Design of a Compact Reversible Fault Tolerant Field Programmable Gate Array : A Novel Approach in Reversible Logic Synthesis

by

Md. Shamsujjoha

Exam Roll : Curzon – 417

Registration No: HA – 3765

Session : 2005 – 06

A Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY OF DHAKA

June 2012

Abstract

This thesis demonstrates the reversible logic synthesis for Plessey logic block and basic input-output block for the **F**ield **P**rogrammable **G**ate **A**rray (FPGA). The circuits are designed using only reversible fault tolerant Fredkin and Feynman double gates. Thus, the entire scheme inherently becomes fault tolerant. Algorithms to design compact reversible fault tolerant n-to-2^n decoder, $4n$-to-n multiplexer, random access memory and Plessey logic block for FPGA have been presented. In addition, several lower bounds on the number of garbage outputs, constant inputs and quantum cost of the FPGA have been proposed. Transistor simulations of the proposed FPGA have been shown to check the correctness of the function of the circuit using standard p-MOS 901 and n-MOS 902 model with delay of $0.030ns$ and $0.12\mu m$ channel length. The comparative results show that the proposed design is much better in terms of gate count, garbage outputs, constant inputs, quantum cost, delay, hardware complexity and has significantly better scalability in comparison to previous approaches. Moreover, the proposed fault tolerant structure of reversible FPGA is a novel idea in literature to the best of our knowledge.

Declaration

I, hereby, declare that the work presented in this thesis is the outcome of the investigation performed by me under the supervision of Dr. Hafiz Md. Hasan Babu, Professor, Department of Computer Science and Engineering, University of Dhaka. I also declare that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

Signature

____

(Candidate)

Approval Signature

____

(Supervisor)

Publications

During the course of this thesis, a number of refereed journal and conference articles have been published or are in review processing based on the work presented in this thesis. They are listed below for the reference.

Journal Articles

1. "Design of Optimal Reversible Carry Look-Ahead Adder with Optimal Garbage and Quantum Cost." *The International Journal of Engineering & Technology.* vol:2, no:1, pp 44-50, Jan-2012.
2. "A Novel Approach to Design Reversible Fault Tolerant Barrel Shifters." *Dhaka University Journal of Applied Science & Engineering.* Accepted in April-28, 2012.

Conference Articles

1. "Design of A Compact Reversible Random Access Memory." *4th IEEE International Conference on Computer Science and Information Technology, Chengdu, China.* vol:10, pp 103-107, Jun-2011.
2. "An Efficient Reversible Fault Tolerant Plessey Logic Block of Field Programmable Gate Arrays." *4th Workshop on Reversible Computation, Copenhagen, Denmark.* Accepted in May-7, 2012.

Articles In Review

1. "Design of a Compact Reversible Fault Tolerant Field Programmable Gate Array : A Novel Approach in Reversible Logic Synthesis." *Microelectronics Journal, Elsevier.* Submitted in Apr-29, 2012.
2. "An Efficient Approach for Realization of a Compact Reversible Fault Tolerant Decoder Using MOS Transistors." *International Conference on Computer-Aided Design, California, USA, Nov-2012.* Submitted in Apr-15, 2012.

Acknowledgements

First, I am thankful and expressing my gratefulness to Almighty who offers me divine blessings, patience, mental and psychical strength to complete this thesis. The progression of this thesis could not possibly be carried out without the help of several people who, directly or indirectly, are responsible for the completion of this work. I deeply indebted to my thesis supervisor Prof. Dr. Hafiz Md. Hasan Babu. His scholarly guidance, especially for his tolerance with my persistent bothers and unfailing support. He give me the freedom to pursue aspects of reversible fault tolerant computing which I found interesting and compelling. This helped my thesis to achieve its desired goals. Secondly to Ms. Lafifa Jamal, whom I have collaborated closely over the last two years and who have not only brought out the best in my research, but also have always been up for a few drinks wherever we have been in the world.

I wish to thank the great people of CSEDU. A special thank goes to Dr. Md. Mustafizur Rhaman and Dr. Mamun Or Rashid for their well-disposed instructions and Encouragements. Laboratories for Computer Science and Engineering has provided an ideal working environment. I offer thanks to the Lab assistants who have been particularly supportive with my incessant requests. I am also indebted to the library of our department, from where this research began. I also like to thank all the researchers within these field whose profound papers, reports and journals have helped to cultivate my interest in the subject of reversible, fault tolerant and quantum computing.

Finally, I would like to thank my non-CSE friends and family. Their continued tolerance with my moods and tendency to disappear for weeks at a time gave me a much needed break from the world computing.

Dedicated To Those People

Who are Honest, Spiritual and Patriot

Who Work for the Welfare of Human Being

Contents

List of Figures

List of Tables

I don't know anyone who has gotten to the top without hard work. That is the recipe. It will not always get you to the top, but it will get you pretty near.

–Margaret Thatcher

(Britain's first female prime minister)

Chapter 1

Introduction

The connection between a single bit and corresponding minimum quantity of entropy was ascertained by Maxwell and Szilard in [1, 2], which state, "a minimum amount of energy ($KT\ ln(m)$, where K is Boltzmann's constant of $1.38{\times}10^{23}$ JK^{-1} and T is the absolute temperature of the environment and m is an integer proportional to the number of computed bit) must be dissipated during every elemental enactment of computation". Later Landauer proved that energy dissipation is only avoidable if the system is made reversible [3]. In [4], Bennett showed that a reversible computation, in which no information is destroyed, dissipate an arbitrarily small amount of energy which is equals to $KT\ ln(1)$, *i.e.*, logically zero energy dissipation. It has also been pointed out that, an irreversible system has a fundamental lower limit to the energy dissipation during a computation which is equals to $KT\ ln(2)$ for each erased bit [3, 4]. An irreversible system can also store the information that is produced during a computation rather than erasing it, but doesn't always provide a unique path from each state to its previous state. Energy used to store these information is unrecoverable unless the computation is performed reversibly. If we are to continue the revolution in performance of computer hardware, we must reduce these energy dissipations.

Although power dissipation due to information loss is negligible at present, but it will become a substantial factor by 2020 as a result of increased density in the computer hardware if Moore's Law continues to be in effect [5]. Gordon Moore predicted that [6], *"The number of transistors and resistors on a chip doubles every 18 months."* Which signifies, *"it would continue to improve at an exponential rate with the performance per unit cost increasing by a factor of 2 every 18 months or so"*. This assumption has a great importance lies in the future technological necessity that the power dissipation should be minimized, otherwise internal overheating of a chip will demolish it. In this regard, reversible logic will play an extensively crucial role in the upcoming days. Moreover, reversible circuit can be viewed as a special case of quantum circuit as quantum evolution must be reversible [7, 8]. On the other hand, reversible fault tolerant circuit allows to detect faulty signal in the primary outputs of the circuit through parity checking [9, 10]. Parity checking is the one of the oldest and widely used mechanisms for detecting single level fault in communication. It's most common use is for detecting errors in the storage or transmission of information, primarily because most arithmetic and other processing functions tend not to preserve the parity of the data [11, 12]. If the parity of the input data is maintained throughout the computation then intermediate checking wouldn't be required and an entire circuit can preserve parity if its individual gate is parity preserving [13]. In other words, reversible fault tolerant circuit can capture any erroneous result that tends to propagate through the downstream of modules without a danger of corrupting additional information. Over the past few years, both reversible and fault tolerant circuitry gained remarkable interest in the field of DNA-technology [14], nano-technology [15, 16], optical computing [17], program debugging and testing [18], database recovery, quantum dot cellular automata [19, 20], discrete event simulation [21], modeling of biochemical systems [22], in the development of highly efficient algorithms [23, 24, 25], etc. Thus, the reversible, fault tolerant and quantum circuits have been implemented and are seen as promising alternatives to conventional technologies.

New products in shortest possible time have become the catchword in today's electronic industry. By being able to test the product even before the fabrication **F**ield **P**rogrammable **G**ate **A**rray (FPGA) has become an extremely useful medium for the digital designs. It enables designers to avoid the pitfalls of nano-meter designs till the last second. Usually FPGA consists of an array of programmable logic blocks, interconnects (routing channels) and I/O cells. Logic blocks can be configured to implement sequential and complex combinational functions. Hence, influence both speed and density of an FPGA. As FPGAs are approximately 10 times less dense and three times slower than the mask programmed gate arrays [26, 27], researchers are motivated to explore new logic blocks such that these density and speed gap become as minimum as possible. Most popular logic blocks of the FPGAs are based on look-up tables (LUTs) [28, 29, 30] and design form Plessey [31, 32, 33]. A look-up table can implement any function of its inputs and accordingly described by their number of inputs. With more inputs, a LUT can implement more logic, and hence fewer logic blocks are needed. This helps in routing by asking for less area, since there are fewer connections to route between the logic blocks. However, as the number of inputs increases, the complexity grows exponentially. Plessey logic block solves this problem through cluster of components [26]∼[34].

In these consequences, this thesis investigate the reversible design methodologies of Plessey logic block and basic input-output cell of the FPGA. To our best knowledge, there are three such approaches in the literature [35, 36, 37]. However, all these schemes are not generalized and scalable. Thus, the main objective of this research is to develop a generalized structure of the reversible FPGA so that it can solve the scalability problems in addition to the fault detection capabilities. Researchers have shown, generally fault tolerant designs have more cost than the non-fault tolerant designs [9]∽[13]. However, the proposed fault tolerant scheme performs much better than the existing non-fault tolerant counterparts in terms of all the performance evaluation parameters of reversible logic synthesis.

1.1 Methodologies of the Research

While working on this research, the following important steps are followed:

- First, understanding of reversibility, its importance in low power circuitry, the basics of fault tolerance, its synthesis, the basics of quantum computation, its synthesis, various existing reversible and fault tolerant logic gates along with the quantum and transistor equivalent realizations etc.
- Designing various reversible and fault tolerant combinational circuits, analyzing several FPGA's structure and their properties, studying the existing reversible FPGA design approaches, then analyzing the designs, working procedures, advantages and shortcomings.
- Inventing and contriving the ideas for the fault tolerant Plessey logic block and basic I/O cell circuitry under a general and scalable structure. Establishing the novelty of the proposed methods through theoretical explanations, lower bounds in particular. Finally, performing a comparative study among the proposed and the existing works through simulations.

1.2 Organization of Dissertation

The next chapter (Chap.2) briefly discusses the basic definition and literature overview relating to reversible and fault tolerant computing. The study includes understanding of the reversible and fault tolerant logic gates along with their quantum equivalent realizations and applications. Chap.3 introduces FPGA and reversible FPGA design approaches. Chap.4 illustrates the design methodologies of the proposed reversible fault tolerant Plessey logic block and basic I/O cell of the FPGA. Here, an elaborated design and working procedure of the proposed

FPGA have been rendered with formal algorithmic representations and theoretical explanations. Several lower bounds on number of garbage outputs, constant inputs and quantum cost of the reversible fault tolerant FPGA have also been proposed in this chapter. Chap.5 evidenced that the proposed design is much better with respect to number of gates, garbage outputs, quantum cost, constant inputs, delay, hardware complexity and has significantly better scalability in comparison to previous approaches. Finally based on all this discussion, a conclusion is drawn in Chap.6.

1.3 Summary

This chapter demonstrates motivations and objective of this thesis. Then the methodologies of the research that is being followed are discussed here. A brief elementary instructional text of remaining chapters of this thesis have also been described.

Chapter 2

Basic Definitions and Literature Review

Grandness of the reversible and fault tolerant logic synthesis are detailed the previous chapter. Initially, this chapter defines the reversible and fault tolerant gate in addition with their fundamental properties, which will be used in the proceeding chapters. Sec.2.2 briefly defines the qubit and quantum cost. Definitions of critical path delay, hardware complexity and garbage output are presented in Sec.2.3. Sec.2.4 introduces the implementation techniques of reversible and fault tolerant logic synthesis. Sec.2.5 demonstrates the popular reversible and fault tolerant gates along with their input-output specifications, quantum equivalent representations. This section also present the designs of the transistor representations of the individual gates in order to implement the circuit using MOS transistors. Transistor simulations have been shown to check the correctness of the function of the individual gates using standard p-MOS 901 and n-MOS 902 model [38, 39, 40] with average delay of $0.030ns$ and $0.12\mu m$ channel length. All the transistor level simulations presented in this thesis, it has been considered that the signals less than $0.001ns$ stability are the glitches.

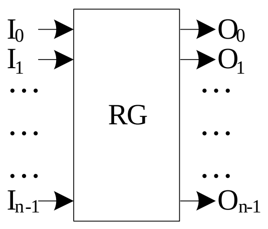


Figure 2.1: Block diagram of an $n \times n$ reversible gate

2.1 Reversible Gate

Block diagram of an $n \times n$ ***reversible gate*** is shown in Fig.2.1, which is a data stripe block that uniquely maps between input vector $I_v = (I_0, I_1, ..., I_{n-1})$ and output vector $O_v = (O_0, O_1, \ldots, O_{n-1})$ denoted as $I_v \leftrightarrow O_v$ [41, 42]. This one-to-one mapping among all inputs and outputs for each input-output sequence results in no information loss, *i.e.*, if we design a circuit using only reversible gate it will provide a unique path from each stage to its previous stage, thus entire circuit becomes information lossless. More formally, there are two prime requirements for the reversible logic synthesis that are as follows:

- There should be equal number of inputs and outputs.
- There should be one-to-one correspondence among all input-output sequences.

A ***Fault tolerant gate*** is a reversible gate which constantly preserves same parity between input and output. More specifically, an $n \times n$ fault tolerant gate clarifies the following property between input and output vectors:

$$I_0 \oplus I_1 \oplus ... \oplus I_{n-1} = O_0 \oplus O_1 \oplus ... \oplus O_{n-1} \tag{2.1}$$

The parity preserving property of Eq.2.1 allows to detect a faulty signal from the circuit's primary output. Researchers showed that, if a reversible circuit is drawn using only reversible fault tolerant gates, then the entire circuit preserves parity, thus able to detect a faulty signal [11, 13, 43, 44].

2.2 Qubit and Quantum Cost

First proposed in the 1970s, quantum computing relies on quantum physics, taking the advantages of certain physical properties of atoms or nuclei that allow them to work together as quantum bits, or qubits [7, 45, 46]. The biggest difference between the conventional bit and the qubit is that, qubit can form linear combination of states $|0>$ or $|1>$ called superposition, while the basic states $|0>$ or $|1>$ are an orthogonal basis of two-dimensional complex vector space. A superposition can be denoted as, $|\psi> = \alpha|0> + \beta|1>$ which stands for the probability of particle being measured in states 0 is $|\alpha|^2$, or results 1 with probability $|\beta|^2$, and of-course, $|\alpha|^2$+$|\beta|^2$=1 [47, 48, 49]. Hence, the information stored by a qubit are different when given different α and β. For example, let us consider a system of n bits and qubits, the system in the former is possible to store either total n 1-bit binary numbers or an n-bit binary number compared to 2^n n-qubit number in the later. Because of such properties, qubits can perform certain calculations exponentially faster than the conventional bits. This is one of the main motivation behind the quantum computing which demands its underneath circuitry be reversible.

The ***quantum cost*** for all 1×1 and 2×2 reversible gates are considered as 0 and 1, respectively [41, 50]. Hence, quantum cost of a reversible circuit is the total number of 2×2 quantum gates used in that reversible circuit. In other words, number of 2×2 Ex-OR, controlled-V (square root of NOT, *i.e.,* SRN) or controlled-V^+ (hermitian of SRN) represent the quantum cost of the circuit[1]. A controlled-V gate is shown in Fig.2.2(a). In a controlled-V gate, when the control signal $a = 0$, the qubit b will pass through the controlled part and keep it unchanged, *i.e.*, we will have $Q = b$, and if the value of $a = 1$, then $Q = V(b)$, where,

$$V = (i+1)/2 \begin{pmatrix} 1 & -i \\ -i & -1 \end{pmatrix}$$

[1]There are some exceptional cases, more detail can be found in [15, 41, 47, 48, 49, 50, 51, 52].

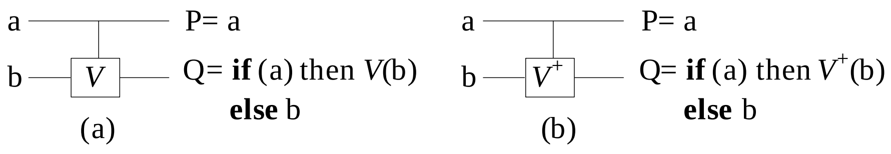

a
P= a
b
V
Q= **if** (a) then V(b) **else** b
(a)
a
P= a
b
V+
Q= **if** (a) then V+(b) **else** b
(b)

Figure 2.2: Quantum realization of controlled (a) V gate (b) V^+ gate

In the above equation, i is the basic imaginary unit, *i.e.*, $i = \sqrt{-1}$. A controlled-V^+ gate is shown in Fig.2.2(b). In a controlled-V^+ gate, when the control signal $a = 0$, the qubit b will pass through the controlled part and keep it unchanged, *i.e.*, we will have $Q = b$. If the value of $a = 1$, then $Q = V^+(b)$, where V^+ is the hermitian of V ($V^+ = V^{-1}$, when $a = 1$). Thus $V \times V$ or $V^+ \times V^+$ creates a unitary matrix of NOT gate, whereas, $V \times V^+$ or $V^+ \times V$ is an identity matrix (I) describing just a quantum wire, both of which have zero quantum cost.

2.3 Delay, Garbage Output and Hardware Complexity

In a logic circuit or reversible logic circuit, the path consists of maximum number of gates from any input to any output is known as critical path. However, it is an NP complete problem to find the critical path specially for large circuit [53, 54, 55]. Also there may be more than one critical path in a circuit. Thus, researchers used to pick the path which is the most likely candidates for the critical paths, and ***delay*** of a logic circuit is the delay of this paths. Generally in reversible logic synthesis, the most probable critical path delay is calculated considering the following two assumptions:

- Each gate performs computation in unit time. This means, each gate in the given circuit will take same amount of time for internal logic operations.

- All inputs to the circuit are known before the computation begins, *i.e.*, internal structure and operations of each gate is known before the calculation.

Unwanted or unused output of a reversible gate (or circuit) is known as ***garbage output***, *i.e.*, the output which are needed only to maintain the reversibility are known as garbage output [15, 41, 51, 56]. In other words, outputs that neither used as primary outputs nor as input to another reversible gate are garbage outputs. For example, to perform the exclusive OR between two inputs, a 2×2 reversible Feynman gate (FG) can be used. This realization produces an extra dummy output along with the desired output signal, which is needed to preserve the reversibility. This extra output is the garbage output, denoted by P in Fig.2.3. Heavy price is paid off for each garbage output. Thus, minimizing garbage outputs from the circuit is one of the main challenges for the researchers.

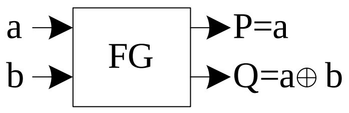


Figure 2.3: Block diagram of 2×2 reversible Feynman gate

As mentioned earlier, determining the critical path from the large circuit is not easy. So, researchers determined the ***hardware complexity*** of the circuit and is considered as a seemingly equivalent performance evaluation criteria. The number of basic operations (Ex-OR, AND, NOT etc.) needed to realize the circuit is referred to as the hardware complexity of the circuit. Actually, a constant complexity is assumed for each basic operation of the circuit, such as, α for Ex-OR, β for AND, γ for NOT etc. Then, total hardware complexity is calculated with respect to this assumed complexities, *i.e.*, in terms of α, β, and γ. For example, the hardware complexity of the circuit shown in Fig.2.3 is α, since it can be realized with a single Ex-OR only. Therefore, if a circuit only consists of x number of reversible Feynman gate, then the hardware complexity of that circuit is $x\alpha$.

2.4 Implementation Techniques

One of the early implementation techniques for reversible circuits is **C**harge **R**ecovery **L**ogic (CRL) [57]. CRL relies on explicit reversible pipelined logic gates, where the information necessary for energy recovery are used to compute a value and is provided by computing its functional inverse. **S**plit-level **C**harge **R**ecovery **L**ogic (SCRL) was proposed in [58], which uses split-level voltages. Lim *et.al* proposed **R**eversible **E**nergy **R**ecovery **L**ogic (RERL) for ultra-low-energy consumption in [59, 60]. Later they proposed **n**MOS **R**eversible **E**nergy **R**ecovery **L**ogic (nRERL) in [61]. nRERL performs better than previous methods. The nano-electronic and opto-electronic implementations of reversible gates can be found in [62, 63]. Transistor level realization of reversible circuits is a relatively new idea and extensively used by the researchers because of its scalability [64, 65, 66, 67, 68].

2.5 Popular Reversible and Fault Tolerant Gates

There are several reversible and fault tolerant gates in the literature among which, Feynman gate (FG) [69], Toffoli gate (TG) [70], Fredkin gate (FRG)[71] and Feynman double gate ($F2G$) are the most popular. Fig.2.3 represents the block diagram of reversible Feynman gate. As shown earlier, the hardware complexity of reversible Feynman gate is α, where α is the hardware complexity of two input Ex-OR operations. According to our previous discussion in Sec.2.2, the quantum cost of reversible Feynman gate is one, since, it is realized with a 2×2 primitive gate. Following subsections present the above mentioned remaining gates along with their input-output specifications, quantum equivalent realizations, transistor level implementations, working procedures and applications. CAD tools DSCH-2.7 [72] is used as simulator. Because it has a built-in extractor which generates a NETLIST [73, 74], VERILOG descriptions [75, 76, 77] and the timing diagrams.

2.5.1 Toffoli Gate:

Input and output vectors for 3×3 Toffoli gate (TG) is defined as follows: Input vector $I_v = (a, b, c)$ and output vector $O_v = (a, b, ab \oplus c)$. Block diagram of TG is shown in Fig.2.4(a). Fig.2.4(b) represents the quantum equivalent realization of TG. From Fig.2.4(b), we find that, Toffoli gate is realized with two 2×2 Ex-OR, two V and one $V+$ gates, *i.e.*, total of five 2×2 primitive gates. So, quantum cost of TG is five. However, 4×4 reversible Toffoli gate is realized with eight quantum cost [21, 52]. Transistor level realization TG required six transistors, as shown in Fig.2.4(c). Fig.2.4(d) represent the corresponding working procedure of Fig.2.4(c).

