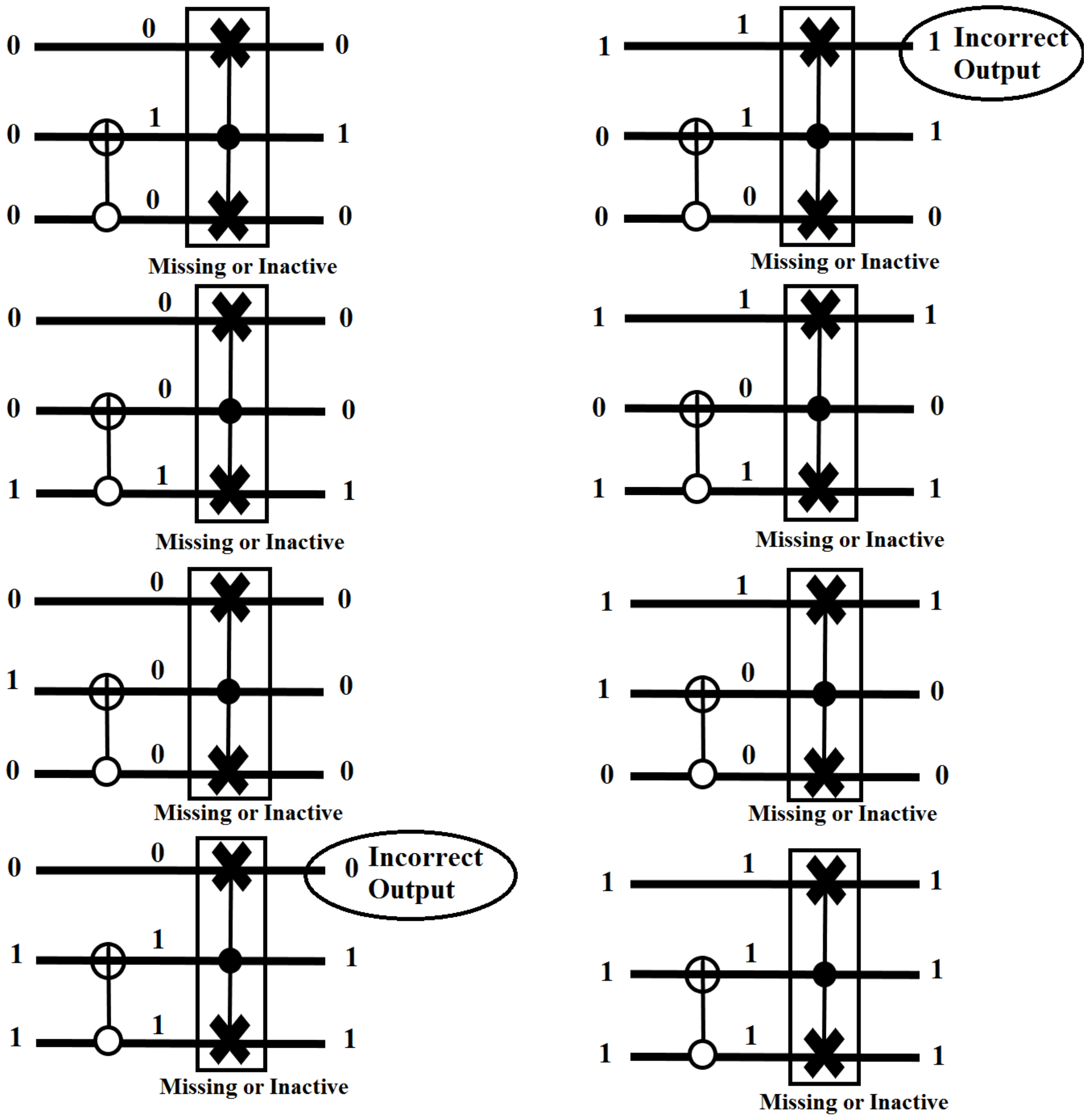


Figure A.2: Circuit outputs for missing or inactive positive-controlled Fredkin gate.