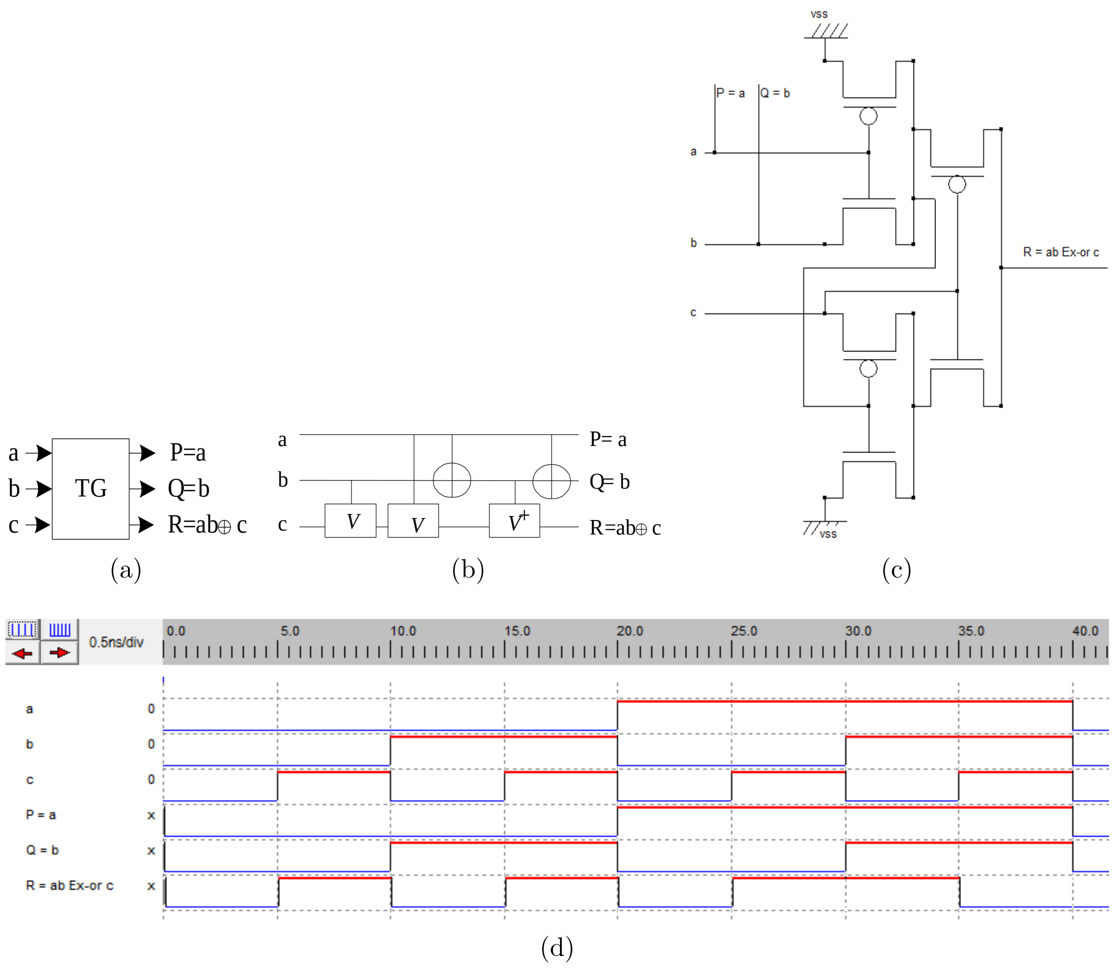


Figure 2.4: Reversible 3×3 Toffoli gate (a) Block diagram (b) Quantum equivalent realization (c) Transistor realization (d) Timing diagram

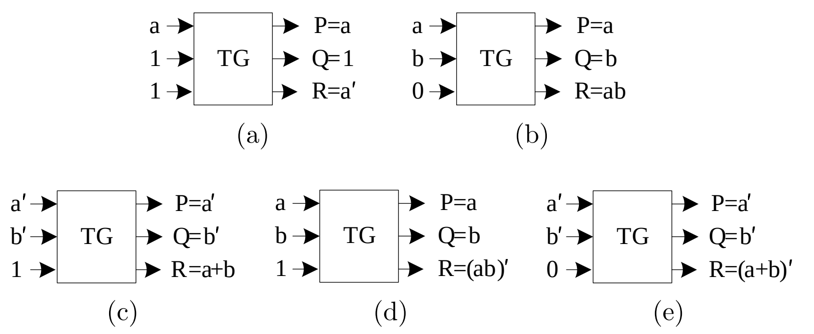


Figure 2.5: Reversible 3×3 Toffoli gate as (a) NOT gate (b) AND gate (c) OR gate (d) NAND gate (e) NOR gate

Using only reversible Toffoli gate all basic logic operations can be implemented. Figs.2.5(a)$\sim$(e), depict how a TG can be used to implement NOT, AND, OR, NAND, and NOR operations, respectively. For this reason, it is also known as a reversible universal gate. It had been shown by the researchers any logic circuit or equations can be realized using only Toffoli and Feynman gates [78, 79].

2.5.2 Feynman Double Gate

Input vector (I_v) and output vector (O_v) for 3×3 reversible Feynman double gate ($F2G$) is defined as follows: $I_v = (a, b, c)$ and $O_v = (a, a \oplus b, a \oplus c)$. Block diagram of $F2G$ is shown in Fig.2.6(a). Fig.2.6(b) represent the quantum equivalent realization of $F2G$. From Fig.2.6(b), we find that it is realized with two 2×2 Ex-OR gate. Thus, according to our previous discussion in Sec.2.2, the quantum cost of Feynman double gate is two. According to our design procedure, twelve transistors are required to realize Feynman double gate reversibly, which is shown in Fig.2.6(c). Fig.2.6(d) represents the corresponding timing diagram of the transistor realization of the reversible Feynman double gate.

Feynman double gate can be used as copying gate. For example, if the input vector (I_v) of Feynman double gate is set as $(a, b = 0, c = 0)$, then the output vector (O_v) of Feynman double gate will be (a, a, a). This helps to maintain the maximum one fan-out restriction of reversible logic synthesis [15, 41, 50, 51].

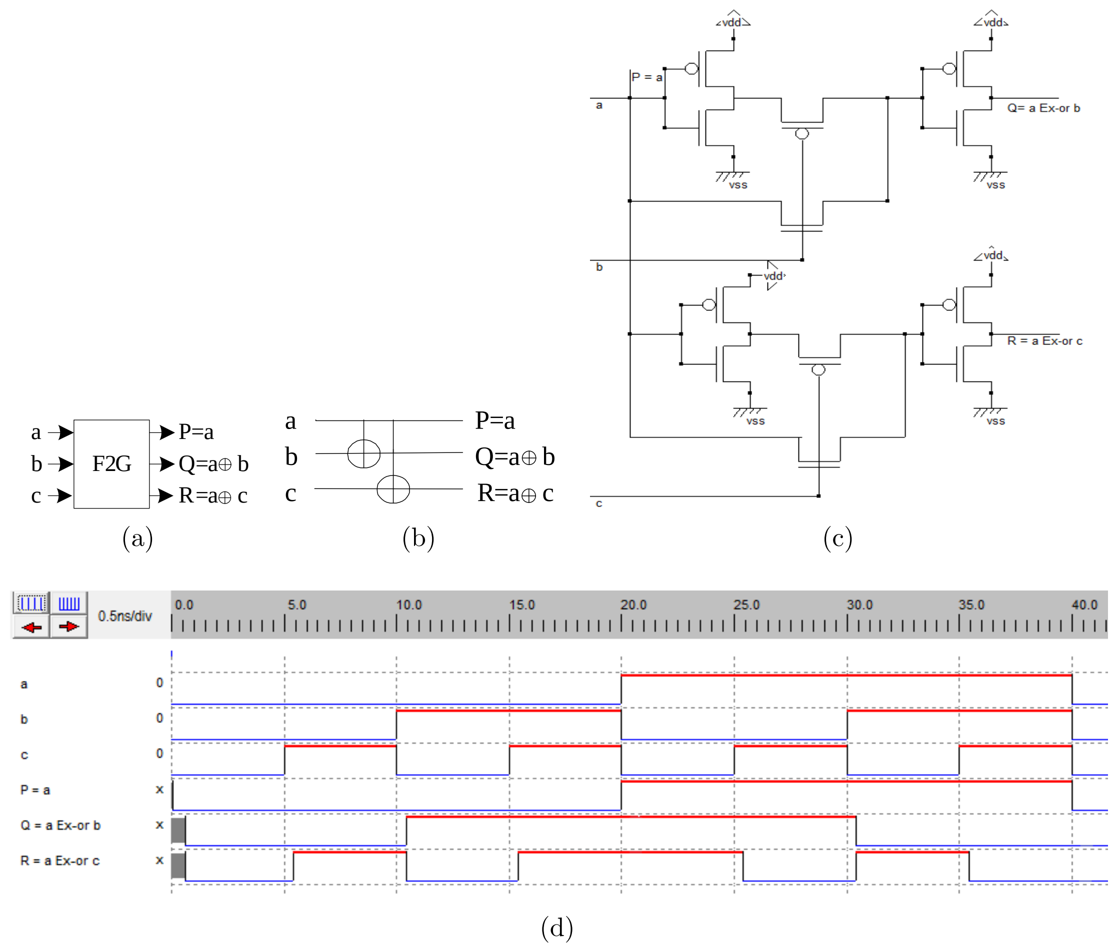


Figure 2.6: Reversible 3×3 Feynman double gate (a) Block diagram (b) Quantum equivalent realization (c) Transistor realization (d) Timing diagram

2.5.3 Fredkin Gate

Over the years, Fredkin gate had been extensively used by the researchers. This one of very few reversible gate which can realize all basic operations while preserving the parity. Input and output vectors for 3×3 Fredkin gate (FRG) are defined as follows: $I_v = (a, b, c)$ and $O_v = (a, a'b \oplus ac, a'c \oplus ab)$. Block diagram of FRG is shown in Fig.2.7(a). Fig.2.7(b) represents the quantum equivalent realization of FRG. In Fig.2.7(b), each rectangle is equivalent to a 2×2 quantum primitive, therefore its quantum cost is considered as one[41]$\sim$[56]. Thus, the quantum cost of FRG is five [66, 67]. To realize the FRG, four transistors are needed as shown in Fig.2.7(c) and its corresponding timing diagram is shown in Fig.2.7(d).

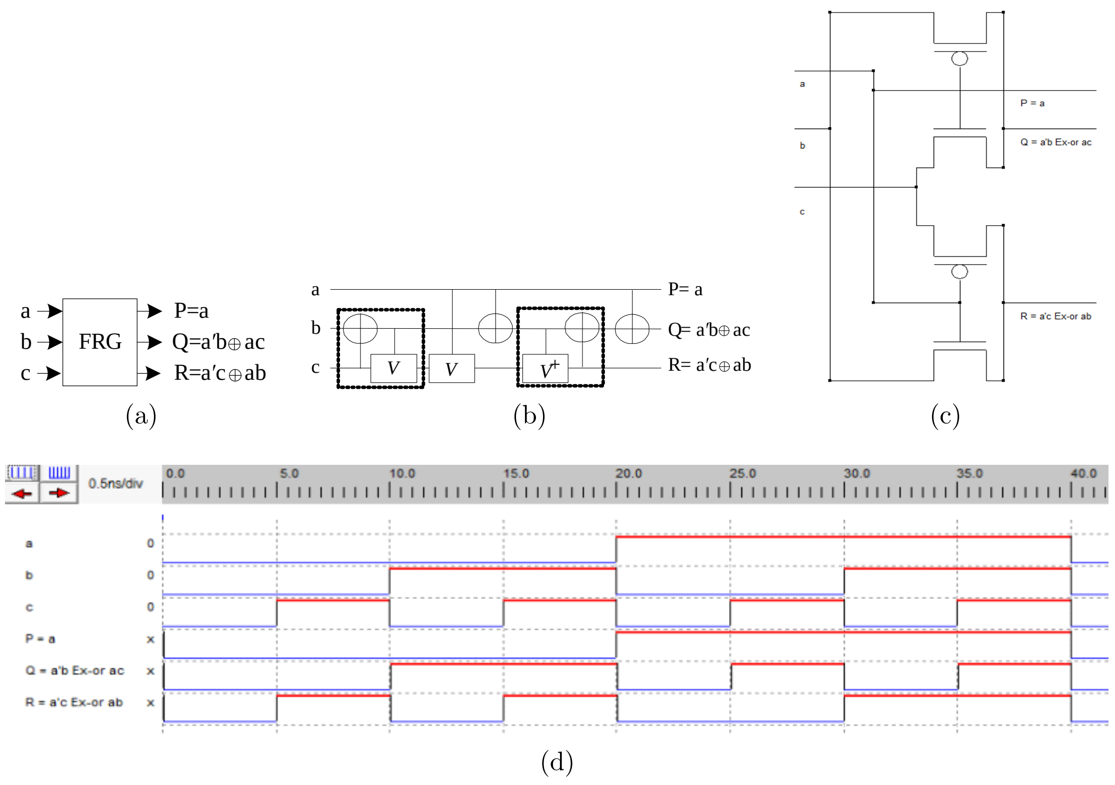


Figure 2.7: Reversible 3 × 3 Fredkin gate (a) Block diagram (b) Quantum equivalent realization (c) Transistor realization (d) Timing diagram

Fredkin gate can be used to swap 2^{nd} and 3^{rd} inputs using first input as a controlled input, *i.e.*, it can work as 2:1 Multiplexer (Mux). Referring to the input-output combinations of Fredkin gate, when $a = 1$, the inputs b and c will be swapped. The resulting value of the outputs are $Q = c$ and $R = b$. Whereas, if $a = 0$, then outputs P, Q and R are directly connected to inputs $a(= 0)$, b, and c, respectively (shown in Fig.2.8(a)). Fredkin gate is also an universal gate like TG. The universality of Fredkin gate is shown in Figs.2.8(b)∼(d).

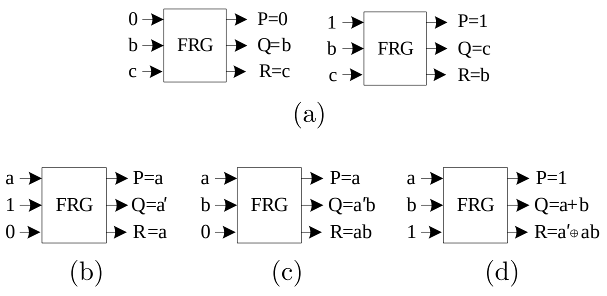


Figure 2.8: FRG as (a) 2:1 Mux (b) NOT gate (c) AND gate (d) OR gate

Table 2.1: Truth table for $F2G$ and FRG

Input			Output of $F2G$			Output of FRG			
A	B	C	P	Q	R	P	Q	R	Parity
0	0	0	0	0	0	0	0	0	Even
0	0	1	0	0	1	0	0	1	Odd
0	1	0	0	1	0	0	1	0	Odd
0	1	1	0	1	1	0	1	1	Even
1	0	0	1	1	1	1	0	0	Odd
1	0	1	1	1	0	1	1	0	Even
1	1	0	1	0	1	1	0	1	Even
1	1	1	1	0	0	1	1	1	Odd

Among the reversible gates discussed above, Fredkin and Feynman double gate comply with the rule of Eq.2.1. Therefore, according to our previous discussion in Sec.2.1, if a circuit is designed using only Fredkin and Feynman double gates the circuit will inherently become fault tolerant. The parity preserving (fault tolerant) property of Fredkin and Feynman double is shown in Table.2.1.

2.6 Summary

A brief literature overview and the related terminologies regarding reversible and fault tolerant logic synthesis are presented in this chapter. Definitions of three most popular reversible and fault tolerant logic gates devoted here as well. Equivalent quantum and transistor representations of those reversible and fault tolerant logic gates have also been depicted in this chapter.

Chapter 3

Existing FPGAs

Essential technical background on FPGA are presented in this chapter which is required to understand the proposed work. Secs.3.1 and 3.2 present the design and working procedure of basic irreversible and reversible FPGAs, respectively.

3.1 Basic Structure's of FPGA

A crucial part of FPGA creation lies in their architecture, shown in Fig.3.1(a). This architecture governs the nature of its programable logic functionality. As shown in the figure, user-programable routing network provides connections between the logic blocks and the logic performs the combinational functions as discussed in Chap.1. The architecture of the Plessey logic block is shown in Fig.3.1.(b). Here, the basic block is a two-input NAND gate. The logic is formed by connecting NAND gate to implement the desired functions. Other essential components of Plessey logic block are, **R**andom **A**ccess **M**emory (RAM), 8-to-2 **Mu**ltiple**x**er (Mux) and D-Latch, where the RAM requires an n-to2^n decoder and the write enable master-slave flip-flops. More detail can be found in [26, 33, 34, 80, 81].

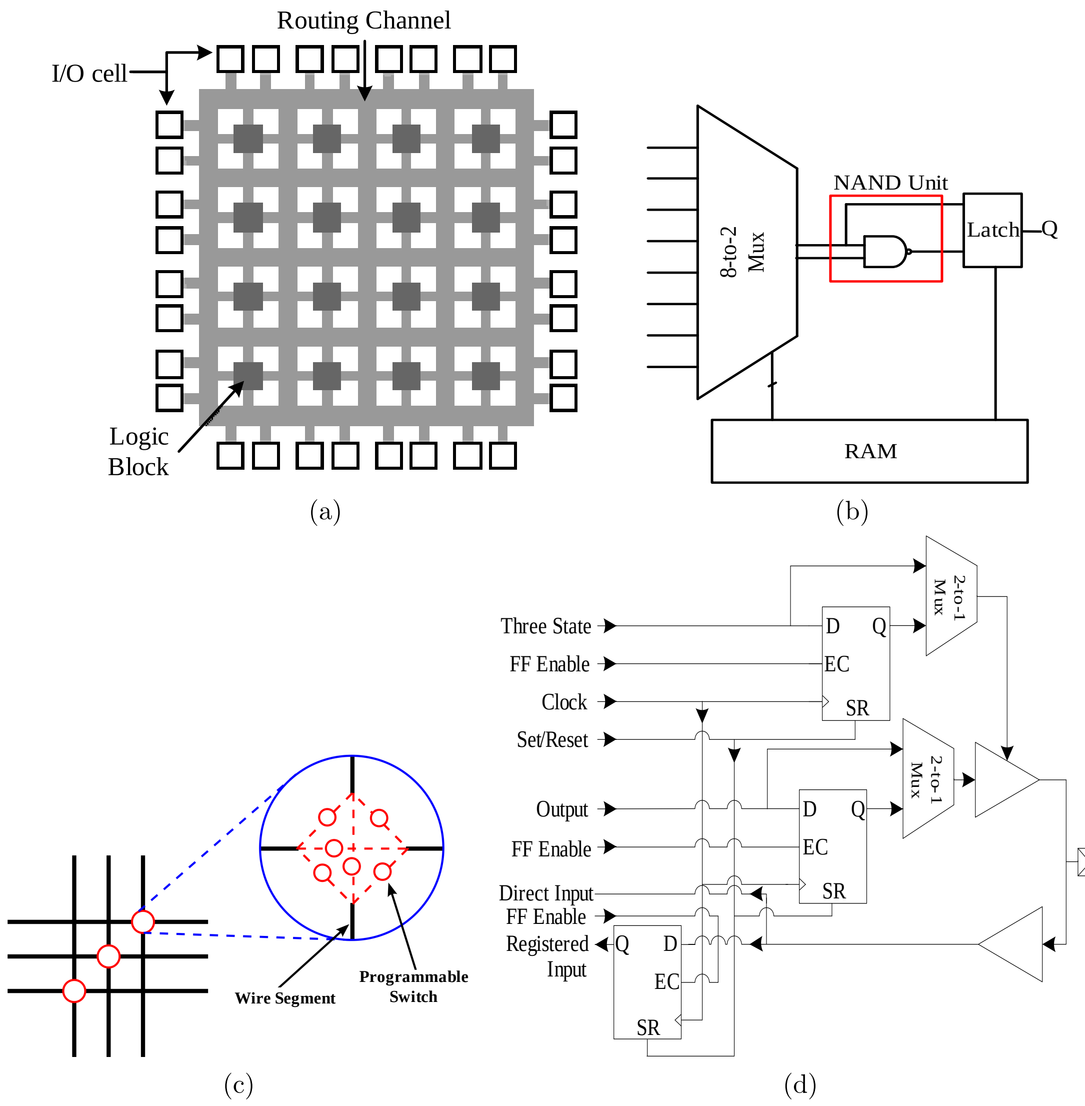


Figure 3.1: Block diagram of (a) Basic irreversible FPGA (a) Plessey logic block (c) Routing channel (d) Basic input-output block

Figs.3.1(c) and (d) represent the magnified diagrams of the routing channel and basic input-output block of the FPGA, respectively. From Fig.3.1(d), we find that, basic input-output block consists of 2-to-1 Muxs, input buffer, output driver and D-latches. The latches of basic input-output block are identical to the latches of Plessey logic block. From Fig.3.1(d), we also find that the SET/RESET and clock signals of all the latches are common *i.e.*, are the shared signals, which are generally are adjusted by the programmers/designers [82].

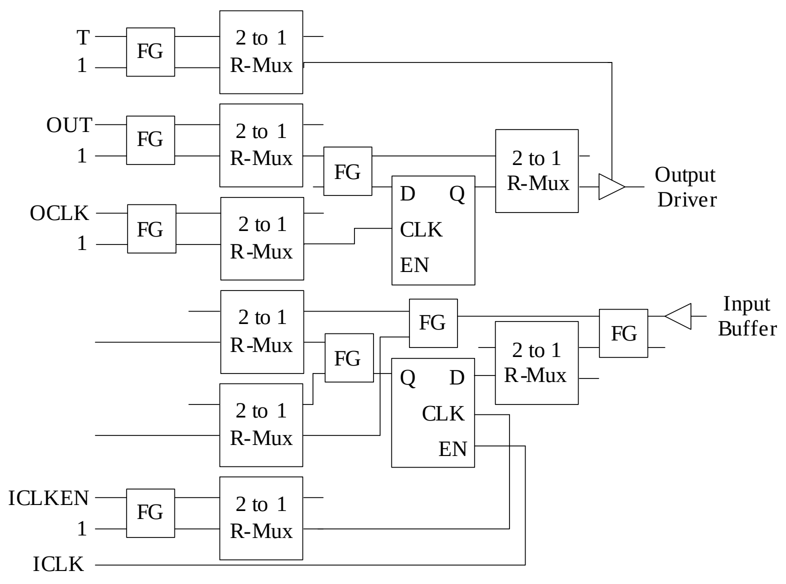


Figure 3.2: Block diagram of existing reversible input-output pad [37]

3.2 Reversible FPGAs

To the best of our knowledge, three papers have been published so far on reversible FPGA. First paper on reversible FPGA had been published in 2009 [36]. This paper showed a reversible Plessey logic block of the FPGA. Recently in 2011, an improved design of reversible plessey logic block had been proposed in [35]. Reversible input-output block had been attempted in [37], however had not completed. This design is shown in Fig.3.2. The incompleteness of the existing input-output pad [37] can easily be asserted from Fig.3.2. More importantly, none the existing designs [35, 36, 37] are generalized, compact and scalable.

To design all the components of reversible Plessey logic block, authors in [36] used reversible NH, BSP, Feynman (FG) and Toffoli (TG) gates. The quantum cost of FG, NH, and TG are 1, 4, and 5, respectively. However, the quantum cost of BSP gate is not shown in [36]. From the architecture of BSP gate it can be determined that it is the combinations of 3×3 and 4×4 Toffoli gate, thus its quantum cost is 13 (referring to the Sec.2.5).

On the other hand, the reversible plessey logic block from [35] consist of reversible Mux and Peres gates (MG and PG, respectively) along with the above mentioned gates. The quantum cost of both Mux and Peres gate are 4. More detail on reversible NH, BSP, FG, TG, MG and PG can also be found in [35, 36, 37, 51]. In Appendix A, the block diagrams, quantum equivalent realizations, transistor realizations and the timing diagrams of these gates are given for the references.

3.3 Summary

This chapters demonstrate the working procedure of basic irreversible FPGA with the help of their structures. The design approaches towards the reversible FPGA have also been presented here.

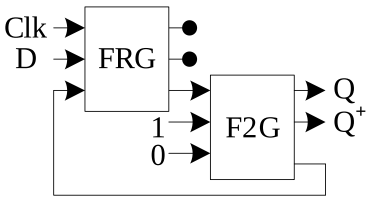
Chapter 4

Proposed Reversible Fault Tolerant FPGA

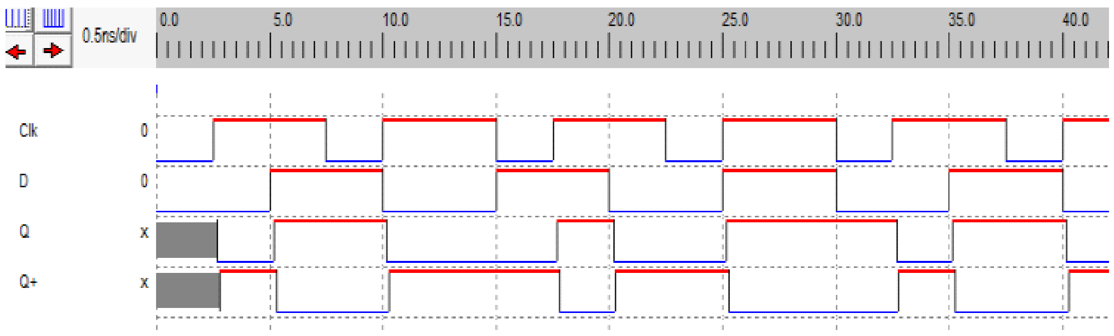
This chapter illustrates the design layouts, algorithms, theoretical properties and working procedures of the proposed reversible fault tolerant FPGA. As mentioned in Chap.3, Plessey logic block consists of NAND Unit, D-latch, 8-to-2 multiplexer (Mux) and random access memory (RAM). The RAM requires underneath n-to-2^n decoder and write enable master-slave flip-flop circuitry. Therefore, Sec 4.1 proposes reversible fault tolerant D-latch of the FPGA. Reversible fault tolerant write enable master-slave flip-flop of the FPGA is proposed in Sec.4.2. Sec.4.3 demonstrates the proposed fault tolerant n-to-2^n decoder of the FPGA. In Sec.4.4, the $4n$-to-n reversible fault tolerant Mux of the FPGA has been proposed. RAM for fault tolerant FPGA is proposed in Sec.4.5. To design the RAM for fault tolerant Plessey logic block, we extend our proposed reversible design of the RAM [83] to the reversible fault tolerant RAM with more optimally. Then, combining all these components the architecture of the fault tolerant Plessey logic block is presented in Sec.4.6. The chapter ends by proposing reversible fault tolerant basic input-output block of the FPGA in Sec.4.7.

4.1 Proposed Reversible Fault Tolerant D-Latch for FPGA

Fig.4.1(a) represents the architecture of proposed reversible fault tolerant D-latch for the FPGA, from where we find that, a Fredkin and a Feynman double gate are used. The Fredkin gate is needed to produce Q, whereas Feynman double gate produces Q^+ and Q. The equations for the proposed fault tolerant latch are $Q_n = D.Clk + Clk'.Q_{n-1}$ and $Q^+ = Q'$ (n is time varying), which are used to capture the logic level when the data is present and the clock is SET.



(a)



(b)

Figure 4.1: (a) Block diagram of the proposed reversible fault tolerant D-Latch (b) Timing diagram of the proposed reversible fault tolerant D-Latch

Corresponding timing diagram of the proposed latch is shown in Fig.4.1(b). The circuit is simulated with the average current of $0.001mA$ and the average power of $0.001mW$. From Fig.4.1(b), we find that when clock is low the value of D has no effect on Q (0 to $2.5ns$). However, when clock is high Q follows D, *i.e.*, are in transparent mode (2.5 to $7.5ns$). If the clock is low again, then Q retains the previous value of D, *i.e.*, the value of D when clock went from high to low ($1 \rightarrow 0$)

Table 4.1: Comparison of reversible D-Latch

	GT	GO	QC	HC	UD	TR
Existing Circuit [36]	2	2	9	$6\alpha + 6\beta + 3\gamma$	2	18
Existing Circuit [37, 84]†	2	2	8	$5\alpha + 4\beta + 2\gamma$	2	22
Existing Circuit [85]	5	3	12	*	3	*
Proposed Circuit	2	2	7	$4\alpha + 4\beta + 2\gamma$	2	16

† The D-latch designs from [37] and [84] are identical.
*[85] has circuitry of quantum level. So, we don't compare the proposed work with [85] in-terms of HC and TR.

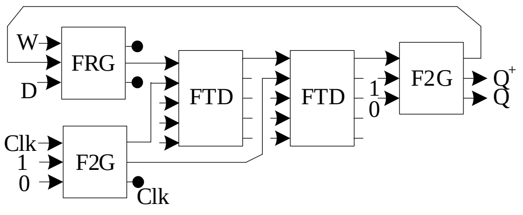
last time (7.5 to $10ns$, 15 to $17.5ns$ and so on). Table.4.1 shows the comparison of proposed latch with the existing latches. In this table and all the following tables, we consider, GT = No of Gates, GO = Garbage Outputs, QC = Quantum cost, HC = Hardware Complexity, UD = Unit Delay of the critical path, and TR = No of Transistors. Throughout this dissertation, it is also considered that α, β, and γ are the hardware complexity of two-input Ex-OR, AND and NOT calculations, respectively. From Table.4.1, we find that the proposed design performs better than the existing designs. Since, both the reversible gates used to design the proposed latch are parity preserving (shown in Sec.2.5), hence, the proposed latch inherently preserves parity, *i.e.*, fault tolerant.

Lemma 1: A reversible fault tolerant D-latch for FPGA can be realized with 2 reversible fault tolerant gates, 2 garbage outputs, 7 quantum cost, $4\alpha + 4\beta + 2\gamma$ hardware complexity and 16 transistors.

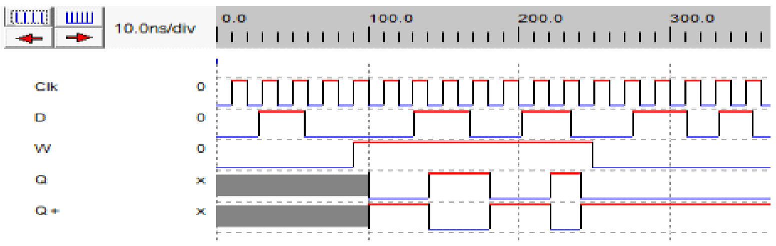
Proof: Fault tolerant D-Latch is realized with a FRG and a $F2G$, where the FRG produces 2 garbage outputs only. Quantum cost and hardware complexity of a FRG and $F2G$ are 5, $2\alpha+4\beta+2\gamma$ and 2, 2α, respectively (referring to Sec.2.5). According to the design procedure, 4 and 12 transistors are required for their transistor level realization. So, fault tolerant D-latch can be realized with 1+1 = 2 gates, 1+1 = 2 garbage outputs, 5+2 = 7 quantum cost, $2\alpha+4\beta+2\gamma+2\alpha = 4\alpha+4\beta +2\gamma$ hardware complexity and 4+12 = 16 transistors. It holds Lemma 1. □

4.2 Proposed Reversible Fault Tolerant Write Enable Master-Slave Flip-Flop for FPGA

Write enable master-slave flip-flop is the most significant element for a RAM of FPGA. Fig.4.2(a) shows the architecture of the proposed **R**eversible **F**ault tolerant write enable **M**aster-**S**lave **F**lip-**F**lop (RFMSFF). From now on, we denote a reversible fault tolerant write enable master-slave flip-flop as RFMSFF which works on two modes, *e.g.*, read and write, since data are both read from and written into RAM of FPGA. In Fig.4.2(a), W works as the write enable bit. From block diagram, we find that the proposed RFMSFF consists of a FRG, two $F2G$ and two reversible **F**ault **T**olerant **D**-latch (FTD) (proposed in Sec.4.1). Corresponding Timing diagram of Fig.4.2(a) is shown in Fig.4.2(b). From Fig.4.2(b), we find that the value of D has no effect on Q or Q^+ (0 to 100ns) until the write enable bit (W) is SET. However, when W is in high state, Q follows D (100 to 250ns). Also, Q keeps the previous value of D when W gets low again (250 to 365ns). This previous value is the value of D, when W is changed into low form high last time (250ns), and of-course, the values are changed according to the clock pulses.



(a)



(b)

Figure 4.2: Proposed reversible fault tolerant write enable master-slave flip-flop (a) Architectural block diagram (b) Timing diagram

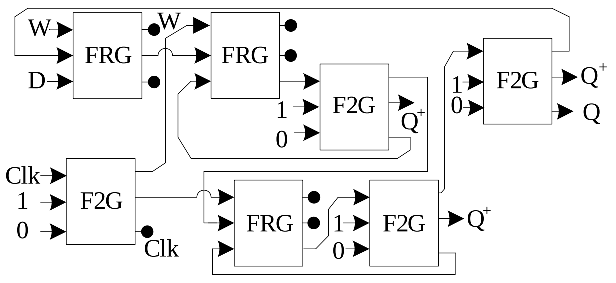


Figure 4.3: Alternative architectural block diagram of the proposed reversible fault tolerant write enable master-slave flip-flop

Table 4.2: Comparison of reversible write enable master-slave flip-flop

	GT	GO	QC	HC	UD
Existing Circuit [36]	7	9	25	$14\alpha + 20\beta + 8\gamma$	7
Proposed Circuit	7	7	23	$14\alpha + 12\beta + 6\gamma$	6

An alternate design of the proposed RFMSFF without FTD is shown in Fig.4.3. This work is already published in 4^{th} IEEE International Conference on Computer Science and Information Technology, Jun–2011 [83]. From Fig.4.3, we find that this alternative design consists of three FRGs, four $F2G$s and produces seven garbage outputs. From above two figures, we also find that both designs are identical in all aspects of evaluation parameters of reversible logic synthesis, thus, either of them can be used for the RAM of the FPGA.

Table.4.2 shows the comparison of proposed reversible fault tolerant write enable master-slave flip-flop with the existing reversible write enable master-slave flip-flop. This table evidenced that the proposed design performs better than the existing one. In [36], unused clock pulse(s) (Clk) was(were) not considered as garbage output. However, according to our previous discussion in Sec.2.3, the unused clock pulse(s) should be considered as garbage output(s). Considering unused clock pulse as garbage outputs, design [36] has 7 garbage outputs, whereas, the proposed design has 6 garbage outputs. Most interestingly, proposed write enable masters-slave flip-flops can detect the faulty signal at its primary outputs, since the proposed circuits are designed using only reversible fault tolerant gates. Fault tolerant design of write enable master-slave flip-flop is a novel idea in the literature to our best knowledge. The theoretical explanations of the reversible fault tolerant write enable master-slave flip-flop for the FPGA with respect to number of gates, garbage outputs, quantum cost and hardware complexity are presented below.

Lemma 2: A reversible fault tolerant write enable master-slave flip-flop for FPGA can be realized with seven gates and seven garbage outputs.

Proof: According to the design procedures, reversible fault tolerant write enable master-slave flip-flop for FPGA is realized with one FRG, two $F2G$s and two FTDs. In Lemma 1, we have proved that a FTD is realized with one FRG and one $F2G$, where each FTD produces two garbage outputs. In addition to that, the remaining $F2G$s and FRGs of the RFMSFF produce one and two garbage outputs, respectively. Hence, a reversible fault tolerant write enable master-slave flip-flop for the FPGA can be realized with 1+2+(2×2) = 7 fault tolerant gates, and 2+1+(2×2) = 7 garbage outputs. □

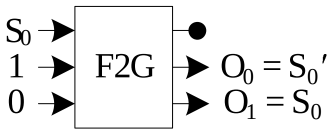
Lemma 3: A reversible fault tolerant write enable master-slave flip-flop for FPGA can be realized with 23 quantum cost and $14\alpha + 12\beta + 6\gamma$ hardware complexity.

Proof: In Lemma 2, we have proved that a reversible fault tolerant write enable master-slave flip-flop for FPGA can be realized with 7 reversible fault tolerant gates. Among which, 3 are FRGs and 4 are $F2G$s. Quantum cost and hardware complexity of a FRG are 5 and $2\alpha + 4\beta + 2\gamma$, respectively. On the other hand, quantum cost and hardware complexity of a $F2G$ are 2 and 2α, respectively (referring to Sec.2.5). Hence, a reversible fault tolerant write enable master-slave flip-flop for FPGA can be realized with (3×5)+(4×2) = 23 quantum cost, and $3\times(2\alpha+4\beta+2\gamma)+(4\times 2\alpha) = 14\alpha+12\beta+6\gamma$ hardware complexity. □

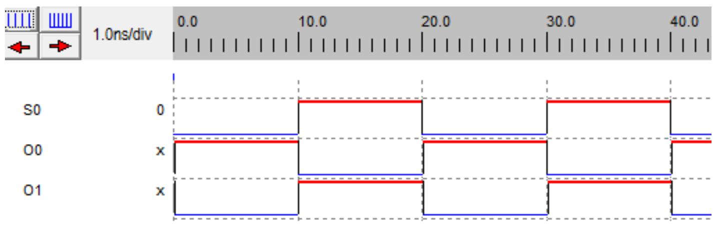
4.3 Proposed Reversible Fault Tolerant n-to-2^n Decoder for FPGA

Decoder is the collection of logic gates fixed up in a specific way such that, for an input combination, all output *terms* are low except one. These *terms* are the *minterms* which use a variable once, and once only [86, 87]. Thus, when an input combination changes, two outputs will change. Let there be n inputs. So, number of outputs will be 2^n. In other words, there will be one line at the output for each possible input. There are several designs of reversible decoders in the literature, but the design [18] is the only reversible decoder which can preserve parity too. However, the design [18] is not generalized and compact. Therefore, this section propose a reversible fault tolerant decoder, which is compact and generalized.

Considering the simplest case, $n = 1$, we have a 1-to-2 decoder. Only a $F2G$ can work as 1-to-2 **R**eversible **F**ault tolerant **D**ecoder (RFD) as shown in Fig.4.4(a). The corresponding timing diagram of Fig.4.4(a) is shown in Fig.4.4(b). From now on, we denote a reversible fault tolerant decoder as RFD.



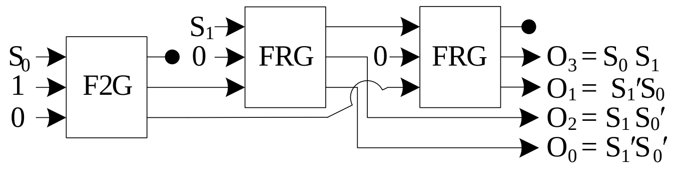
(a)



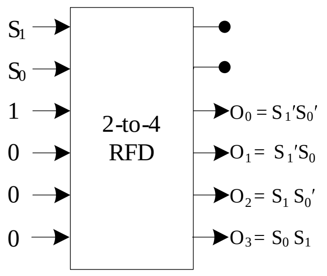
(b)

Figure 4.4: Proposed 1-to-2 RFD (a) Block diagram (b) Timing diagram

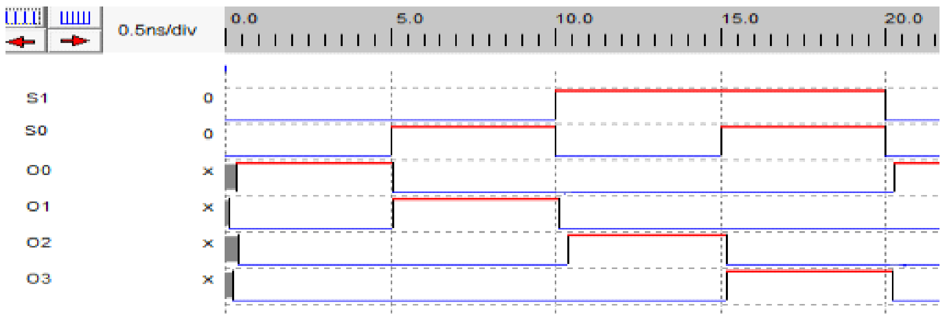
Increasing n from 1 to 2, then 2 to 3 we have 2-to-4 and 3-to-8 decoders. Figs.4.5(a) and (d) represent the architectures of 2-to-4 and 3-to-8 RFD, and their timing diagrams are shown in Figs.4.5(c) and (e), respectively. From Fig.4.5(d) we find that 3-to-8 RFD is designed using 2-to-4 RFD, thus a schema of Fig.4.5(a) is created which is shown in Fig.4.5(b).



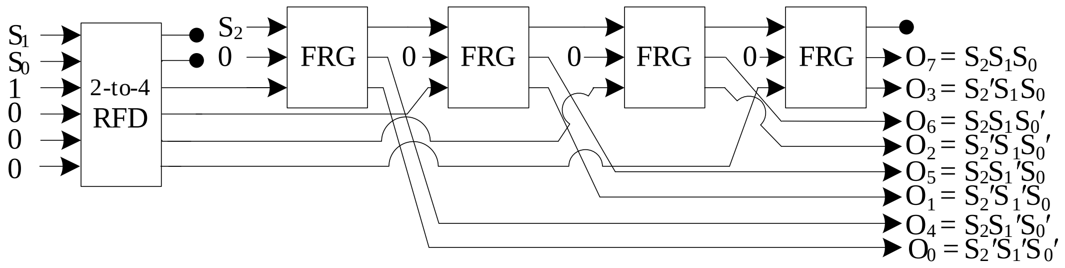
(a)



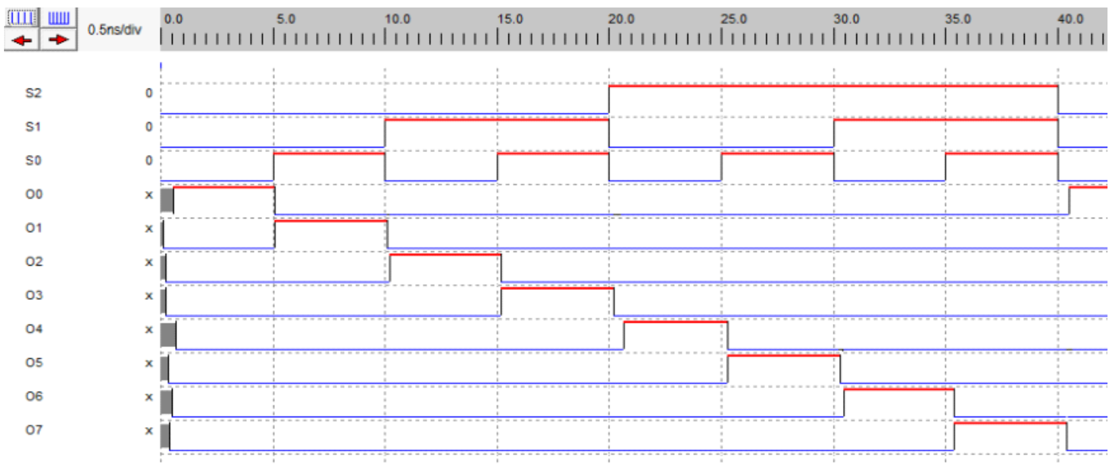
(b)



(c)



(d)



(e)

Figure 4.5: (a) Block diagram of the proposed 2-to-4 RFD. (b) Schematic diagram of the proposed 2-to-4 RFD. (c) Timing diagram of the proposed 2-to-4 RFD. (d) Block diagram of the proposed 3-to-8 RFD. (c) Timing diagram of the proposed 3-to-8 RFD.

Algorithm 1: Algorithm for the proposed reversible fault tolerant n-to-2^n decoder for FPGA, ***RFD(S, F2G, FRG)***

```
Input  : Data input set $S(S_0, S_1, ..., S_{n-1})$
         Feynman double gate ($F2G$) and Fredkin gate ($FRG$)
Output: $n$-to-$2^n$ reversible fault tolerant decoder circuit
begin
    $i = input$
    $o = output$
    for $j \leftarrow 0$ to $n - 1$ do
        if $j = 0$ then
            $S_j \rightarrow first.i.F2G$, $1 \rightarrow second.i.F2G$, $0 \rightarrow third.i.F2G$
        end if
        else
            $S_j \rightarrow first.i.FRG$, $0 \rightarrow second.i.FRG$
            if $j$=2 then
                $third.o.F2G \rightarrow third.i.FRG$
            end if
            else
                call RFD($j$-1), $RFD.o.j \rightarrow third.i.FRG_j$
            end if
        end if
    end for
    return $FRG.o.3$ and $FRG.o.2 \rightarrow$ desired output
           remaining $F2G.o$ and $FRG.o \rightarrow$ garbage output.
end
```

Algorithm 1 presents the design procedure of the proposed n-to-2^n RFD. Primary input to the algorithm are n control bits. Line 6 of the proposed algorithm assigns the input to the Feynman double gate for the first control bit (S_0), whereas line 9 assigns first two inputs to the Fredkin gates for all the remaining control bits. Line 10-12 assign third input to the Fredkin gate for $n = 2$, while line 13-15 assign third input to the Fredkin gate through a recursive call to previous RFD for $n > 2$. Line 14-15 returned the outputs. The worst case time complexity of this algorithm is $O(n)$, where n is the number of data bits. To our best knowledge, no other polynomial time algorithm exists for n-to-2^n reversible fault tolerant decoder in the literature. According to the Algorithm 1, the architecture of the proposed n-to-2^n RFD is shown in Fig.4.6.

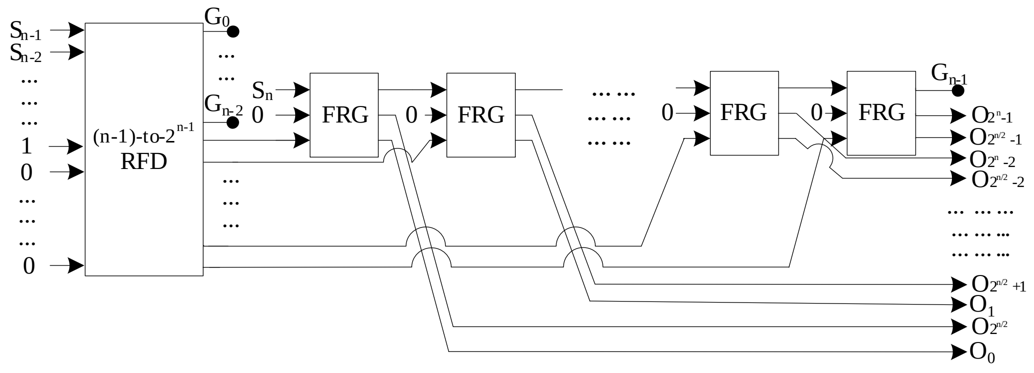


Figure 4.6: Block diagram of the proposed n-to-2^n RFD.

Table 4.3: Comparison of reversible fault tolerant decoders

	GT	GO	QC	HC	UD
2-to-4 Existing Circuit [18]	3	2	15	$6\alpha + 12\beta + 6\gamma$	3
2-to-4 Proposed Circuit	3	2	12	$6\alpha + 8\beta + 4\gamma$	2
3-to-8 Existing Circuit [18]*	≥ 7	≥ 3	≥ 35	$\geq 14\alpha + 28\beta + 14\gamma$	≥ 7
3-to-8 Proposed Circuit	7	3	32	$14\alpha + 24\beta + 12\gamma$	4

* The design is not generalized one, *i.e.*, it is not an n-to-2^n decoder.

Table.4.3 shows a comparative study of proposed decoders with existing one. From this table we find that proposed decoders perform better than its existing counterparts. Sec.2.5 presents the designs of the transistor representations of FRG and $F2G$. These representations are finally used to get the MOS circuit of the proposed decoder. The timing diagrams of individual circuits also show the correctness of the function of the decoder. Lower bounds on the number of garbage outputs, constant inputs and quantum cost of the reversible fault tolerant decoder is given below.

Theorem 1: An n-to-2^n reversible fault tolerant decoder can be realized with at least n garbage outputs and 2^n constant inputs, where n is the number of data bits.

Proof: An n-to-2^n decoder has n inputs and 2^n outputs. Thus, to maintain the property of reversibility, there should be at least $(2^n - n)$ constant inputs. However, this combinations doesn't preserve parity. To preserve the parity, at least n more constant inputs are needed. So, there should be at least n garbage outputs. $\square$

Example 1: Let the value of n be 1. Then, we have the 1-to-2 reversible fault tolerant decoder. As shown in Chap.2, for a reversible circuit it is necessary to maintain the one-to-one correspondence between input and output vectors. Thus, any reversible circuit should have equal number of inputs and outputs. In the 1-to-2 decoder, there are 2 primary outputs (O_0, O_1) but 1 input (S_0). So, 1-to-2 reversible decoder should have at least 1 constant input. The value of this constant input can be either 0 or 1. Table.4.4[1] shows that whatever the value of this constant input, it will never be able to preserve the parity between input and output vectors, which is the prime requirement of the reversible fault tolerant logic circuit. Therefore, to preserve the parity for the 1-to-2 reversible fault tolerant decoder we need at least one more constant input, *i.e.*, total of 2 constant inputs.

Table 4.4: 1-to-2 decoder with 1 constant input

Inputs		Outputs		
CI	S_0	O_0	O_1	Parity
0	0	1	0	I_p=E, O_p=O
0	1	0	1	I_p=O_p
1	0	1	0	I_p=O_p
1	1	0	1	I_p=E, O_p=O

Next, we must prove the existence of combinational circuit which can realize the reversible fault tolerant 1-to-2 decoder by 2 constant inputs. This can easily be accomplished by the circuit shown in Fig.4.4(a). It can be verified that Fig.4.4(a) is reversible and fault tolerant with the help of its corresponding truth table.