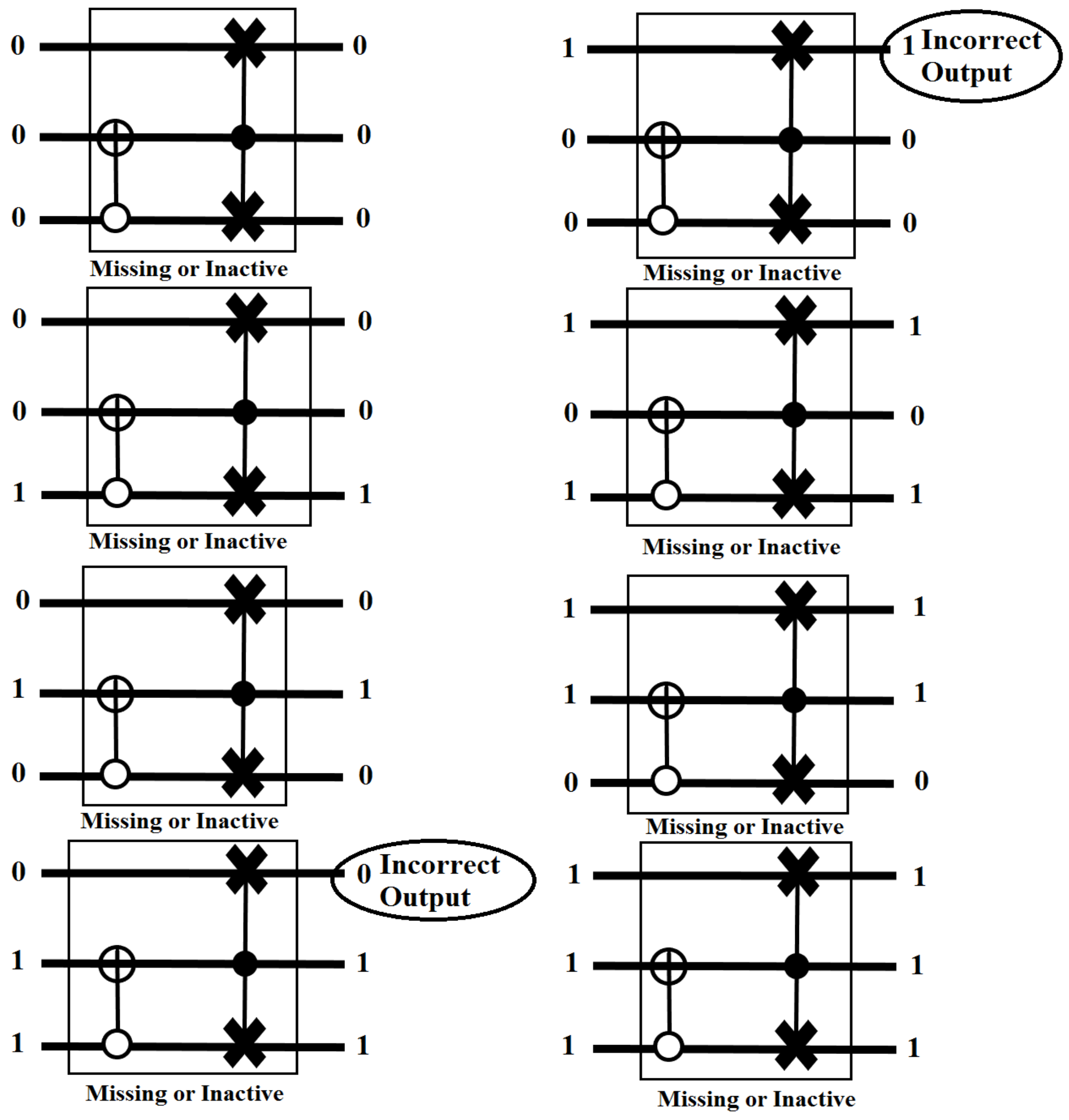


Figure A.3: Circuit outputs for missing or inactive gates.