Now, in 1-to-2 reversible fault tolerant decoder there are at least 2 constant inputs and 1 primary input, *i.e.*, total of 3 inputs. Thus, in the 1-to-2 reversible fault tolerant decoder there should be at least 3 outputs, otherwise it will never comply with the properties of reversible parity preserving circuit. Among these 3 outputs, only 2 are primary outputs. So, remaining 1 output is the garbage output, which holds Theorem 1 for $n = 1$.

[1]In this table and the following tables, it is considered that, E = Even, O = Odd, I_p = Input parity, O_p = Output parity, CI = Constant input.

Theorem 2: A 2-to-4 reversible fault tolerant decoder can be realized with at least 12 quantum cost.

Proof: A 2-to-4 decoder has 4 different 2×2 logical AND operations. A reversible fault tolerant AND[2] operation requires at least 3 quantum cost. Therefore, 2-to-4 reversible fault tolerant decoder can't be realized with less than 12 quantum cost. □

Example 2: Fig.4.5(a), the discussions in Secs.2.2 and 2.5 are the proof for the existence of 2-to-4 reversible decoder with 12 quantum cost. Next, we want to prove that it is not possible to realize a reversible fault tolerant 2-to-4 decoder fewer than 12 quantum cost. In the 2-to-4 decoder, there are 4 different AND operations ($S_1'S_0'$, $S_1'S_0$, S_1S_0', S_1S_0). It will be enough if we are able to prove that it is not possible to realize a reversible fault tolerant AND with fewer than three quantum cost. Consider,

(i) If we make use of one quantum cost to design the AND, that of course is not possible according to our discussion in Sec.2.2.

(ii) If we make use of two quantum cost to design AND, then we must make use of two 1×1 or 2×2 gates. Apparently two 1×1 gates can't generate the AND. Aiming at two 2×2 gates, we have two combinations, which are shown in Figs.4.7(a) and (b). In Fig.4.7(a), the output must be (a, ab) if the inputs are (a, b). The corresponding truth table is shown in Table.4.5.

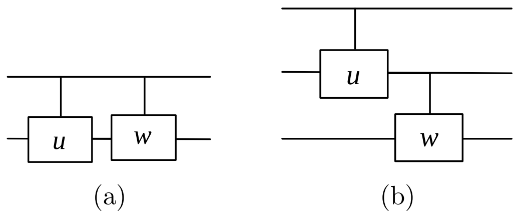


Figure 4.7: Combinations of the two 2 ×2 quantum primitive gates

[2]AND means a two-input logical AND.

Table 4.5: Truth table of Fig.4.7(a)

a	b	a	ab
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	1

From Table.4.5, we find that, first and second row have the same output combinations for different input combinations, *i.e.*, outputs are not at all unique to its corresponding input combinations. So it can't achieve the reversible AND. For Fig.4.7(b) if inputs are (a, b, c) then, the outputs of the lower level will be offered to the next level as a controlled input, this means that second output of Fig.4.7(b) have to be ab, otherwise it will never be able to get output ab, since, third output of Fig.4.7(b) is controlled by the second output, thereby according to Table.4.6, we can assert that the second combination is impossible to realize the AND no matter how we set the third output of Fig.4.7(b) (third column of Table.4.6), the input vectors will never be one-to-one correspondent with the output vectors. Therefore, we can conclude that, a combinational circuit for reversible fault tolerant 2×2 logical AND operation can't be realized with less than three quantum cost.

Table 4.6: Truth table of Fig.4.7(b)

a	b	c	a	ab	
0	0	0	0	0	
0	0	1	0	0	
0	1	0	0	0	
0	1	1	0	0	
1	0	0	1	0	
1	0	1	1	0	
1	1	0	1	1	
1	1	1	1	1	

The above example clarifies the lower bound in terms of quantum cost of 2-to-4 RFD. Similarly, it can be proved that the n-to-2^n RFD can be realized with $5(2^n\text{-}\frac{8}{5})$ quantum cost, when $n \geq 1$, and by assigning different values to n, the validity of this equation can be proved.

Lemma 4: An n-to-2^n RFD can be realized with (2^n-1) reversible fault tolerant gates, where n is the number of data bits.

Proof: According to the Algorithm 1, an n-to-2^n RFD requires an $(n-1)$-to-2^{n-1} RFD plus n number of Fredkin gates, which requires an $(n-2)$-to-2^{n-2} RFD plus $(n-1)$ Frdekin gates, and so on, till we reach to 1-to-2 RFD. 1-to-2 RFD requires a reversible fault tolerant Feynman double gate only. Thus, total number of gates required for an n-to-2^n RFD is

$$1+2+4+\cdots+n^{th}\, term = \frac{2^0(2^n-1)}{(2-1)} = 2^n-1$$

□

Example 3: From Fig.4.5(d), we find that the proposed 3-to-8 RFD requires total number of 7 gates. If we replace n with 3 in Lemma 4, we get the value 7 as well.

Lemma 5: An n-to-2^n RFD can be realized with $(2^{n+1}-2)\alpha + (2^{n+2}-8)\beta + (2^{n+1}-4)\gamma)$ hardware complexity, where n is the number of data bits.

Proof: In Lemma 4, we proved that an n-to-2^n RFD can be realized by using a $F2G$ and (2^n-2) FRGs. The hardware complexity of a FRG and a $F2G$ are $2\alpha + 4\beta + 2\gamma$ and 2α, respectively. So, total hardware complexity for n-to-2^n RFD is

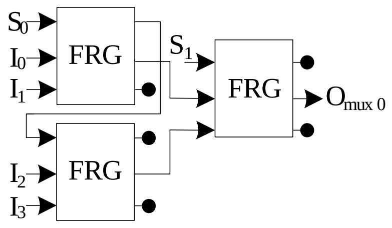
$$(2^n-2)(2\alpha+4\beta+2\gamma)+2\alpha = (2^{n+1}-2)\alpha + (2^{n+2}-8)\beta + (2^{n+1}-4)\gamma$$

□

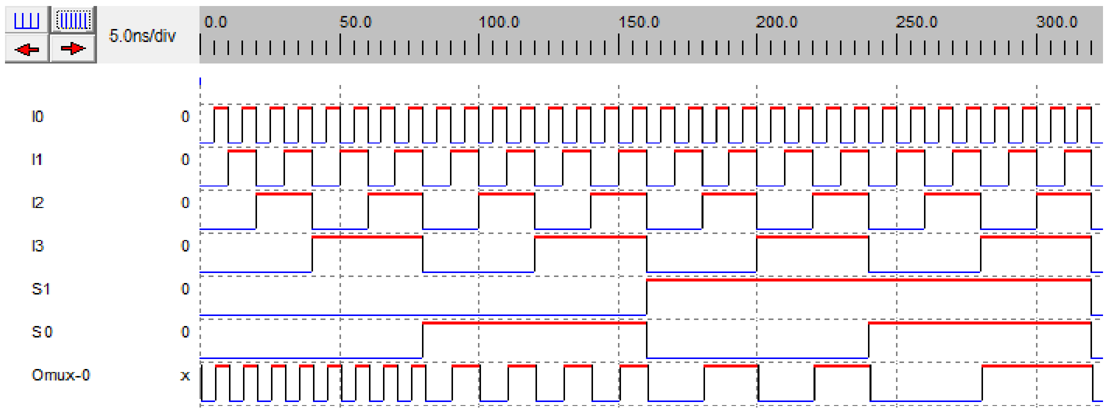
Example 4: Fig.4.5(d) presents that the proposed 3-to-8 RFD requires 6 FRG and 1 $F2G$. According to our previous discussion in Sec.2, hardware complexity of a $F2G$ and a FRG are 2α and $2\alpha+4\beta+2\gamma$, respectively. Thus, the hardware complexity of Fig.4.5(d) is $6\times(2\alpha+4\beta+2\gamma)+2\alpha = 14\alpha+24\beta+12\gamma$. In Lemma 5, if we put $n=3$, we get exactly the same value.

4.4 Proposed Reversible Fault Tolerant $4n$-to-n Multiplexer for FPGA

Sec.3.1 shows that the Plessey logic block of FPGA requires an 8-to-2 Mux. A Mux is a device that sends multiple signals on a carrier channel at the same time in the form of a single complex signal to another device that recovers the separate signals at the receiving end. In other words, Mux selects one of many analog or digital data sources and outputs that source into a single channel. Thus, Mux is also known as a data selector. This subsection elaborately describes the proposed algorithm, design and working procedure of $4n$-to-n **R**eversible **F**ault tolerant **Mux** (RFMux) for the FPGA. From now on, we denote a reversible fault tolerant Mux as RFMux. Considering the simplest case, $n = 1$, we have a 4-to-1 RFMux. Fig.4.8(a) shows the architecture of the proposed 4-to-1 RFMux for the FPGA. The corresponding timing diagram of the proposed 4-to-1 RFMux is shown in Fig.4.8(b).



(a)



(b)

Figure 4.8: Proposed 4-to-1 RFMux (a) Architecture (b) Timing diagram

From Fig.4.8(b), we find that, when both control signals are low, the final output O_{mux0} follows I_0 (from the start to $80ns$). However, when control signal $S_0 = 1$ and $S_1 = 0$, the final output O_{mux0} follows I_1 (80 to $160ns$), when control signal $S_0 = 0$ and $S_1 = 1$, the final output O_{mux0} follows I_2 (160 to $240ns$) and so on.

From Fig.4.8(a), we find that the 4-to-1 RFMux produces 5 garbage outputs, whereas no constant input is needed. These are the minimal garbage outputs and constant input for 4-to-1 RFMux which is proved below in Theorem 3 and Example 5. Table.4.7 shows comparison of the 4-to-1 proposed Mux and the existing Muxs [36, 37]. The bottom row of the table depicts the improvements in the proposed design in percentage with respect to existing designs. In this table and all the following tables, improvement ratio (IR) $= \frac{LVE-VPD}{LVE} \times 100\%$, where, LVE = **L**owest **V**alue between the **E**xisting designs and VPD = **V**alue of the **P**roposed **D**esign. Algorithm 2 represents the design procedure of the proposed $4n$-to-n RFMux. Initially, the algorithm builds a 4-to-1 RFMux circuit, then recursively builds a $4n$-to-n RFMux using the procedure of 4-to-1 RFMux.

Table 4.7: Comparison of 4-to-1 reversible Mux

	GT	GO	QC	UD	TR
Existing Design [36]	9	11	57	6	110
Existing Design [37]	7	11	57	7	110
Proposed Design	3	5	15	3	12
Improvement Ratio (IR)	>57%	>54%	>73%	≥50%	>89%

Theorem 3: A $4n$-to-n reversible Mux requires at least($4n$+1) garbage outputs and no constant inputs, where n is number of data bits.

Proof: Let n be the number of data bits. Then, for a $4n$-to-n Mux, there are $4n$ data inputs and (n+1) select inputs. So, to preserve the one-to-one mapping of reversibility, there should be at least ($5n$+1) outputs. Among these ($5n$+1) outputs, there are n primary outputs. Hence, there should be at least ($4n$+1) garbage outputs for $4n$-to-n reversible Mux, which causes no constant input. □

Algorithm 2: Algorithm for the proposed reversible fault tolerant $4n$-to-n multiplexer for FPGA, ***RFMux(I, S, F2G, FRG)***

```
Input  : Data input $I(I_0, I_1, ..., I_{4n-1})$, Control input $S(S_0, S_1, ..., S_n)$
         $n$ and Fredkin gate ($FRG$)
Output: $4n$-to-$n$ RFMux circuit
begin
    $i = input$, $o = output$, $j = 0$
    begin procedure (4-to-1 RFMux)
        $S_j \rightarrow first.i.FRG.1\&2$, $I_j \rightarrow second.i.FRG.1$
        $I_{j+2} \rightarrow second.i.FRG.2$ $I_{j+1} \rightarrow third.i.FRG.1$
        $I_{j+3} \rightarrow third.i.FRG.2$, $j++$, $S_j \rightarrow first.i.FRG.3$,
        $second.o.FRG.1 \rightarrow second.i.FRG.3$,
        $second.o.FRG.2 \rightarrow third.i.FRG.3$,
        return $second.o.FRG.3 \rightarrow desired\ output$
               $remaining.FRG.o \rightarrow garbage\ output.$
    end procedure
    for $i \leftarrow 0$ to $n-1$ do
        goto line 3
    end for
    return $4n$-to-$n$ RFMux circuit
end
```

Example 5: Let the value of n be 1. Then, we have 4-to-1 Mux. The 4-to-1 Mux has four data inputs (I_0, I_1, I_2 and I_3) and two control inputs (S_0 and S_1). So, the total number of inputs is six. To maintain one-to-one correspondence, the reversible 4-to-1 Mux must have at least six outputs. Among these six outputs, only one is the primary output. Thus, there are at least five outputs that remain unused, *i.e.*, garbage outputs. If we replace n with 1 in Theorem 3, we get five garbage output as well. From Fig.4.8(a), we also find that the proposed 4-to-1 RFMux has five garbage outputs. Next, since it is possible to realize the 4-to-1 Mux reversibly with total of six outputs, thus it will not require any constant input. Circuitry of Fig.4.8(a) also has no constant input. So, the proposed 4-to-1 RFMux (Fig.4.8(a)) is the optimum design of reversible Mux in terms of garbage outputs and constant input as it adapts Theorem 3. Because the 4-to-1 RFMux is used in $4n$-to-n RFMux as a main a procedure which is recursively called n times, thus the above optimality statement is relevant to any $4n$-to-n RFMux.

Lemma 6: Let n be number of data bits. Then $4n$-to-n RFMux for FPGA can be realized with $3n$ Fredkin gates.

Proof: An $4n$-to-n RFMux recursively called 4-to-1 RFMux n times. According to our design procedure, 4-to-1 RFMux requires 3 Fredkin gates. So, $4n$-to-n RF-Mux for FPGA can be realized with $3n$ Fredkin gates. □

Example 6: From Fig.4.8(a), we find that the proposed 4-to-1 reversible fault tolerant Mux requires three Fredkin gates. If we replace n with one in Lemma 6, we get exactly the same value, *i.e.*, three.

Lemma 7: An $4n$-to-n RFMux for FPGA can be realized with $15n$ quantum cost and $(6n\alpha + 12n\beta + 6n\gamma)$ hardware complexity, where n is the number of data bits.

Proof: In Lemma 6, we proved that the $4n$-to-n RFMux for FPGA can be realized with $3n$ Fredkin gates. Quantum cost and Hardware complexity of the FRG are 5 and $(2\alpha+4\beta+2\gamma)$, respectively. Therefore, the quantum cost of $4n$-to-n RFMux is $3n \times 5 = 15n$ and hardware complexity of $4n$-to-n RFMux is $3n \times (2\alpha + 4\beta + 2\gamma) = (15n\alpha + 12n\beta + 6n\gamma)$. □

Example 7: From Fig.4.8(a), we find that the proposed 4-to-1 reversible fault tolerant Mux requires three reversible fault tolerant Fredkin gates. According to our previous discussion in Sec.2, quantum cost and hardware complexity of a reversible fault tolerant Fredkin gate are 5 and $2\alpha + 4\beta + 2\gamma$, respectively. Thus, the quantum cost of Fig.4.8(a) is $3 \times 5 = 15$, whereas its hardware complexity is $3 \times (2\alpha + 4\beta + 2\gamma) = 6\alpha + 12\beta + 6\gamma$. In Lemma 7, if we put $n = 1$, the values we get are 15 and $6\alpha + 12\beta + 6\gamma$, respectively, which are identical to the quantum cost and hardware complexity of Fig.4.8(a).

4.5 Proposed Reversible Fault Tolerant Random Access Memory for FPGA

The **R**andom **A**ccess **M**emory (RAM) for the FPGA can be considered as an array of individual memory cells. More specifically, there are 2^n rows where each row contains m master-slave flip-flops. To address these rows and columns a decoder is needed. In the decoder, only one out of 2^n outputs is SET. The write bit W determines whether a read or a write operation is to be performed. The SET output of the decoder selects one row of the flip-flops. If W is SET, then m flip-flops of the selected row are written with the inputs D_0 to D_{m-1}. On the other hand, if W is LOW, Q_0 to Q_{m-1} holds the previously stored bits in the flip-flops of the selected row and are refreshed with the stored bits.

We have proposed a reversible RAM in [83]. This design consists of an n-to-2^n reversible decoder, Toffoli gates, write enable master-slave flip-flops, 3×3 and $2^n \times 2^n$ (2^n input bits) Feynman gates. To design the **R**eversible **F**ault tolerant **RAM** (RFRAM) for the FPGA, we extend this reversible design [83] to reversible fault tolerant design. The extension is straight forward, we transformed each of the reversible component [83] to reversible fault tolerant components with more optimality. From now on, we denote a reversible fault tolerant RAM of the FPGA as RFRAM. We have already projected a compact design of reversible fault tolerant write enable master-slave flip-flop in Sec.4.2. In addition to that, an optimal design of reversible fault tolerant decoder is presented in Sec.4.3 (Algorithm 1). We have also discussed how a reversible Ex-OR operation can be realized while preserving the parity in Sec.2.5. Now, replacing Feynman gate (FG) of Fig.4.9 with Feynman double gate ($F2G$), the task remains to realize the reversible Toffoli gate[3] so that it can preserve parity as well. Fig.4.10(a) represents the architecture of a **F**ault **T**olerant realization of reversible **T**offoli **G**ate, hence named as FTTG. The corresponding timing diagram of FTTG is shown in Fig.4.10(b).

[3]Toffoli gate means 3×3 reversible Toffoli gate unless mentioned explicitly.

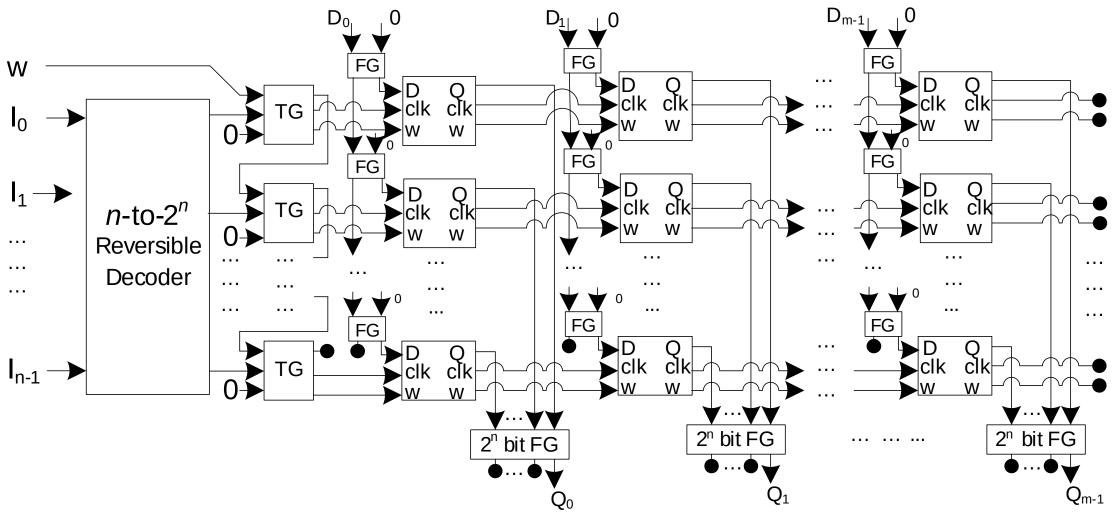
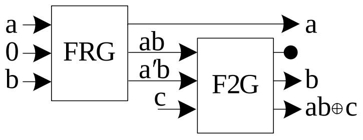
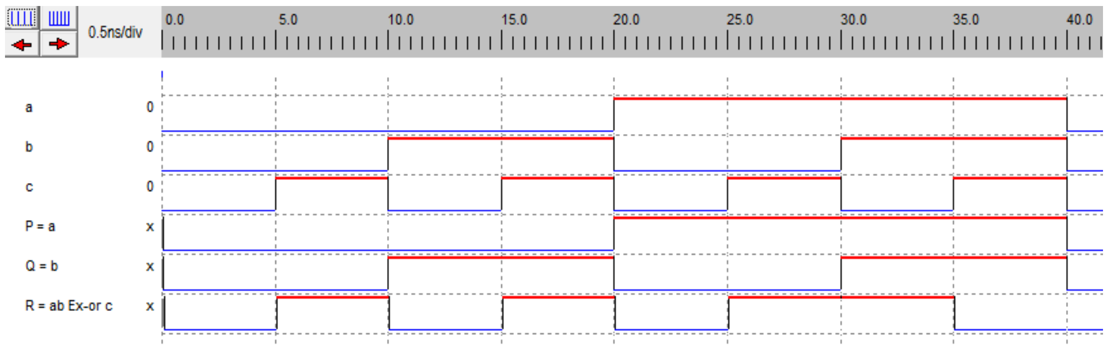


Figure 4.9: Architecture of our proposed reversible RAM [83]



(a)



(b)

Figure 4.10: Proposed fault tolerant realization of Toffoli gate (a) Architectural block diagram (b) Timing diagram

From Fig.4.10(a), we find that the fault tolerant realization of Toffoli gate requires a Fredkin gate (FRG) and a Feynman double gate ($F2G$). We also find that, this circuit produces one garbage output, requires one constant input, and its quantum cost is 7, since the quantum cost of a FRG is 5 and $F2G$ is 2. In the following theorems, we prove that these are the minimal number of garbage outputs, constant inputs and quantum cost if we want to realize the reversible Toffoli gate in parity preserving mode.

Theorem 4: The fault tolerant realization of reversible Toffoli gate requires at least one garbage output and one constant input.

Proof: Input-output combinations of reversible Toffoli gate don't preserve the parity. To preserve the parity at the output level, at-least one more output is needed, which demands at-least one constant input at the input level. Thus, fault tolerant realization of reversible toffoli gate requires at least one garbage output and one constant input. □

Example 8: Fig.4.10(a) is a straight forward proof for the existence of fault tolerant circuit which can realize the reversible Toffoli gate with one garbage output and one constant input. Next, we must demonstrate the nonexistence of the reversible fault tolerant circuit which can realize the reversible Toffoli gate without the garbage output and without the constant input. Input and output vectors for reversible Toffoli gate (TG) are: $I_v = (a, b, c)$ and O_v=$(P = a, Q = b, R = ab \oplus c)$. The corresponding truth table for these input-output combinations is shown in Table.4.8.

Table 4.8: Truth table of reversible Toffoli gate

Inputs			Outputs			
a	b	c	P	Q	R	Parity
0	0	0	0	0	0	$I_p = O_p$
0	0	1	0	0	1	$I_p = O_p$
0	1	0	0	1	0	$I_p = O_p$
0	1	1	0	1	1	$I_p = O_p$
1	0	0	1	0	0	$I_p = O_p$
1	0	1	1	0	1	$I_p = O_p$
1	1	0	1	1	1	I_p=E, O_p=O
1	1	1	1	1	0	I_p=O, O_p=E

From Table.4.8, we find that the input combinations 1-1-0 and 1-1-1 (last two rows) have different parity at the input and output level. This result establishes the nonexistence of the reversible fault tolerant circuit which can realize the Toffoli gate without the garbage output and without the constant input.

Theorem 5: A fault tolerant Toffoli gate can be realized with at least seven quantum cost.

Proof: Theorem 4 and Fig.2.4(b) are the proof for the nonexistence of reversible Toffoli gate with 5 quantum cost in parity preserving fashion. Here, it is also not possible to preserve the parity at the output level with six quantum cost. However, seven 2×2 primitive gates can realize reversible Toffoli gate while preserving parity. Thus, at least seven quantum cost are required to realize reversible fault tolerant Toffoli gate. □

Example 9: Fig.4.10(a) is a straight forward proof for the existence of fault tolerant circuit which can realize the reversible Toffoli gate by seven quantum cost. Next, we must demonstrate the nonexistence of the reversible fault tolerant circuit which can realize the reversible Toffoli gate with less than seven quantum cost. Quantum equivalent realization of reversible Tofoli gate consists of five 2×2 primitive gates as shown in Fig.2.4(b) (Sec.2.5). Example 8 shows that this realization doesn't preserve the parity. Like Theorem 2 and Example 2 (Sec.4.3), it can be shown that no matter how did we use one more primitive gate to this quantum equivalent realization of Fig.2.4(b) or six equivalent primitives gates in total, the corresponding circuit will not be able to maintain the input-output combinations or will violates the parity preserving property. More specifically, either it will violates the one-to-one correspondence of reversibility or fail to adapts with Eq.2.1, *i.e.*, not able to preserve the parity. This exemplifies the nonexistence claim stated above.

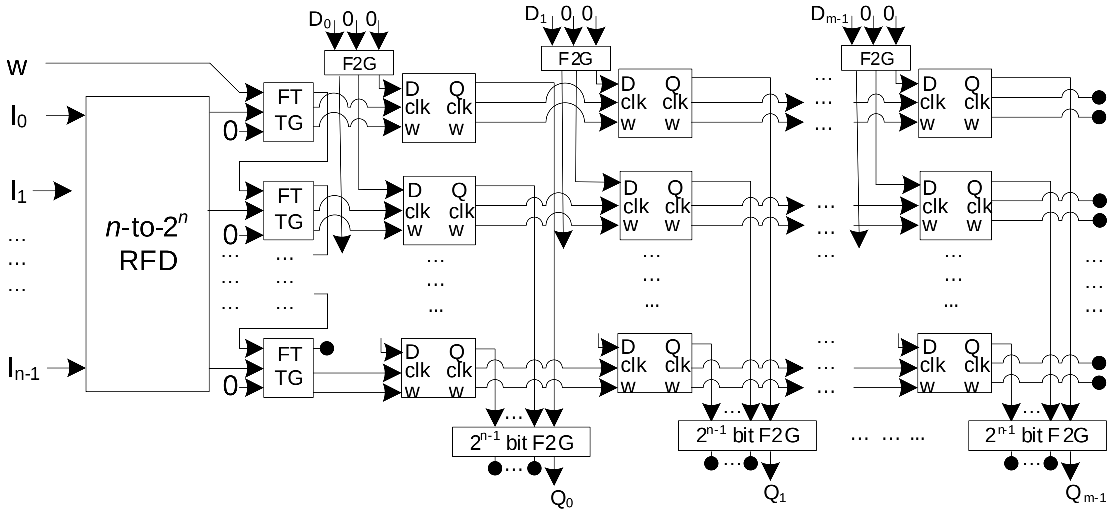


Figure 4.11: Architecture of the proposed RFRAM for FPGA

At this point, we have designed reversible fault tolerant circuitry of each component of the proposed RFRAM for the FPGA. Combining all these components the architectural block diagram of the proposed RFRAM is shown in Fig.4.11. Algorithm 3 demonstrates the design procedures of the proposed RFRAM. Primary inputs to the algorithm are data input sets $I(I_0, I_1, \ldots, I_{n-1})$, $D(D_0, D_1, \ldots,$ $D_{m-1})$ and write enable bit W. From the architectural block diagram of Fig.4.11, we find that the proposed RFRAM for FPGA demands an underneath n-to-2^n reversible fault tolerant decoder (RFD) circuitry. Thus, a call to RFD[4] is specified by line 3 in this proposed algorithm. Lines 4 to 13 assign inputs to fault tolerant Toffoli gates according to our design procedure of reversible fault tolerant random access memory for the FPGA. The inputs to $F2G$s and $RFMSFF$s are assigned by the lines 14 to 21. From Fig.4.11, we also find that there are m number of primary outputs which are returned by 2^{n-1} bit Feynman double gates. In Algorithm 3, these outputs are returned by the line 24 whereas line 26 finds all garbage outputs of RFRAM. Time complexity of Algorithm 3 depends on the For loops and there are at-most two levels of dependency among them, n then m or vice versa. Thus, complexity of this algorithm is $O(mn)$.

[4]Algorithm 1 ($RFD(I, F2G, FRG)$) presents detailed design procedure of n-to2^n RFD.

Algorithm 3: Algorithm for the proposed reversible fault tolerant random access memory for FPGA, ***RFRAM(I, D, ,W F2G, FRG)***

```
Input  : Data $I(I_0, I_1, ..., I_{n-1})$, $D(D_0, D_1, ..., D_{m-1})$,$W$,$FRG$, $F2G$
Output: reversible fault tolerant RAM circuit for FPGA
begin
    $i = input$, $o = output$
    call $RFD(I,\ F2G,\ FRG)$
    for $k \leftarrow 0$ to $n-1$ do
        if $k = 0$ then $W \rightarrow first.i_k.FTTG$ end if
        else $first.o_{k-1}.FTTG \rightarrow first.i_k.FTTG$ end if
        $RFD.o_k \rightarrow second.i_k.FTTG$, $0 \rightarrow third.i_k.FTTG$
    end for
    for $k \leftarrow 0$ to $m-1$ do
        $D_k \rightarrow first.i_k.F2G$, $0 \rightarrow second\ \&\ third.i_k.F2G$
        for $l \leftarrow 1$ to $n/2$ do
            $F2G.o_k \rightarrow first.i_l.F2G$, $0 \rightarrow second\ \&\ third.i_l.F2G$
        end for
        for $l \leftarrow 0$ to $n-1$ do
            $D_k(F2G.o_k l) \rightarrow first.RFMSFF.i$
            $x = second\ \&\ third$
                if $l = 0$ then $x.FTTG.o_l \rightarrow x.RFMSFF.i_l$ end if
                else $x.RFMSFF.o_{l-1} \rightarrow x.RFMSFF.i_l$ end if
            $first.RFMSFF.o.1 \rightarrow first.2^{n-1}bitF2G.i$
        end for
    end for
    return
    for $j \leftarrow 0$ to $m-1$ do
        $2^{n-1}bitF2G.o.2^{n-1} \rightarrow desired\ output$
    end for
    all remaining $F2G.o, FRG.o$ & $2^{n-1}bitF2G.o \rightarrow$ garbage output.
end
```

Lemma 8: A RFRAM for the FPGA is realized with $(2^n + 2n + \frac{15nm}{2} + m - 1)$ reversible fault tolerant gates, where both n,$m \in N$, the set of positive integers and $nm \geq 2$.

Proof: According to our design procedure, RFRAM for FPGA consists of an n-to-2^n RFD, n number of FTTG, $\frac{nm}{2}$ number of 3×3 $F2G$, where both n and m belongs to the set of positive integers and greater than or equal to 2. In addition to that, nm number of RFMSFF and m number of $2^{n-1} \times 2^{n-1}$ Feynman double gates are also required. In Lemma 4 (Sec.4.3), it is proved that the n-to-2^n RFD is realized with (2^n-1) reversible fault tolerant gates. As shown earlier, each FTTG requires two reversible fault tolerant gates. Lemma 2 proved that, each RFMSFF requires 7 reversible fault tolerant gates. Hence, number of gates required for a RFRAM for the FPGA is

$$\begin{aligned} & (2^n - 1) + \sum_{0}^{n-1} 2 + \frac{nm}{2} + 7nm + m \\ = \; & 2^n + 2n + \frac{15nm}{2} + m - 1 \end{aligned}$$

□

Lemma 9: A RFRAM for FPGA is realized with $(2n+(m(2^{n-2}+1+7n))$ garbage outputs, where both n,$m \,\epsilon\, N$, the set of positive integers and $nm \geq 2$.

Proof: Lemma 8 proved that the RFRAM of FPGA consists of an n-to-2^n RFD, n number of FTTG, and $\frac{nm}{2}$ number of 3×3 $F2G$, nm number of RFMSFF and m number of $(2^{n-1} \times 2^{n-1})$ $F2G$. In Theorem 1 (Sec.4.3), it is proved that the n-to-2^n RFD is realized with at least n garbage outputs. Theorem 4 proved that, each FTTG requires at least one garbage output. According to our design procedures, all remaining Feynman double gates combinedly produce $(m(2^{n-2} + 1))$ garbage outputs. In addition to that, each RFMSFF produces seven garbage outputs as shown in Lemma 2. Hence, total garbage outputs produced by the RFRAM for the FPGA is

$$\begin{aligned} & n + \sum_{0}^{n-1} 1 + \sum_{0}^{n-1} \sum_{0}^{m-1} 7 + m(2^{n-2} + 1) \\ = \; & n + n + 7mn + 2^{n-2}m + m \\ = \; & 2n + m(2^{n-2} + 1 + 7n) \end{aligned}$$

□

Lemma 10: A RFRAM for FPGA can be realized with $(2^n + 9nm + n)$ constant inputs, where both n,$m \,\epsilon\, N$, the set of positive integers and $nm \geq 2$.

Proof: Previous two lemmas proved that the RFRAM for FPGA consists of an n-to-2^n RFD, n number of FTTG, $\frac{nm}{2}$ number of 3×3 Feynman double gates. In addition to that, nm number of RFMSFF and m number of $(2^{n-1} \times 2^{n-1})$ Feynman double gates is also required. In Theorem 1 (Sec.4.3), it is proved that, for an n-to-2^n RFD at least 2^n number of constant inputs are required. On the other hand, Theorem 4 proved that the fault tolerant realization of each reversible Toffoli gate (FTTG) requires at least one constant input. According to our design procedure, each 3×3 $F2G$ requires 2 constant inputs. However, no constant input is required for $2^{n-1} \times 2^{n-1}$ Feynman double gates. Also, each RFMSFF requires 8 constant inputs as shown earlier. Hence, the total number of constant inputs for the RFRAM of FPGA is

$$\begin{aligned} & 2^n + \sum_{0}^{n-1} 1 + \frac{nm}{2} \times 2 + \sum_{0}^{n-1} \sum_{0}^{m-1} 8 \\ = \; & 2^n + 9nm + n \end{aligned}$$

□

Lemma 11: A reversible fault tolerant RAM for FPGA can be realized with $(2^n(\frac{m}{4} + 5) + 24mn + 7n - 8)$ quantum cost, where both n,$m \,\epsilon\, N$, the set of positive integers and $nm \geq 2$.

Proof: Lemma 8 proved that the reversible fault tolerant RAM for FPGA can be realized with $(2^n + 2n + \frac{15nm}{2} + m - 1)$ reversible fault tolerant gates. Among these gates, there are m number of 2^{n-1} bit (input) Feynman double gates, $(2^n + 3mn + n - 2)$ number of 3×3 Fredkin gates and the rest are 3×3 Feynman double gates, where n and m are the number of rows and columns of the RAM's structure; both n,$m \,\epsilon\, N$, the set of positive integers and $nm \geq 2$. According to our previous discussions, quantum cost of each 3×3 Fredkin and Feynman double gates are 2 and 5, respectively. Besides this, quantum cost of each 2^{n-1} bit Feynman double gate is 2^{n-2}. Thus, the total quantum cost of RFRAM is

$$\begin{aligned} & 2^{n-2}m + 5(2^n + 3mn + n - 2) + 2(n + 1 + \frac{9nm}{2}) \\ = \; & 2^{n-2}m + (2^n \times 5) + 15mn + 5n - 10) + 2n + 2 + 9nm \\ = \; & 2^n(\frac{m}{4} + 5) + 24mn + 7n - 8 \end{aligned}$$

□

Lemma 12: Let α, β and γ be the hardware complexity of two-input Ex-OR, AND and NOT calculations, respectively. Then, the total hardware complexity of RFRAM for FPGA is $(2^{n-2}(m+8)+15mn+4n-2)\alpha+(2^{n+1}+6mn+2n-4)(2\beta+\gamma)$, where both n,$m \,\epsilon\, N$, the set of positive integers and $nm \geq 2$.

Proof: Lemma 8 proved that the reversible fault tolerant RAM for FPGA can be realized with $(2^n + 2n + \frac{15nm}{2} + m - 1)$ reversible fault tolerant gates. Among these gates, there are m number of 2^{n-1} bit (input) Feynman double gates, $(2^n + 3mn + n - 2)$ number of 3×3 Fredkin gates and the rest are 3×3 Feynman double gates, where n and m are the number of rows and columns of the RAM's structure; both n,$m \,\epsilon\, N$, the set of positive integers and $nm \geq 2$. According to our previous discussions, of each 3×3 Feynman double and Fredkin gates are 2α and $2\alpha + 4\beta + 2\gamma$, respectively. Besides this, the hardware complexity of each 2^{n-1} bit Feynman double gate is $2^{n-2}\alpha$. Thus, the total hardware complexity of RFRAM for FPGA is

$$\begin{aligned} & m \times 2^{n-2}\alpha + (2^n + 3mn + n - 2)(2\alpha + 4\beta + 2\gamma) + (n + 1 + \frac{9nm}{2})2\alpha + \\ = \; & (2^{n-2}(m + 8) + 15mn + 4n - 2)\alpha + (2^{n+1} + 6mn + 2n - 4)(2\beta + \gamma) \end{aligned}$$

□

4.6 Proposed Reversible Fault Tolerant Plessey Logic Block for FPGA

The logic blocks of FPGA are connected through programmable interconnects or wire segments. Wire segment consists of programmable switches. Generally, One or two switches are attached in a segment and each end of a segment has a switch attached to it. A routing channel is a group of parallel tracks. The track consists of one or more segment(s) arranged in a sequence which is shown earlier in Figs.3.1(a)∼(c) (Sec.3.1).

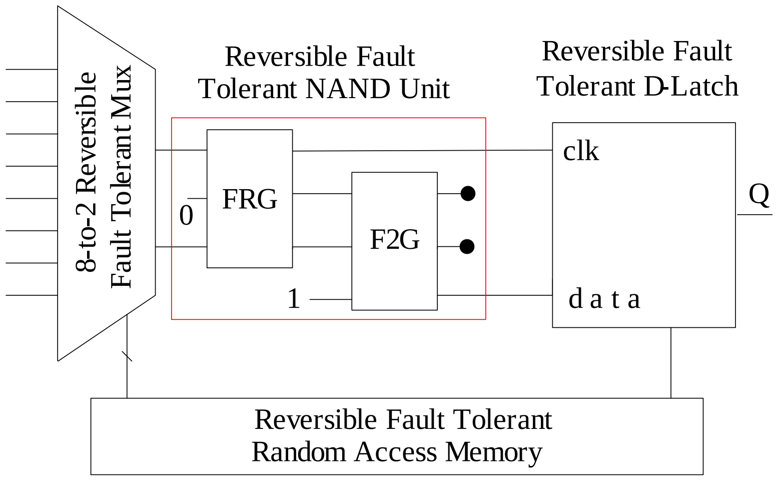

8-to-2 Reversible Fault Tolerant Mux
Reversible Fault Tolerant NAND Unit
Reversible Fault Tolerant D-Latch
FRG
F2G
0
1
clk
data
Q
Reversible Fault Tolerant Random Access Memory

Figure 4.12: Proposed Reversible Fault Tolerant Plessey logic block for FPGA

Fig.4.12 shows the block diagram of the proposed reversible fault tolerant Plessey logic block of static RAM based FPGA. This architecture consists of a reversible fault tolerant D-latch (FTD), 8-to-2 multiplexer (RFMux), random access memory (RFRAM) and an extra Fredkin and Feynman double gates. The FTD, RFMux and RFRAM have already been proposed in Secs.4.1, 4.4 and 4.5, respectively. The extra Fredkin and Feynman double gates are needed to implement the NAND unit of Plessey logic block. First input to the Fredkin gate of this NAND unit comes from first 4-to-1 RFMux where third input to the Fredkin gate of this NAND unit comes from second 4-to-1 RFMux, *i.e.*, inputs to this NAND unit are simply the outputs of the 8-to-2 RFMux. Inputs of each RFMux are either connected to the output of the previous NAND unit in the row or the output of the NAND unit below or above this block (to the closer one). From Fig.4.12, we find that the proposed NAND unit of Plessey logic block produces two garbage outputs. This is the optimum number of garbage outputs for the two-input NAND unit as proved in the following theorem.

Theorem 6: At least two garbage outputs are required to realize a two-input NAND operation in reversible and fault tolerant mode.

Proof: In a two-input (a, b) NAND operation, there is one primary output $(ab)'$, thus it has at least one garbage output. These output combinations don't preserve the one-to-one mapping of reversibility with respect to the input combinations, hence an additional garbage output is required, which also preserves parity. Thus, a reversible fault tolerant two-input NAND operations is realized with at least two garbage outputs. □

Example 10: Let the inputs are (a, b) for the two-input NAND operation. Then, the output of two-input NAND operation is $(ab)'$. According to the property of reversibility, there should be equal number of inputs and outputs, otherwise it will never be able to maintain the one-to-one correspondence. Therefore, there should be al least one more output. Let this output be q. This $(a, b) \leftrightarrow ((ab)', q)$ combinations have one garbage output namely q. The corresponding truth table of this input-output combinations is shown Table.4.9.

Table 4.9: Two input NAND operation with one garbage

a	b	$(ab)'$	q
0	0	1	
0	1	1	
1	0	1	
1	1	0	

According to the Table 4.9, it is clear that there are three identical outputs at the output level at $(ab)'$ (first three cells) for three different input combinations, which result's whatever the value of q, the output combinations of Table.4.9 don't able to maintain the one-to-one mapping of reversibility with respect to the input combinations. Hence, an additional garbage output is required. Thus, a reversible two input NAND operation requires at least two garbage outputs. Next, we must prove this two input NAND operation with two garbage outputs maintain the fault tolerant property of Eq.2.1, *i.e.*, it can preserve the parity for all input-output combinations. Now, we have one primary output and two garbage outputs, *i.e.*, total of three outputs but two inputs. So, there should be at least one constant input. Let the later garbage output be r and constant input (CI) be 1. Table.4.10 shows how this input-output combinations preserve the parity.

Table 4.10: Two input NAND operation with two garbage

Inputs			Outputs			
a	b	CI	$(ab)'$	q	r	Parity
0	0	1	1	0	0	$I_p = O_p = O$
0	1	1	1	1	0	$I_p = O_p = E$
1	0	1	1	0	1	$I_p = O_p = E$
1	1	1	0	1	0	$I_p = O_p = O$

Algorithm 4 presents a formal representation of the proposed reversible fault tolerant Plessey logic block for FPGA. From Fig.4.12, we find that, fault tolerant Plessey logic block considerably depends on RFRAM, thus a call to RFRAM is specified in line 3. The outputs of RFRAM are the selection bits for 8-to-2 RFMux. Thus, it calls $4n$-to$-n$ RFMux with n=2^5, which is defined in line 4 of Algorithm 4. On the other hand, outputs of RFMux are the inputs to the NAND unit (shown in Fig.4.12). Line 5 of the algorithm assigns input to the reversible fault tolerant NAND unit and D-latch of Plessey logic block for FPGA. Line 6 returns the primary output Q which actually is the input to the next logic block. Finally, line 7 returns all the garbage outputs of a reversible fault tolerant Plessey

[5] Algorithm 2 ($RFMux(I,\ S,\ F2G,\ FRG)$) presents detailed design procedure of $4n$-to-n RFMux.

Algorithm 4: Algorithm for the proposed reversible fault tolerant Plessey logic block of the FPGA

```
Input : Data input sets $I, S, D$, Write enable bit $(W)$
        Fredkin gate $(FRG)$ and Feynman double gate $(F2G)$
Output: Reversible fault tolerant Plessey logic block's circuit
1 begin
2     $i = input$, $o = output$
3     call $RFRAM(I, D, W, F2G, FRG)$
4     call $RFMux(I, S, 2, FRG)$; where $S \leftarrow RFRAM.o$
5     $RFMux.o.1 \rightarrow FRG.i.1$, $0 \rightarrow FRG.i.2$, $RFMux.o.2 \rightarrow FRG.i.3$;
      $FRG.o.2 \rightarrow F2G.i.1$, $FRG.o.3 \rightarrow F2G.i.2$, $1 \rightarrow F2G.i.3$;
      $FRG.o.1 \rightarrow D-Latch.i.1$, $F2G.o.3 \rightarrow D-Latch.i.2$
6     return $D-Latch.o \rightarrow Q$ (input for next logic block)
7            remaining $F2G.o$ & $FRG.o \rightarrow$ garbage output.
8     goto line 2 if this is not the last logic block
9 end
```

logic block for FPGA. As mentioned in Secs.1 and 3, there are clusters of logic blocks in the FPGA, but the above procedure is capable to build a single reversible fault tolerant Plessey logic, thus, to iterate the entire procedure till the last logic block, a recursive call has been specified in line 8 of this algorithm.

Lemma 13: Let $\boldsymbol{GT}$ and gt_{RAM} be the required numbers of gates for a reversible fault tolerant Plessey logic block and RFRAM of the FPGA, respectively. Then,

$$\boldsymbol{GT} \geq 10 + gt_{RAM}$$

Proof: A reversible fault tolerant Plessey logic block of the FPGA consists of a reversible fault tolerant D-latch, a two-input NAND unit, a 8-to-2 multiplexer (RFMux), and a random access memory(RFRAM). Let gt_{DL}, gt_{NAND} and gt_{Mux} be the numbers of gates for the fault tolerant D-latch, two-input NAND unit, 8-to-2 RFMux, respectively. Then, the number of gates for the reversible fault tolerant Plessey logic block, $\boldsymbol{GT} \geq (gt_{DL} + gt_{NAND} + gt_{Mux} + gt_{RAM})$. In Lemma 1 (Sec.4.1), it has been proved that the fault tolerant D-latch requires a reversible fault tolerant Fredkin and a Feynman double gates. The NAND unit of reversible fault tolerant Plessey logic block is also realized with a Fredkin and a Feynman double gates as shown in Fig.4.12. According to the design procedures of Algorithm 2 (Sec.4.4), a 8-to-2 RFMux requires total of six reversible fault tolerant gates. Therefore,

$$\begin{aligned} &\boldsymbol{GT} \geq 2 + 2 + 6 + gt_{RAM} \\ \Rightarrow \quad &\boldsymbol{GT} \geq 10 + gt_{RAM} \end{aligned}$$

□

Lemma 14: Let $\boldsymbol{GO}$ and go_{RAM} be the numbers of garbage outputs produced by a reversible fault tolerant Plessey logic block and RFRAM of the FPGA, respectively. Then,

$$\boldsymbol{GO} \geq 13 + go_{RAM}$$

Proof: As shown in Lemma 13, the reversible fault tolerant Plessey logic block of the FPGA consists of a reversible fault tolerant D-latch, a NAND unit, a 8-to-2 RFMux and a RFRAM. Lemma 1 (Sec.4.1) proved that the fault tolerant D-latch produces two garbage outputs. NAND unit of reversible fault tolerant Plessey logic block also produces two garbage outputs as shown in Fig.4.12. Theorem 3 (Sec.4.4) proved that the 8-to-2 RFMux produces nine garbage outputs. Therefore, the number of garbage outputs ($\boldsymbol{GO}$) for the reversible fault tolerant Plessey logic block of the FPGA is

$$\begin{aligned} &\boldsymbol{GO} \geq 2 + 2 + 9 + go_{RAM} \\ \Rightarrow \quad &\boldsymbol{GO} \geq 13 + go_{RAM} \end{aligned}$$

□

Lemma 15: Let $\boldsymbol{CI}$ and ci_{RAM} be the required numbers of constant inputs for the reversible fault tolerant Plessey logic block and RFRAM of the FPGA, respectively. Then,

$$\boldsymbol{CI} \geq 4 + ci_{RAM}$$

Proof: Lemma 13 proved that reversible fault tolerant Plessey logic block of the FPGA consists of a reversible fault tolerant D-latch, a reversible fault tolerant NAND unit, a 8-to-2 RFMux, and a RFRAM. In Sec.4.1, we proved that the fault tolerant D-latch requires two constant inputs. The NAND unit of reversible fault tolerant Plessey logic block of the FPGA also requires two constant inputs as shown above. However, no constant input is required for the 8-to-2 RFMux according to Theorem 3 (Sec.4.4). Thus, the required number of constant inputs ($\boldsymbol{CI}$) for the reversible fault tolerant Plessey logic block is

$$\begin{aligned} & \boldsymbol{CI} \geq 2 + 2 + ci_{RAM} \\ \Rightarrow \quad & \boldsymbol{CI} \geq 4 + ci_{RAM} \end{aligned}$$

□

Lemma 16: Let $\boldsymbol{QC}$ and qc_{RAM} be the quantum cost for a reversible fault tolerant Plessey logic block and RFRAM of the FPGA, respectively. Then,

$$\boldsymbol{QC} \geq 44 + qc_{RAM}$$

Proof: Lemma 13 proved that a reversible fault tolerant Plessey logic block of the FPGA consists of $(10 + gt_{RAM})$ reversible fault tolerant gates, where gt_{RAM} is the required number of gates for RFRAM. Among the remaining 10 gates, there are two 3×3 Feynman double gates while the rest are 3×3 Fredkin gates. According to our previous discussions in Secs.2.2 and 2.5, the quantum cost of a 3×3 Fredkin gate and Feynman double gates are 2 and 5, respectively . Therefore, the quantum cost ($\boldsymbol{QC}$) for the reversible fault tolerant Plessey logic block is

$$\begin{aligned} & \boldsymbol{QC} \geq (2 \times 2) + (8 \times 5) + qc_{RAM} \\ \Rightarrow\quad & \boldsymbol{QC} \geq 4 + 40 + qc_{RAM} \\ \Rightarrow\quad & \boldsymbol{QC} \geq 44 + qc_{RAM} \end{aligned}$$

□

Lemma 17: Let $\boldsymbol{HC}$ and hc_{RAM} be the hardware complexities of a reversible fault tolerant Plessey logic block and RFRAM of the FPGA, respectively. Let α, β and γ be the hardware complexity of two-input Ex-OR, AND and NOT calculations, respectively. Then,

$$\boldsymbol{HC} \geq 20\alpha + 32\beta + 16\gamma + hc_{RAM}$$

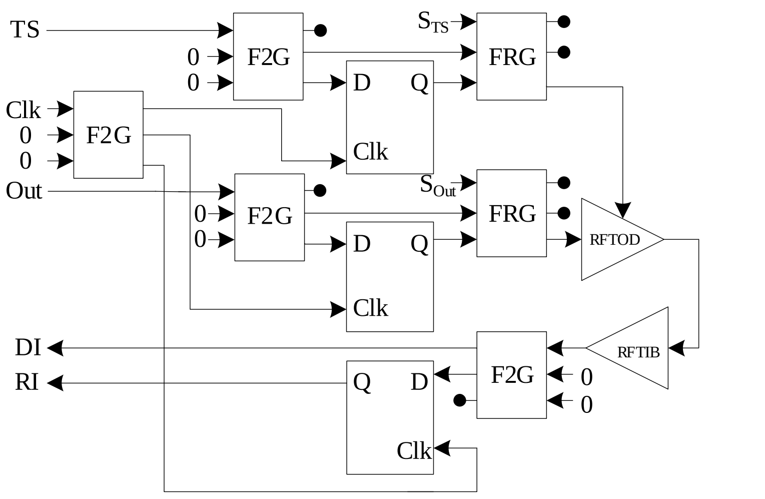
Proof: Lemma 13 proved that a reversible fault tolerant Plessey logic block of the FPGA consists of $(10 + gt_{RAM})$ reversible fault tolerant gates, where gt_{RAM} is the required number of gates for RFRAM. Among the remaining 10 gates, there are two 3×3 Feynman double gates while the rest are 3×3 Fredkin gates. According to our previous discussions, the hardware complexity of a 3×3 Feynman double gate is 2α, whereas the hardware complexity of 3×3 Fredkin gate is $2\alpha + 4\beta + 2\gamma$. Therefore, the hardware complexity ($\boldsymbol{HC}$) for the reversible fault tolerant Plessey logic block of the FPGA is

$$\begin{aligned} & \boldsymbol{HC} \geq (2 \times 2\alpha) + 8(2\alpha + 4\beta + 2\gamma) + hc_{RAM} \\ \Rightarrow\quad & \boldsymbol{HC} \geq 4\alpha + 16\alpha + 32\beta + 16\gamma + hc_{RAM} \\ \Rightarrow\quad & \boldsymbol{HC} \geq 20\alpha + 32\beta + 16\gamma + hc_{RAM} \end{aligned}$$

□

4.7 Proposed Reversible Fault Tolerant Input-Output Block for FPGA

There are several standards for input-output block in recent FPGAs, however it is unlikely that all standards are supported under a single FPGA structure. Basic input-output block from Xilinx is one of the most popular as it maintains a hierarchical structures while providing interface between the package pins and control blocks. Moreover, it can work in uni- or bi-directional modes. Outputs can be forced into high impedance[6] inputs. Fig.4.13 shows the block diagram of the proposed reversible fault tolerant input-output block from Xilinx FPGA without the SET/RESET and flip flop enable options. The explanations to exclude these options have already been presented in Sec.3.1. From Fig.4.13, we find that in the proposed input-output block, there are two FRGs and each one of them produces two garbage outputs. Here, each FRG works as 2-to-1 Mux. The following theorem proves that, these are the minimal number of garbage outputs of a 2-to-1 reversible fault tolerant Mux.



TS : Three State
Clk : Clock
Out : Output
DI : Direct Input
RI : Registered Input
RFTOD : Reversible Fault Tolerant Output Driver
RFTIB : Reversible Fault Tolerant Input Buffer
S_{TS} : Three State Selector Signal
S_{Out} : Output Selector Signal

Figure 4.13: Proposed Reversible Fault Tolerant Basic input-output Block for FPGA

[6]Impedance is the measurement of the opposite flow of an alternating current in a circuit due to resistance and reactance.

Theorem 7: A 2-to-1 reversible fault tolerant Mux requires at least two garbage outputs.

Proof: In a 2-to-1, Mux there are two primary inputs and one selector input. Thus, according to the property of reversibility, there should be at least three outputs. Among these three outputs, only one is the primary output. This input-output combination preserves parity too. Thus, a reversible fault tolerant 2-to-1 Mux is realized with at least two garbage outputs. □

From Fig.4.13, we find that there are three $F2G$s and three D flip-flops. The D flip-flop functions quite differently than the D-latch. However, both are identical without the options of SET/RESET and enables of flip-flop. Thus, in Fig.4.13, proposed reversible fault tolerant D-latch (Sec.4.1) has been used. In the proposed architecture of input-output block, a **R**eversible **F**ault **T**olerant **I**nput **B**uffer (RFTIB) and a **R**eversible **F**ault **T**olerant **O**utput **D**river (RFTOD) have also been used. Table.4.11 presents the performance of the proposed input-output block for the FPGA with the existing one [37]. In this table, the improvement ratio (IR) is calculated according to Eq.4.1 (Sec.4.4). In Chap.3, it is evidenced that the attempts towards the reversible input-output block for the FPGA have not been completed [37]. However, Table.4.11 evidenced that the proposed design is not only complete and but also performs much better than its existing counter-part. Moreover, the proposed input-output block can detect the fault signal in its primary outputs, which is not at-all present in the existing design.

Table 4.11: Comparison of reversible input-output block

	GT	GO	QC	CI	HC	UD	TR
Existing Design [37]	20	20	64	17	$34\alpha + 40\beta + 20\gamma$	6	124
Proposed Design	12	13	39	14	$24\alpha + 20\beta + 10\gamma$	4	104
Improvement Ratio (IR)	40%	35%	≥39%	>17%	>29%	>33%	>16%

Lemma 18: A reversible fault tolerant input-output block of the FPGA can be realized with twelve gates.

Proof: A reversible fault tolerant input-output block consists of four $F2G$, two FRG and three reversible fault tolerant D flip-flops. According to our design procedure, each D flip-flop consists of two gates. Thus, the number of gates for a reversible fault tolerant input-output block of the FPGA is $(4+2+(3\times 2)) = 12$.□

Lemma 19: A reversible fault tolerant input-output block of the FPGA can be realized with thirteen garbage outputs.

Proof: A reversible fault tolerant input-output block consists of four $F2G$, two FRG and three reversible fault tolerant D flip-flops. According to our design procedure, each D flip-flop and FRG produce two garbage outputs. In addition to that, all remaining $F2G$s combinedly produce three garbage outputs. Thus, the number of garbage outputs produced by a reversible fault tolerant input-output block of the FPGA is $((2\times2)+(3\times2)+3) = 13$. □

Lemma 20: A reversible fault tolerant input-output block of the FPGA can be realized with fourteen constant inputs.

Proof: A reversible fault tolerant input-output block consists of four $F2G$, two FRG and three reversible fault tolerant D flip-flops. According to our design procedure, each D flip-flop and $F2G$ requires two constant inputs. However, no constant input is needed for the FRGs. Thus, the number of constant inputs for a reversible fault tolerant input-output block of the FPGA is $((4\times2)+(3\times2)) = 14$. □

Lemma 21: The quantum cost for a reversible fault tolerant input-output block of a reversible fault tolerant FPGA is 39.

Proof: Lemma 18 proved that a reversible fault tolerant input-output block consists of 12 reversible fault tolerant gates, among which 7 are $F2G$ and the rest are FRGs. According to our previous discussions, quantum cost of each FRG and $F2G$ are 5 and 2, respectively. Thus, total quantum cost of a reversible fault tolerant input-output block of the FPGA is $((5 \times 5) + (2 \times 7)) = 39$. $\square$

Lemma 22: Let $\boldsymbol{H}$ be the hardware complexity of reversible fault tolerant input-output block of the FPGA. Let α, β and γ be the hardware complexity of two-input Ex-OR, AND and NOT calculations, respectively. Then,

$$\boldsymbol{H} = 24\alpha + 20\beta + 10\gamma$$

Proof: Lemma 18 proved that a reversible fault tolerant input-output block of the FPGA consists of 12 reversible fault tolerant gates among which 5 are FRGs and the rest are $F2G$s. According to our previous discussions, the hardware complexity of a $F2G$ is 2α, whereas the hardware complexity of FRG is $(2\alpha + 4\beta + 2\gamma)$. Therefore,$\boldsymbol{H} = 7\times(2\alpha)+5\times(2\alpha+4\beta+2\gamma) \Rightarrow \boldsymbol{H} = 24\alpha+20\beta+10\gamma$ $\square$

4.8 Summary

This chapter detailed the design and working procedures of the proposed reversible fault tolerant Plessey logic block and basic input-output block for the fault tolerant FPGA. Here, several lower bounds on the number of garbage outputs, constant inputs and quantum cost of fault tolerant FPGA have also been proposed. It has also been evidenced that the proposed components are optimized greatly from the existing components.

Chapter 5

Performance Evaluation of the Proposed Method

In the previous chapter, the performance of the components of reversible fault tolerant FPGA and the existing components of reversible FPGA had been shown. This chapter provides the overall performances of the proposed reversible fault tolerant scheme with the existing reversible methods. The performance of the proposed method is evaluated by the required number of gates, garbage outputs, constant inputs, quantum cost, unit delay and hardware complexity. As discussed in Chap.4, there are two variables n and m that fix the structure of the FPGA. Therefore, to detect the scalability of the proposed method, the overall comparative results are shown in contrast with three possible design approaches:

(i) Varying m while keeping n constant

(ii) Varying n while keeping m constant

(iii) Varying both m and n

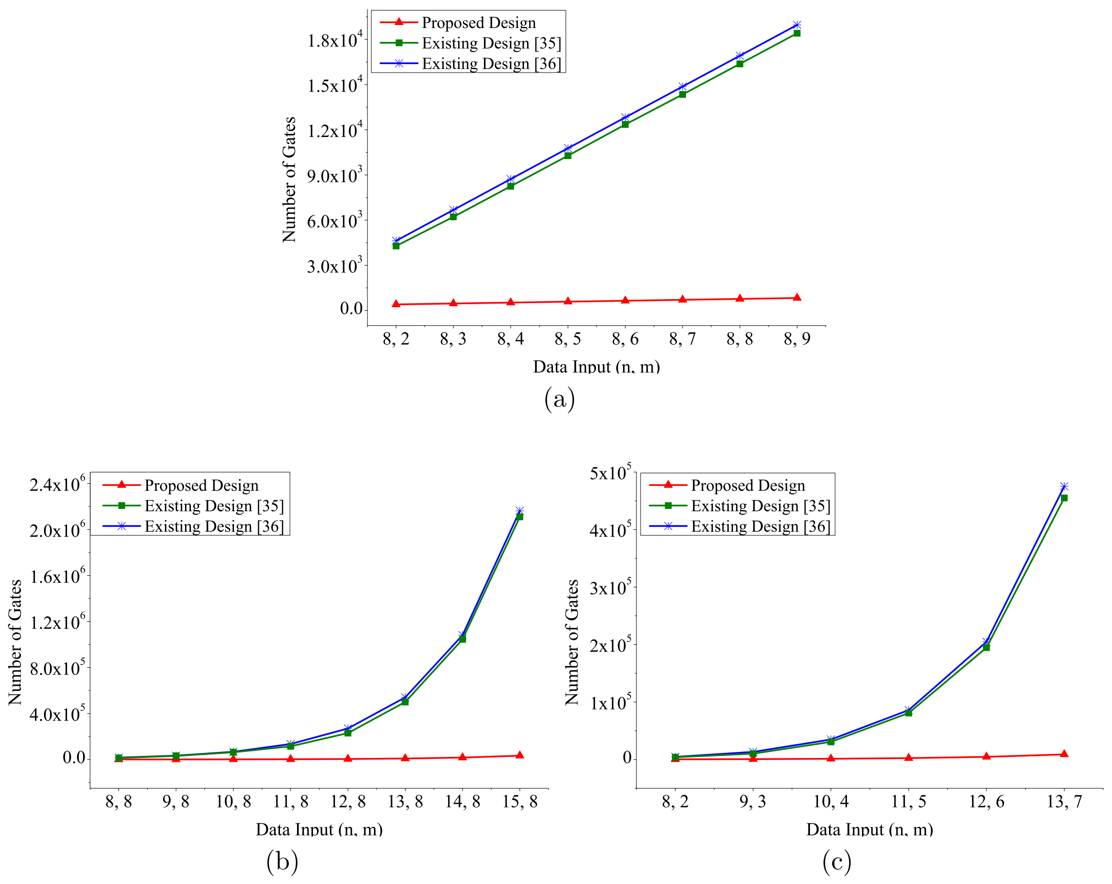
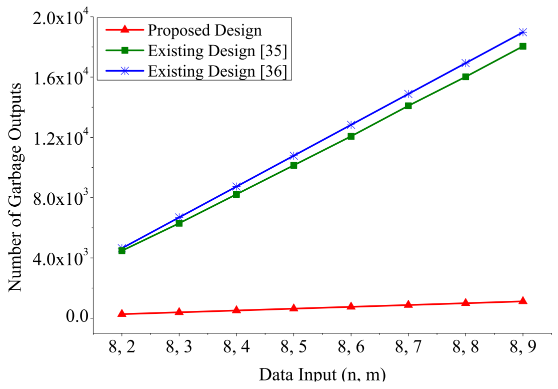


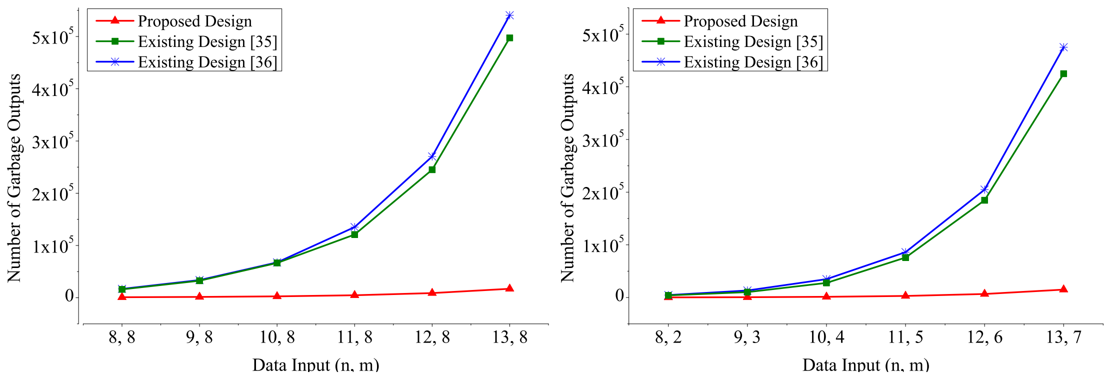
Figure 5.1: Performance of the proposed work and existing works [35, 36] with respect to the number of gates (a) Varying m while keeping the constant n (b) Varying n while keeping the constant m (c) Varying both m and n.

5.1 Gate Comparison

The number of pins and area of the chip increases with increasing number of gates. In case of FPGA, this will create substantial difficulties for the programmable routing networks. Fig.5.1 represents the performance of the proposed reversible fault tolerant FPGA with respect to the existing reversible FPGAs in terms of number of gates. Figs.5.1(a), (b) and (c) represent the above mentioned (i), (ii) and (iii) design approaches, respectively. The plotted graphs of Fig.5.1 show that the proposed scheme outperforms the existing works. Most interestingly, for the last two cases, the rate of changes with respect to number of gates in existing designs are exponential, whereas the proposed design changes linearly.



(a)

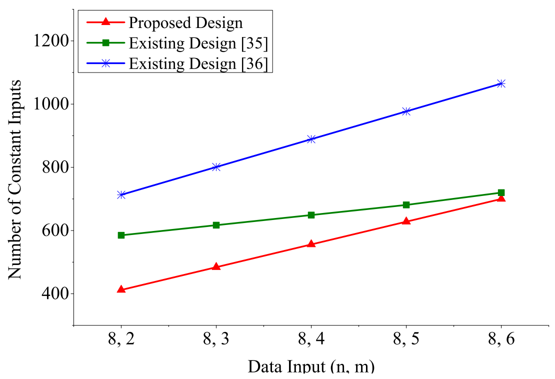


(b) (c)

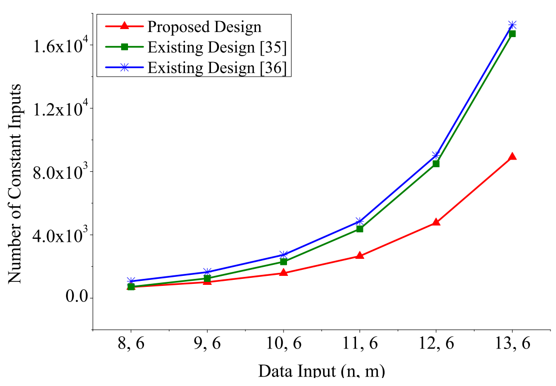
Figure 5.2: Performance of the proposed work and existing works [35, 36] with respect to the garbage outputs (a) Varying m while keeping the constant n (b) Varying n while keeping the constant m (c) Varying both m and n.

5.2 Garbage Outputs Comparison

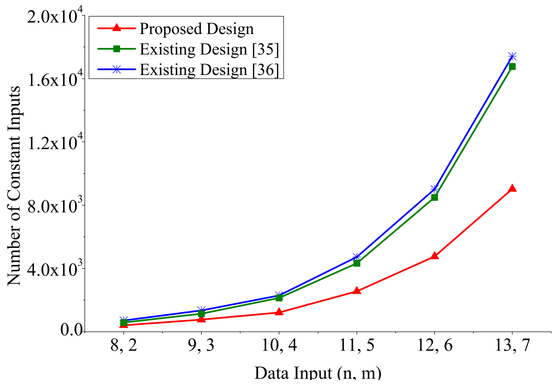
The garbage outputs represent the number of outputs that do not perform any useful operation. Thus, it is considered as an overhead. Researchers have showed that, heavy price is paid off for each garbage output. Fig.5.2 represents the performance of the proposed FPGA with respect to the existing FPGAs in terms of garbage outputs. With respect to the garbage outputs produced, the above mentioned (i), (ii) and (iii) design approaches are shown in Figs.5.2(a), (b) and (c), respectively. Fig.5.2 show that the proposed design performs much better than than existing works for all possible circumstances and for the last two cases existing designs increases exponentially, whereas the proposed design increases linearly.



(a)



(b)

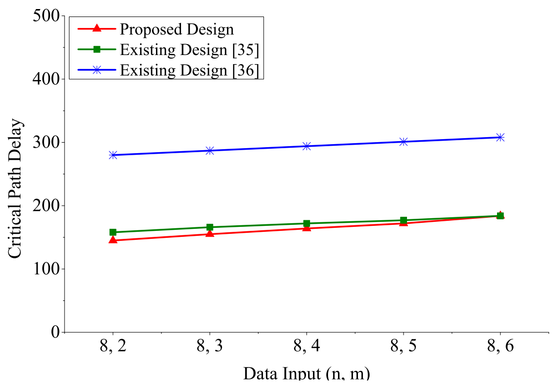


(c)

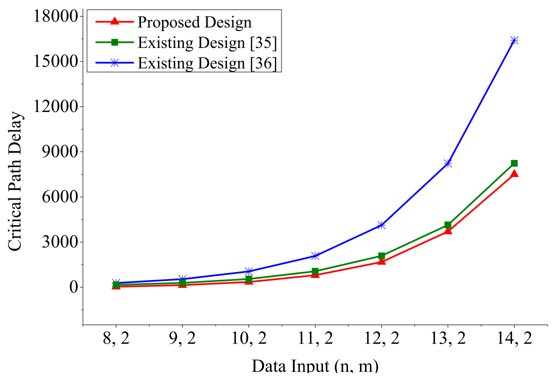
Figure 5.3: Performance of the proposed work and existing works [35, 36] with respect to the constant inputs (a) Varying m while keeping the constant n (b) Varying n while keeping the constant m (c) Varying both m and n.

5.3 Constant Inputs Comparison

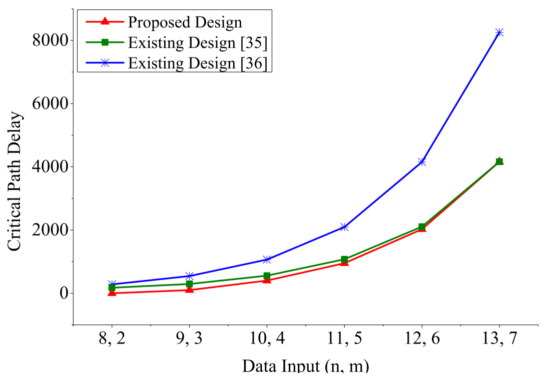
The quantum circuit with many constant bits which have the reversible gates as a building block are unmanageable. Thus, constant input is also considered as a major overhead and is needed to be minimized. Therefore, the proposed design methodologies for the FAPA have also been evaluated with respect to constant inputs. Figs.5.3(a), (b) and (c) present the performance of the proposed FPGA with respect to the existing FPGAs in terms of required number of constant inputs for the above mentioned (i), (ii) and (iii) design approaches, respectively. This graphs of Fig.5.3 show that the slope (growth rate) of all the edges are almost equal, but the performance of the proposed design is the best among them.



(a)



(b)



(c)

Figure 5.4: Performance of the proposed work and existing works [35, 36] with respect to the critical path delay (a) Varying m while keeping the constant n (b) Varying n while keeping the constant m (c) Varying both m and n.

5.4 Delay Comparison

The delay of a logic circuit is the delay of the critical path. However, it is an NP complete problem to find the critical path specially for large circuit (for large n or m in case of FPGA). Thus, researchers used to pick the path which is the most likely candidates for the critical paths. The delay of the proposed circuit is evaluated considering the most probable critical paths. Fig.5.4 presents the performance of the proposed work with respect to the existing work [35, 36] in terms of the delay of the circuit. From Fig.5.4, we find that the proposed design performs better than the existing designs for all possible circumstances.

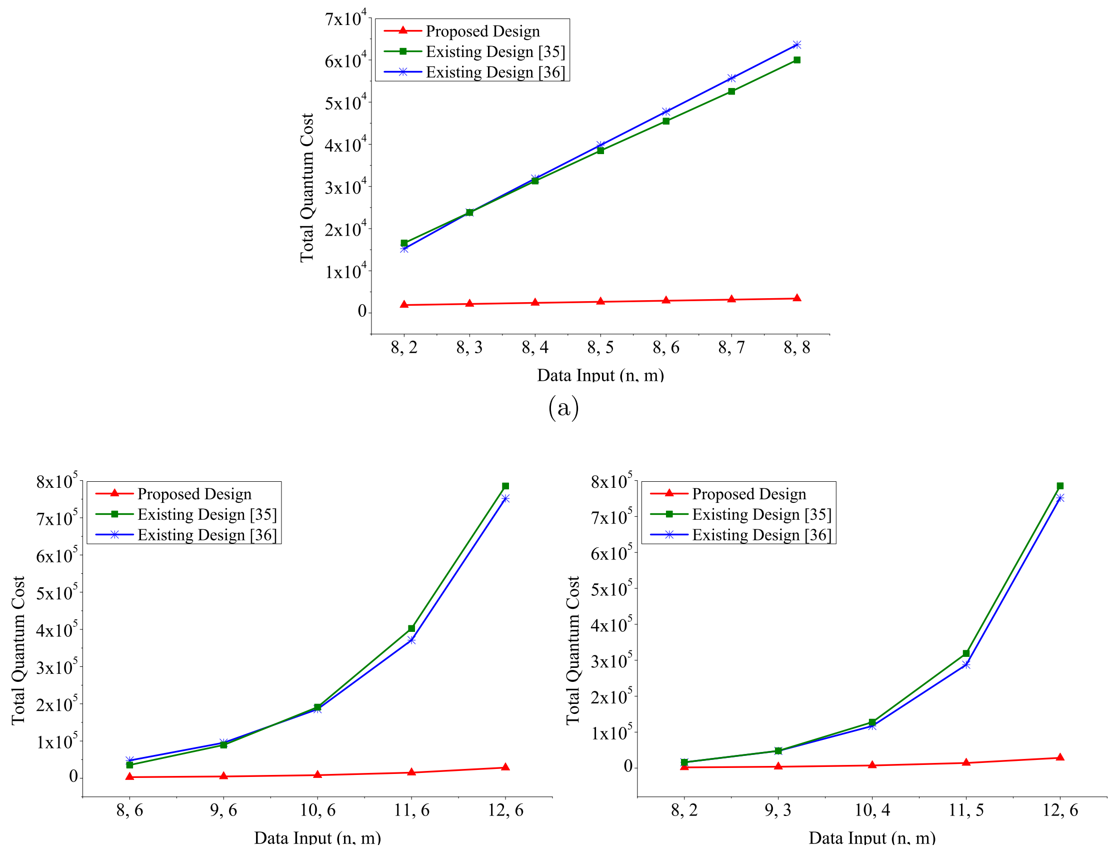


Figure 5.5: Performance of the proposed work and existing works [35, 36] with respect to the quantum cost of the circuit (a) Varying m while keeping the constant n (b) Varying n while keeping the constant m (c) Varying both m and n.

5.5 Quantum Cost Comparison

Chap.2 provides adequate background on the calculation of quantum cost for a reversible gates. Till now, the methods to calculate the quantum cost for a reversible circuit is to count the number of gates and multiply the result by the quantum cost of the number of gates. For example, in a reversible circuit let there be two types of gates, A and B for instance. Let the quantum cost of gate A be X, and quantum cost of gate B be Y. Suppose, the required numbers of gates for A and B types are N and M, respectively. Then, the quantum cost of the entire circuit is $(NX + MY)$. The proposed design methodologies for the reversible FPGA have been evaluated with respect to quantum cost of the circuit according to this above exemplified equation. Fig.5.5 compare the performance of the proposed FPGA with respect to the existing works in terms of total quantum cost of the circuits. Figs.5.5(a), (b) and (c) represents the performance for above mentioned case (i), (ii) and (iii), respectively. The graphs of Fig.5.5 show that, when the number of inputs increases, the design [35] performs better than the design [36] for case (i) but worse for cases (ii), (iii). However, the proposed scheme outperforms the existing schemes [35, 36] for all possible circumstances by a huge margin. Moreover, for the last two cases, the rate of changes with respect to the quantum cost in existing designs are exponential, whereas the proposed design changes polynomially.

5.6 Hardware Complexity Comparison

Finding a critical path from the large circuit is not always easy. Thus, the researchers calculate the hardware complexity from the circuit. Generally, a constant complexity is assumed for each basic operation and the number of operations required to realize the circuit are calculated first. Then, it is multiplied by the constant complexities. For example, let α, β and γ be the hardware complexity of the two-input Ex-OR, AND and NOT calculations, respectively. Let the hardware complexity of gates A and B of our running example be $(E\alpha + F\beta + G\gamma)$ and $(H\alpha + I\beta + J\gamma)$, respectively. Then, the hardware complexity of the circuit is $(NE\alpha + NF\beta + NG\gamma) + (MH\alpha + MI\beta + MJ\gamma) = (NE + MH)\alpha + (NF + MI)\beta + (NG + MJ)\gamma$. Table.5.1 compares the performance of the proposed design with respect to existing work in terms of hardware complexity according to this above exemplified equation. Here, the summations of total number of operations are shown at the bottom of each row. Performance of the proposed method with respect to hardware complexity for case (i) is shown in row (1,2) or (3,4). Row (2,3) shows the performance for case (ii) and the row (1,3) or (1,4) or (2,4) of this table show the performance for case (iii).

Table 5.1: Comparative study with respect to hardware complexity

$(m,\ n)$	Proposed Design	Existing Design [35]	Existing Design [36]
(8, 3)	1114α	2876α	2224α
	1368β	1159β	931β
	684γ	563γ	404γ
	3166	4598	3559
(8, 4)	1298α	2972α	2615α
	1464β	1199β	1059β
	732γ	579γ	452γ
	3494	4750	4126
(10, 4)	3730α	12796α	8897α
	4640β	4313β	2725β
	2320γ	2131γ	1268γ
	10690	19240	12890
(10, 5)	4136α	12916α	10090α
	4760β	4363β	2885β
	2380γ	2151γ	1328γ
	11276	19430	14303

5.7 Summary

This chapter provides the overall performance of the proposed reversible fault tolerant FPGA with respect to the existing reversible FPGAs [35, 36]. All the comparative results evidenced that the performance of the proposed method is much improved in terms of all the performance evaluation criterion of reversible logic synthesis, for all possible design approaches. Most interestingly, the rate of changes with respect to number of gates, garbage outputs and quantum cost in existing designs are near about exponential, whereas the proposed design changes linearly. The graphical comparisons shown here, have almost overlap edges. The reason is, difference between the values are very small with respect to the scaling factors. Therefore, A detailed tabular results is given in Appendix B.

Chapter 6

Conclusions

6.1 Summary and Contributions

This thesis presented the design methodologies of reversible fault tolerant FPGA. In the customized VLSI, FPGA provides low time-to-market and low costs compared to application specific integrated circuit and mask programmable gate arrays. The FPGA consists of an array of programmable logic blocks, input-output cells and interconnects. The programmability of logic blocks enable multiple processes to run without time sharing. Plessey and look-up table (LUT) are the most popular logic blocks of a FPGA. With more inputs, an LUT can implement more logic, hence fewer logic blocks are needed. This saves routing area, however the complexity grows exponentially with the number of inputs. Plessey logic block overcomes this difficulty through clusters of components such as NAND unit, RAM, Mux and latches. The RAM of the FPGA requires an n-to-2^n decoder and write enable master-slave flip-flops. This thesis proposed the algorithms to design compact reversible fault tolerant n-to-2^n decoder, $4n$-to-n Mux, RAM and Plessey logic block of the FPGA. In addition, reversible fault tolerant designs of D-latch, write enable master-slave flip-flop and input-output block have been presented.

Several lower bounds on the numbers of garbage outputs, constant inputs and quantum cost of the reversible fault tolerant FPGA have also been proposed in Theorems 1 to 7. The examples followed these theorems clarifies the representative analysis in detailed. It has been evidenced that the proposed circuits are constructed with the optimum garbage outputs, constant inputs and quantum cost. The efficiency and supremacy of the proposed designs have been proved through Lemmas 1 to 22. Besides, the compliance of the proposed circuits with the lemmas have been shown through several examples.

In order to implement the circuits of the fault tolerant FPGA using MOS transistors, the designs of the transistor representations of the individual gates of the proposed FPGA have been presented using standard p-MOS 901 and n-MOS 902 model with delay of $0.030ns$ and $0.12\mu m$ channel length. Then, considering the transistor level designs of the reversible fault tolerant gates as a schematic, all the proposed circuits are simulated. The simulations evidenced that the proposed fault tolerant circuits work correctly.

The comparative results show that the proposed design is much better in terms of numbers of gates, garbage outputs, quantum cost, delay, hardware complexity and are more scalable than its counterparts. Moreover, all the proposed designs have the capability of detecting errors at circuit's primary outputs which is not at all available in the existing circuits [35, 36, 37].

The proposed circuitry provides a basis for quantum computation with its applications and solves the power dissipation problem of conventional irreversible circuits. On the other hand, fault tolerant property of the proposed circuit solve the bit error problem. Moreover, proposed fault tolerant circuit can also be used in on-site hardware reconfiguration [88, 89], logic emulation [90, 91], network components designing [92, 93], aerospace and defense systems designing [94], computer vision specially in medical imaging [95], speech recognition [96, 97], inexpensive prototype development [98], digital signal processing [99, 100] etc.

6.2 Future work

The proposed FPGA is primarily based on the Plessey logic block. An interesting future enhancement can be designing reversible fault tolerant FPGA with multi-logic block capability, for example, Plessey and LUT based logic block in a single circuit while preserving bi-directionality and fault detection capability. In that case, more signals will be needed to choose among the options.

The proposed Theorems 1, 3 and 7, proved the lower bounds on the numbers of garbage outputs and constant inputs for the reversible fault tolerant decoders and Muxs. A possible future work can be realizing all the basic components of the arithmetic logic unit and the central processing unit with minimal garbage outputs and the constant inputs. Theorem 2 and Example 2 proved that the two-input logical AND operation requires at-least three quantum cost. Here, another interesting future works can be realizing all the basic and universal logical operations with minimal quantum cost. In that case, the similar theorems and examples are needed only for the logical OR, NOR and NAND operations, as the minimal quantum cost for the NOT and two-input Ex-OR are zero and one, respectively. Theorems 4 and 5 proved that the fault tolerant realization of reversible Toffoli gate requires at least one garbage output, one constant input and seven quantum cost. Providing the similar property of other non-fault tolerant reversible gate can be an interesting works. From Theorem 6 and Example 10, it can be found that at least two garbage outputs are required to realize two-input logical NAND operation in reversible and fault tolerant mode. A similar proved is given in Example 2 for the two-input logical AND operation. Here, the possible future enhancement is to realize all the basic operations in a fault tolerant modes with minimal garbage outputs and constant inputs.

The proposed designs of reversible circuits are yet to be fabricated in a chip. The future plan of this study is to manufacture the proposed circuit in a single chip for the quantum computing community.

Appendix A

Various Reversible Gates

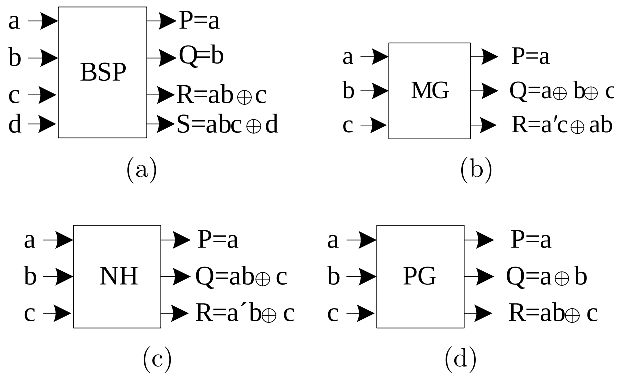


Figure A.1: Block Diagram of (a) *BSP* (b) *MG* (c) *NH* (d) *PG*

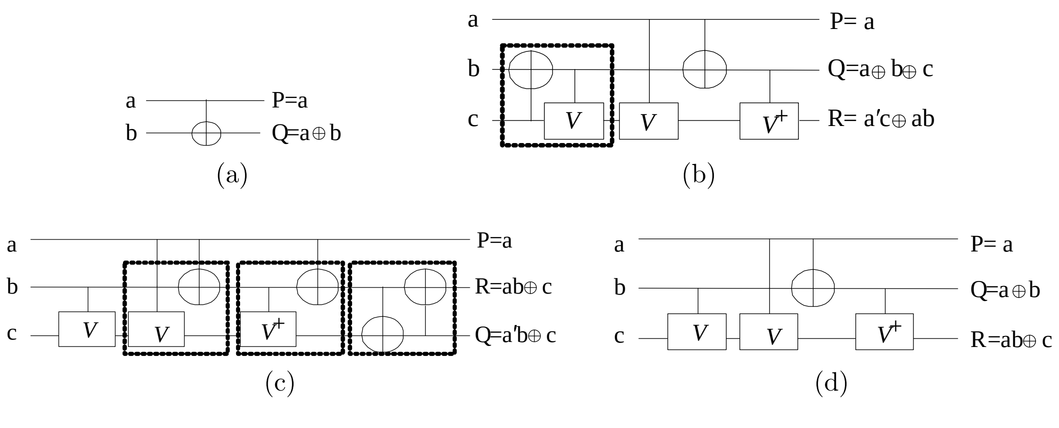
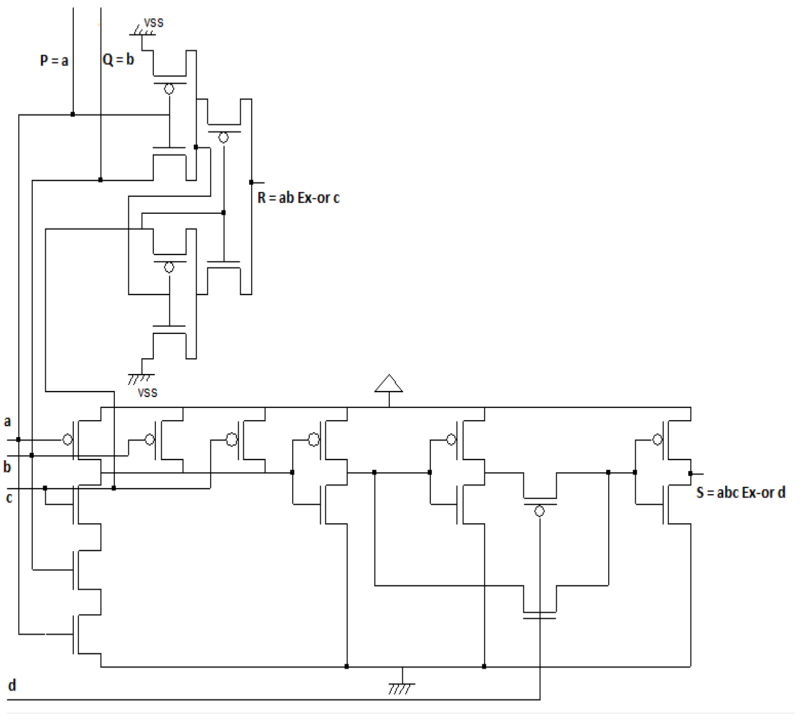
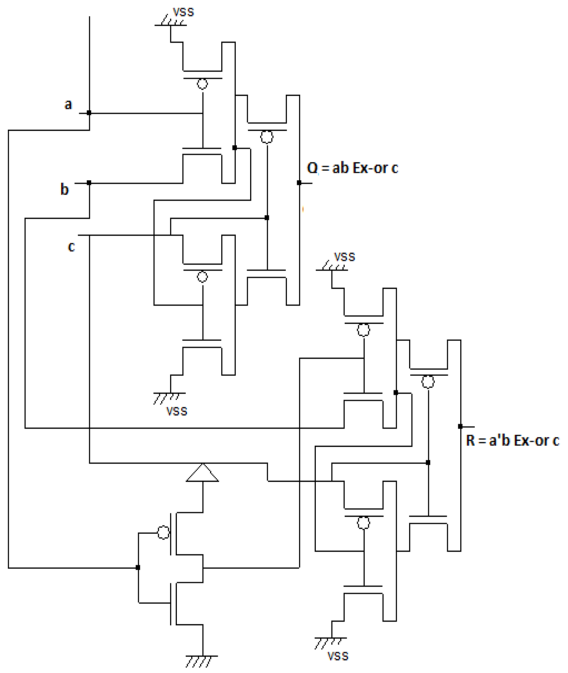
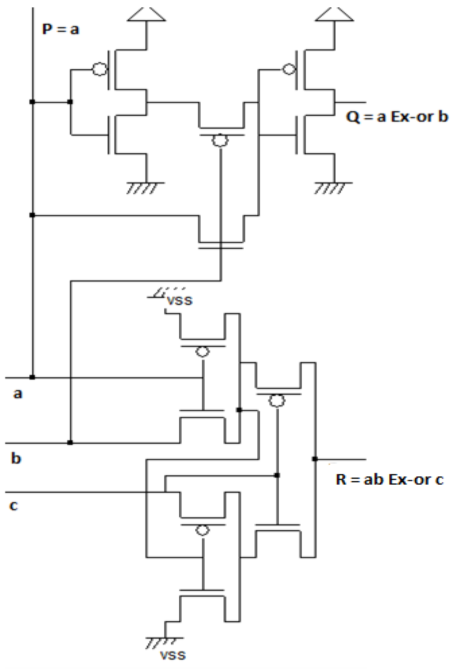


Figure A.2: Quantum realization of (a) *FG* (b) *MG* (c) *NH* (d) *PG*



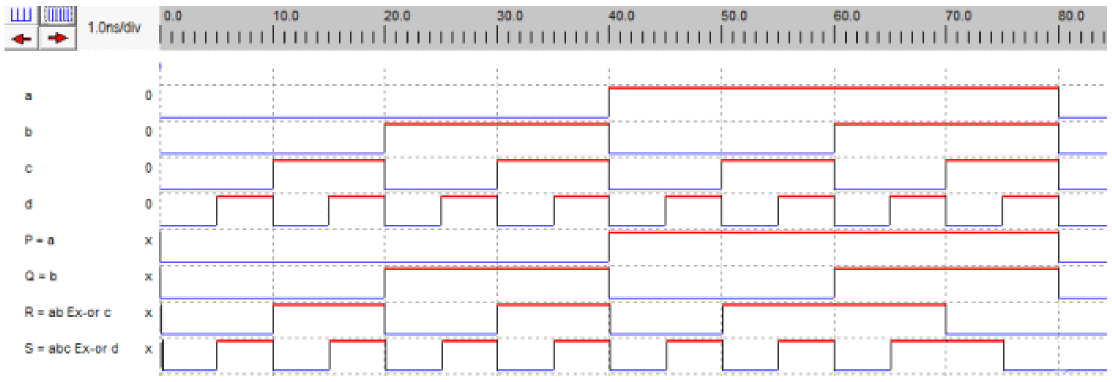
(a)



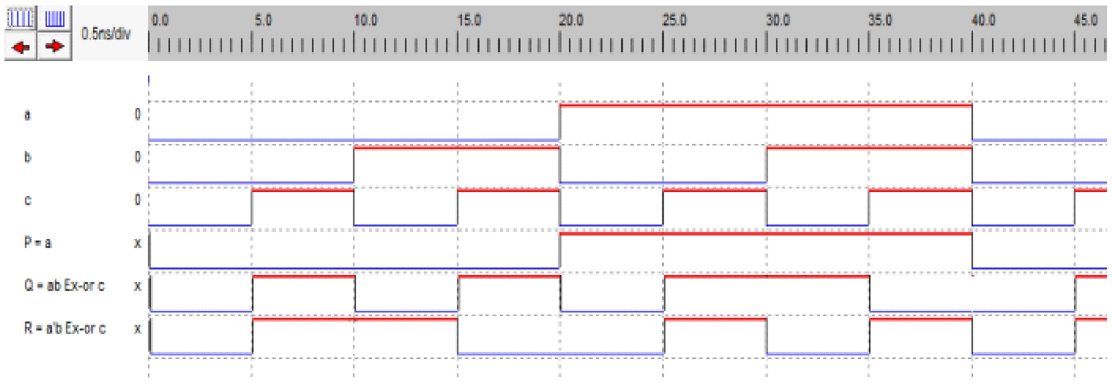


(b) (c)

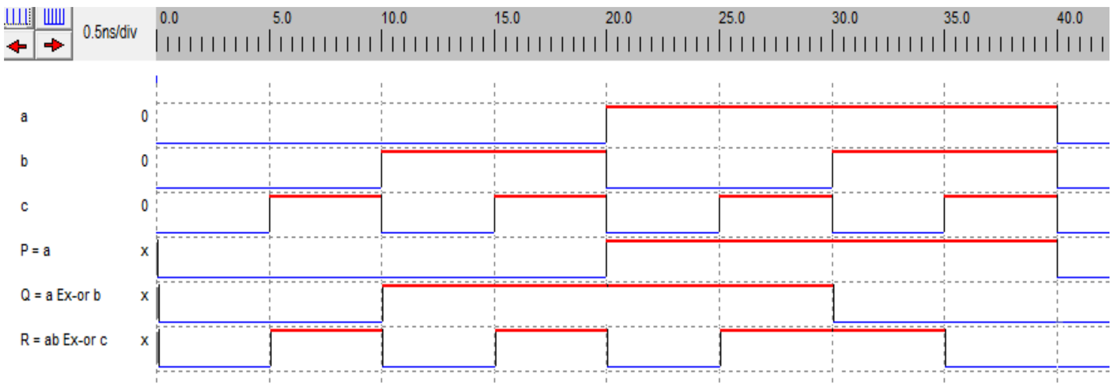
Figure A.3: Transistor realization of (a) *BSP* (b) *NH* (d) *PG*



(a)



(b)



(c)

Figure A.4: Timing Diagram of (a) *BSP* (b) *NH* (c) *PG*

Appendix B

Simulation Results

Table B.1: Number of gates : simulation results

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(8,2)	403	4584	4630
(8,3)	464	6616	6678
(8,4)	525	8648	8726
(8,5)	586	10680	10774
(8,6)	647	12712	12822
(8,7)	708	14744	14870
(8,8)	769	16776	16918
(9,2)	676	9188	9238
(9,3)	745	13266	13334
(9,4)	813	17344	17430
(9,5)	882	21422	21526
(9,6)	950	25500	25622
(9,7)	1019	29578	29718
(9,8)	1087	33656	33814
(9,9)	1156	37734	37910
(10,2)	1205	18400	18454
(10,3)	1281	26572	26646
(10,4)	1357	34744	34838
(10,5)	1433	42916	43030
(10,6)	1509	51088	51222
(10,7)	1585	59260	59414
(10,8)	1661	67432	67606
(10,9)	1737	75604	75798

Table B.2: Number of gates : simulation results (continues. . .)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(10,10)	1813	83776	83990
(11,2)	2246	36828	36886
(11,3)	2330	53190	53270
(11,4)	2413	69552	69654
(11,5)	2497	85914	86038
(11,6)	2580	102276	102422
(11,7)	2664	118638	118806
(11,8)	2747	135000	135190
(11,9)	2831	151362	151574
(11,10)	2914	167724	167958
(11,11)	2998	184086	184342
(12,2)	4311	73688	73750
(12,3)	4402	106432	106518
(12,4)	4493	139176	139286
(12,5)	4584	171920	172054
(12,6)	4675	204664	204822
(12,7)	4766	237408	237590
(12,8)	4857	270152	270358
(12,9)	4948	302896	303126
(12,10)	5039	335640	335894
(12,11)	5130	368384	368662
(12,12)	5221	401128	401430
(13,2)	8424	147412	147478
(13,3)	8523	212922	213014
(13,4)	8621	278432	278550
(13,5)	8720	343942	344086
(13,6)	8818	409452	409622
(13,7)	8917	474962	475158
(13,8)	9015	540472	540694
(13,9)	9114	605982	606230
(13,10)	9212	671492	671766
(13,11)	9311	737002	737302
(13,12)	9409	802512	802838
(13,13)	9508	868022	868374
(14,2)	16633	294864	294934
(14,3)	16739	425908	426006
(14,4)	16845	556952	557078
(14,5)	16951	687996	688150
(14,6)	17057	819040	819222
(14,7)	17163	950084	950294
(14,8)	17269	1081128	1081366
(14,9)	17375	1212172	1212438

Table B.3: Number of gates : simulation results (continues. . .)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(14,10)	17481	1343216	1343510
(14,11)	17587	1474260	1474582
(14,12)	17693	1605304	1605654
(14,13)	17799	1736348	1736726
(14,14)	17905	1867392	1867798
(15,2)	33034	589772	589846
(15,3)	33148	851886	851990
(15,4)	33261	1114000	1114134
(15,5)	33375	1376114	1376278
(15,6)	33488	1638228	1638422
(15,7)	33602	1900342	1900566
(15,8)	33715	2162456	2162710
(15,9)	33829	2424570	2424854
(15,10)	33942	2686684	2686998
(15,11)	34056	2948798	2949142
(15,12)	34169	3210912	3211286
(15,13)	34283	3473026	3473430
(15,14)	34396	3735140	3735574
(15,15)	34510	3997254	3997718
(16,2)	65819	1179592	1179670
(16,4)	66061	2228104	2228246
(16,8)	66545	4325128	4325398
(32,2)	4294967851	77309411208	77309411350
(32,4)	4294968333	146028887816	146028888086
(32,8)	4294969297	283467841032	283467841558
(32,16)	4294971225	558345747464	558345748502
(64,2)	1.844×10^{19}	33.204×10^{19}	33.204×10^{19}
(64,8)	0.0184×10^{21}	1.21×10^{21}	1.22×10^{21}
(64,16)	0.0184×10^{21}	2.39×10^{21}	2.398×10^{21}
(64,32)	0.0184×10^{21}	4.759×10^{21}	4.760×10^{21}
(128,2)	3.40×10^{38}	6.12×10^{39}	6.125×10^{39}
(128,8)	3.402×10^{38}	2.24×10^{40}	2.245×10^{40}
(128,16)	3.402×10^{38}	4.42×10^{40}	4.423×10^{38}
(128,32)	3.4028×10^{38}	8.779×10^{40}	8.780×10^{40}
(128,64)	3.40282×10^{38}	1.749×10^{41}	1.74905×10^{41}
(256,2)	1.157×10^{77}	2.0842×10^{78}	2.08425×10^{78}
(256,4)	1.157×10^{77}	3.9369×10^{78}	3.93693×10^{78}
(256,8)	1.1579×10^{77}	7.642×10^{78}	7.6422×10^{78}
(256,16)	1.1579×10^{77}	$1.5057.6422\times10^{79}$	1.50529×10^{79}
(256,32)	1.1579×10^{77}	2.987×10^{79}	2.9874×10^{79}
(256,64)	1.15792×10^{77}	5.951×10^{79}	5.95171×10^{77}
(256,128)	1.1579208×10^{77}	1.188×10^{80}	1.18802×10^{77}

Table B.4: Number of garbage outputs : simulation results

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(8,2)	271	4576	4641
(8,3)	392	6600	6689
(8,4)	513	8624	8737
(8,5)	634	10648	10785
(8,6)	755	12672	12833
(8,7)	876	14696	14881
(8,8)	997	16720	16929
(9,2)	415	9179	9250
(9,3)	607	13248	13346
(9,4)	799	17317	17442
(9,5)	991	21386	21538
(9,6)	1183	25455	25634
(9,7)	1375	29524	29730
(9,8)	1567	33593	33826
(9,9)	1759	37662	37922
(10,2)	687	18390	18467
(10,3)	1014	26552	26659
(10,4)	1341	34714	34851
(10,5)	1668	42876	43043
(10,6)	1995	51038	51235
(10,7)	2322	59200	59427
(10,8)	2649	67362	67619
(10,9)	2976	75524	75811
(10,10)	3303	83686	84003
(11,2)	1215	36817	36900
(11,3)	1805	53168	53284
(11,4)	2395	69519	69668
(11,5)	2985	85870	86052
(11,6)	3575	102221	102436
(11,7)	4165	118572	118820
(11,8)	4755	134923	135204
(11,9)	5345	151274	151588
(11,10)	5935	167625	167972
(11,11)	6525	183976	184356
(12,2)	2255	73676	73765
(12,3)	3364	106408	106533
(12,4)	4473	139140	139301
(12,5)	5582	171872	172069
(12,6)	6691	204604	204837
(12,7)	7800	237336	237605
(12,8)	8909	270068	270373
(12,9)	10018	302800	303141

Table B.5: Number of garbage outputs : simulation results (continues. . .)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(12,10)	11127	335532	335909
(12,11)	12236	368264	368677
(12,12)	13345	400996	401445
(13,2)	4319	147399	147494
(13,3)	6459	212896	213030
(13,4)	8599	278393	278566
(13,5)	10739	343890	344102
(13,6)	12879	409387	409638
(13,7)	15019	474884	475174
(13,8)	17159	540381	540710
(13,9)	19299	605878	606246
(13,10)	21439	671375	671782
(13,11)	23579	736872	737318
(13,12)	25719	802369	802854
(13,13)	27859	867866	868390
(14,2)	8431	294850	294951
(14,3)	12626	425880	426023
(14,4)	16821	556910	557095
(14,5)	21016	687940	688167
(14,6)	25211	818970	819239
(14,7)	29406	950000	950311
(14,8)	33601	1081030	1081383
(14,9)	37796	1212060	1212455
(14,10)	41991	1343090	1343527
(14,11)	46186	1474120	1474599
(14,12)	50381	1605150	1605671
(14,13)	54576	1736180	1736743
(14,14)	58771	1867210	1867815
(15,2)	16639	589757	589864
(15,3)	24937	851856	852008
(15,4)	33235	1113955	1114152
(15,5)	41533	1376054	1376296
(15,6)	49831	1638153	1638440
(15,7)	58129	1900252	1900584
(15,8)	66427	2162351	2162728
(15,9)	74725	2424450	2424872
(15,10)	83023	2686549	2687016
(15,11)	91321	2948648	2949160
(15,12)	99619	3210747	3211304
(15,13)	107917	3472846	3473448
(15,14)	116215	3734945	3735592
(15,15)	124513	3997044	3997736

Table B.6: Number of garbage outputs : simulation results (continues. . .)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(16,2)	33039	1179576	1179689
(16,3)	49536	1703816	1703977
(16,4)	66033	2228056	2228265
(16,5)	82530	2752296	2752553
(16,6)	99027	3276536	3276841
(16,7)	115524	3800776	3801129
(16,8)	132021	4325016	4325417
(16,9)	148518	4849256	4849705
(16,10)	165015	5373496	5373993
(16,11)	181512	5897736	5898281
(16,12)	198009	6421976	6422569
(16,13)	214506	6946216	6946857
(16,14)	231003	7470456	7471145
(16,15)	247500	7994696	7995433
(16,16)	263997	8518936	8519721
(64,2)	9.22×10^{18}	3.32×10^{20}	3.32×10^{18}
(64,4)	1.84×10^{19}	6.28×10^{20}	6.27×10^{20}
(64,8)	3.69×10^{19}	1.22×10^{21}	1.22×10^{21}
(64,16)	7.38×10^{19}	2.40×10^{21}	2.40×10^{21}
(64,32)	1.48×10^{19}	4.76×10^{21}	4.76×10^{21}
(128,2)	1.70×10^{38}	6.13×10^{39}	6.13×10^{39}
(128,4)	3.40×10^{38}	1.16×10^{40}	1.16×10^{40}
(128,8)	6.80×10^{38}	2.25×10^{40}	2.25×10^{40}
(128,16)	1.36×10^{39}	4.424×10^{40}	4.423×10^{40}
(128,32)	2.72×10^{39}	8.78×10^{40}	8.78×10^{40}
(128,64)	5.44×10^{39}	1.75×10^{41}	1.75×10^{41}
(256,2)	5.79×10^{76}	2.08×10^{78}	2.084×10^{78}
(256,4)	1.16×10^{77}	3.94×10^{78}	3.94×10^{78}
(256,8)	2.32×10^{77}	7.64×10^{78}	7.64×10^{78}
(256,16)	4.63×10^{77}	1.51×10^{79}	1.51×10^{79}
(256,32)	9.26×10^{77}	2.99×10^{79}	2.98×10^{79}
(256,64)	1.85×10^{78}	5.951×10^{79}	5.951×10^{79}
(256,128)	3.70×10^{79}	1.188×10^{80}	1.188×10^{80}
(512,2)	6.7×10^{153}	2.41×10^{155}	2.41×10^{155}
(512,4)	1.34×10^{154}	4.56×10^{155}	4.65×10^{155}
(512,8)	2.68×10^{154}	8.84×10^{155}	8.849×10^{155}
(512,16)	5.36×10^{154}	1.74×10^{156}	1.743×10^{156}
(512,32)	1.07×10^{155}	3.46×10^{156}	3.46×10^{156}
(512,64)	2.15×10^{155}	6.89×10^{156}	6.89×10^{156}
(512,128)	4.29×10^{155}	1.375×10^{157}	1.3756×10^{157}
(512,256)	8.58×10^{155}	2.75×10^{157}	2.75×10^{157}

Table B.7: Number of constant inputs : simulation results

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(8,2)	412	585	713
(8,3)	484	617	801
(8,4)	556	649	889
(8,5)	628	681	977
(8,6)	700	713	1065
(8,7)	772	745	1153
(8,8)	844	777	1241
(9,2)	687	1106	1248
(9,3)	768	1142	1347
(9,4)	849	1178	1446
(9,5)	930	1214	1545
(9,6)	1011	1250	1644
(9,7)	1092	1286	1743
(9,8)	1173	1322	1842
(9,9)	1254	1358	1941
(10,2)	1218	2139	2295
(10,3)	1308	2179	2405
(10,4)	1398	2219	2515
(10,5)	1488	2259	2625
(10,6)	1578	2299	2735
(10,7)	1668	2339	2845
(10,8)	1758	2379	2955
(10,9)	1848	2419	3065
(10,10)	1938	2459	3175
(11,2)	2261	4196	4366
(11,3)	2360	4240	4487
(11,4)	2459	4284	4608
(11,5)	2558	4328	4729
(11,6)	2657	4372	4850
(11,7)	2756	4416	4971
(11,8)	2855	4460	5092
(11,9)	2954	4504	5213
(11,10)	3053	4548	5334
(11,11)	3152	4592	5455
(12,2)	4328	8301	8485
(12,3)	4436	8349	8617
(12,4)	4544	8397	8749
(12,5)	4652	8445	8881
(12,6)	4760	8493	9013
(12,7)	4868	8541	9145
(12,8)	4976	8589	9277
(12,9)	5084	8637	9409

Table B.8: Number of constant inputs : simulation results (continues…)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(12,10)	5192	8685	9541
(12,11)	5300	8733	9673
(12,12)	5408	8781	9805
(13,2)	8443	16502	16700
(13,3)	8560	16554	16843
(13,4)	8677	16606	16986
(13,5)	8794	16658	17129
(13,6)	8911	16710	17272
(13,7)	9028	16762	17415
(13,8)	9145	16814	17558
(13,9)	9262	16866	17701
(13,10)	9379	16918	17844
(13,11)	9496	16970	17987
(13,12)	9613	17022	18130
(13,13)	9730	17074	18273
(14,2)	16654	32895	33107
(14,3)	16780	32951	33261
(14,4)	16906	33007	33415
(14,5)	17032	33063	33569
(14,6)	17158	33119	33723
(14,7)	17284	33175	33877
(14,8)	17410	33231	34031
(14,9)	17536	33287	34185
(14,10)	17662	33343	34339
(14,11)	17788	33399	34493
(14,12)	17914	33455	34647
(14,13)	18040	33511	34801
(14,14)	18166	33567	34955
(15,2)	33057	65672	65898
(15,3)	33192	65732	66063
(15,4)	33327	65792	66228
(15,5)	33462	65852	66393
(15,6)	33597	65912	66558
(15,7)	33732	65972	66723
(15,8)	33867	66032	66888
(15,9)	34002	66092	67053
(15,10)	34137	66152	67218
(15,11)	34272	66212	67383
(15,12)	34407	66272	67548
(15,13)	34542	66332	67713
(15,14)	34677	66392	67878
(15,15)	34812	66452	68043

Table B.9: Number of constant inputs : simulation results (continues. . .)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(16,2)	65844	131217	131457
(16,3)	65988	131281	131633
(16,4)	66132	131345	131809
(16,5)	66276	131409	131985
(16,6)	66420	131473	132161
(16,7)	66564	131537	132337
(16,8)	66708	131601	132513
(16,9)	66852	131665	132689
(16,10)	66996	131729	132865
(16,11)	67140	131793	133041
(16,12)	67284	131857	133217
(16,13)	67428	131921	133393
(16,14)	67572	131985	133569
(16,15)	67716	132049	133745
(16,16)	67860	132113	133921
(64,2)	1.84×10^{19}	3.69×10^{19}	3.69×10^{19}
(64,4)	1.84×10^{19}	3.69×10^{19}	3.69×10^{19}
(64,8)	1.84×10^{19}	3.69×10^{19}	3.69×10^{19}
(64,16)	1.84×10^{19}	3.69×10^{19}	3.69×10^{19}
(64,32)	1.84×10^{19}	3.69×10^{19}	3.69×10^{19}
(128,2)	3.40×10^{38}	6.81×10^{38}	6.81×10^{38}
(128,4)	3.40×10^{38}	6.81×10^{38}	6.81×10^{38}
(128,8)	3.40×10^{38}	6.81×10^{38}	6.81×10^{38}
(128,16)	3.40×10^{38}	6.81×10^{38}	6.81×10^{38}
(128,32)	3.40×10^{38}	6.81×10^{38}	6.81×10^{38}
(128,64)	3.40×10^{38}	6.81×10^{38}	6.81×10^{38}
(256,2)	1.157×10^{77}	2.32×10^{77}	2.32×10^{77}
(256,4)	1.157×10^{77}	2.32×10^{77}	2.32×10^{77}
(256,8)	1.157×10^{77}	2.32×10^{77}	2.32×10^{77}
(256,16)	1.157×10^{77}	2.32×10^{77}	2.32×10^{77}
(256,32)	1.157×10^{77}	2.32×10^{77}	2.32×10^{77}
(256,64)	1.157×10^{77}	2.32×10^{77}	2.32×10^{77}
(256,128)	1.157×10^{77}	2.32×10^{77}	2.32×10^{77}
(512,2)	1.34×10^{154}	2.68×10^{154}	2.68×10^{154}
(512,4)	1.34×10^{154}	2.68×10^{154}	2.68×10^{154}
(512,8)	1.34×10^{154}	2.68×10^{154}	2.68×10^{154}
(512,16)	1.34×10^{154}	2.68×10^{154}	2.68×10^{154}
(512,32)	1.34×10^{154}	2.68×10^{154}	2.68×10^{154}
(512,64)	1.34×10^{154}	2.68×10^{154}	2.68×10^{154}
(512,128)	1.34×10^{154}	2.68×10^{154}	2.68×10^{154}
(512,256)	1.34×10^{154}	2.68×10^{154}	2.68×10^{154}

Table B.10: Critical path delay : simulation results

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(8,2)	152	158	280
(8,3)	160	163	287
(8,4)	168	169	294
(9,2)	280	288	537
(9,3)	288	293	544
(9,4)	296	298	551
(9,5)	304	304	558
(9,6)	312	310	565
(10,2)	537	546	1050
(10,3)	544	551	1057
(10,4)	552	556	1064
(10,5)	560	561	1071
(10,6)	568	567	1078
(10,7)	576	573	1085
(11,2)	1050	1060	2075
(11,3)	1057	1065	2082
(11,4)	1064	1070	2089
(11,5)	1072	1075	2096
(11,6)	1080	1080	2103
(12,2)	2075	2086	4124
(12,3)	2082	2091	4131
(12,4)	2089	2096	4138
(12,5)	2096	2101	4145
(12,6)	2104	2106	4152
(13,2)	4124	4136	8221
(13,3)	4131	4141	8228
(13,4)	4138	4146	8235
(13,5)	4145	4151	8242
(13,6)	4152	4156	8249
(14,2)	8221	8234	16414
(14,3)	8228	8239	16421
(14,4)	8235	8244	16428
(14,5)	8242	8249	16435
(14,6)	8249	8254	16442
(14,7)	8256	8259	16449
(14,8)	8264	8264	16456
(15,2)	16414	16428	32799
(15,3)	16421	16433	32806
(15,4)	16428	16438	32813
(15,5)	16435	16443	32820
(15,6)	16442	16448	32827

Table B.11: Critical path delay : simulation results (continues. . .)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(16,2)	32799	32814	65568
(16,3)	32806	32819	65575
(16,4)	32813	32824	65582
(16,5)	32820	32829	65589
(16,6)	32827	32834	65596
(16,7)	32834	32839	65603
(16,8)	32841	32844	65610
(17,2)	65568	65584	131105
(17,3)	65575	65589	131112
(17,4)	65582	65594	131119
(17,5)	65589	65599	131126
(17,6)	65596	65604	131133
(17,7)	65603	65609	131140
(17,8)	65610	65614	131147
(17,9)	65617	65619	131154
(18,2)	131105	131122	262178
(18,3)	131112	131127	262185
(18,4)	131119	131132	262192
(18,5)	131126	131137	262199
(18,6)	131133	131142	262206
(18,7)	131140	131147	262213
(18,8)	131147	131152	262220
(18,9)	131154	131157	262227
(18,10)	131161	131162	262234
(19,2)	262178	262196	524323
(19,3)	262185	262201	524330
(19,4)	262192	262206	524337
(19,5)	262199	262211	524344
(19,6)	262206	262216	524351
(19,7)	262213	262221	524358
(19,8)	262220	262226	524365
(32,2)	2147483695	2147483726	4294967344
(32,3)	2147483702	2147483731	4294967351
(32,4)	2147483709	2147483736	4294967358
(32,5)	2147483716	2147483741	4294967365
(32,6)	2147483723	2147483746	4294967372
(32,7)	2147483730	2147483751	4294967379
(32,8)	2147483737	2147483756	4294967386
(32,9)	2147483744	2147483761	4294967393
(32,10)	2147483751	2147483766	4294967400
(32,16)	2147483793	2147483796	4294967442
(32,17)	2147483800	2147483801	4294967449

Table B.12: Total quantum cost : simulation results

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(8,2)	1884	16073	16000
(8,3)	2140	23937	23936
(8,4)	2396	31801	31872
(8,5)	2652	39665	39808
(8,6)	2908	47529	47744
(8,7)	3164	55393	55680
(8,8)	3420	63257	63616
(9,2)	3347	32183	31872
(9,3)	3691	47974	47744
(9,4)	4035	63765	63616
(9,5)	4379	79556	79488
(9,6)	4723	95347	95360
(9,7)	5067	111138	111232
(9,8)	5411	126929	127104
(9,9)	5755	142720	142976
(10,2)	6218	64421	63616
(10,3)	6714	96075	95360
(10,4)	7210	127729	127104
(10,5)	7706	159383	158848
(10,6)	8202	191037	190592
(10,7)	8698	222691	222336
(10,8)	9194	254345	254080
(10,9)	9690	285999	285824
(10,10)	10186	317653	317568
(11,2)	11905	128915	127104
(11,3)	12681	192304	190592
(11,4)	13457	255693	254080
(11,5)	14233	319082	317568
(11,6)	15009	382471	381056
(11,7)	15785	445860	444544
(11,8)	16561	509249	508032
(11,9)	17337	572638	571520
(11,10)	18113	636027	635008
(11,11)	18889	699416	698496
(12,2)	23224	257921	254080
(12,3)	24536	384789	381056
(12,4)	25848	511657	508032
(12,5)	27160	638525	635008
(12,6)	28472	765393	761984
(12,7)	29784	892261	888960
(12,8)	31096	1019129	1015936
(12,9)	32408	1145997	1142912

Table B.13: Total quantum cost : simulation results (continues. . .)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(12,10)	33720	1272865	1269888
(12,11)	35032	1399733	1396864
(12,12)	36344	1526601	1523840
(13,2)	45807	515951	508032
(13,3)	48167	769786	761984
(13,4)	50527	1023621	1015936
(13,5)	52887	1277456	1269888
(13,6)	55247	1531291	1523840
(13,7)	57607	1785126	1777792
(13,8)	59967	2038961	2031744
(13,9)	62327	2292796	2285696
(13,10)	64687	2546631	2539648
(13,11)	67047	2800466	2793600
(13,12)	69407	3054301	3047552
(13,13)	71767	3308136	3301504
(14,2)	90918	1032029	1015936
(14,3)	95350	1539807	1523840
(14,4)	99782	2047585	2031744
(14,5)	104214	2555363	2539648
(14,6)	108646	3063141	3047552
(14,7)	113078	3570919	3555456
(14,8)	117510	4078697	4063360
(14,9)	121942	4586475	4571264
(14,10)	126374	5094253	5079168
(14,11)	130806	5602031	5587072
(14,12)	135238	6109809	6094976
(14,13)	139670	6617587	6602880
(14,14)	144102	7125365	7110784
(15,2)	181085	2064203	2031744
(15,3)	189637	3079876	3047552
(15,4)	198189	4095549	4063360
(15,5)	206741	5111222	5079168
(15,6)	215293	6126895	6094976
(15,7)	223845	7142568	7110784
(15,8)	232397	8158241	8126592
(15,9)	240949	9173914	9142400
(15,10)	249501	10189587	10158208
(15,11)	258053	11205260	11174016
(15,12)	266605	12220933	12189824
(15,13)	275157	13236606	13205632
(15,14)	283709	14252279	14221440
(15,15)	292261	15267952	15237248

Table B.14: Total quantum cost : simulation results (continues. . .)

(n,m)	Proposed Work	Existing Work [35]	Existing Work [36]
(16,2)	361364	4128569	4063360
(16,3)	378132	6160041	6094976
(16,4)	394900	8191513	8126592
(16,5)	411668	10222985	10158208
(16,6)	428436	12254457	12189824
(16,7)	445204	14285929	14221440
(16,8)	461972	16317401	16253056
(16,9)	478740	18348873	18284672
(16,10)	495508	20380345	20316288
(16,11)	512276	22411817	22347904
(16,12)	529044	24443289	24379520
(16,13)	545812	26474761	26411136
(16,14)	562580	28506233	28442752
(16,15)	579348	30537705	30474368
(16,16)	596116	32569177	32505984
(64,2)	1.01×10^{20}	1.16×10^{21}	1.14×10^{21}
(64,4)	1.10×10^{20}	2.30×10^{21}	2.28×10^{21}
(64,8)	1.29×10^{20}	4.59×10^{21}	4.57×10^{21}
(64,16)	1.66×10^{20}	9.17×10^{21}	9.15×10^{21}
(64,32)	2.40×10^{20}	1.83×10^{22}	1.83×10^{22}
(128,2)	1.87×10^{39}	2.14×10^{40}	2.11×10^{40}
(128,4)	2.04×10^{39}	4.25×10^{40}	4.22×10^{40}
(128,8)	2.38×10^{39}	8.47×10^{40}	8.43×10^{40}
(128,16)	3.06×10^{39}	1.691×10^{41}	1.687×10^{41}
(128,32)	4.42×10^{39}	3.379×10^{41}	3.375×10^{41}
(128,64)	7.15×10^{39}	6.754×10^{41}	6.751×10^{41}
(256,2)	6.37×10^{77}	7.29×10^{78}	7.18×10^{77}
(256,4)	6.94×10^{77}	1.45×10^{79}	1.44×10^{79}
(256,8)	8.11×10^{77}	2.88×10^{79}	2.87×10^{79}
(256,16)	1.04×10^{78}	5.75×10^{79}	5.74×10^{79}
(256,32)	1.51×10^{78}	1.159×10^{79}	1.148×10^{79}
(256,64)	2.43×10^{78}	2.298×10^{79}	2.297×10^{79}
(256,128)	4.28×10^{78}	4.595×10^{79}	4.594×10^{79}
(512,2)	7.37×10^{154}	8.44×10^{155}	8.31×10^{155}
(512,4)	8.04×10^{154}	1.67×10^{156}	1.66×10^{156}
(512,8)	9.38×10^{154}	3.338×10^{156}	3.325×10^{156}
(512,16)	1.20×10^{155}	6.663×10^{156}	6.650×10^{156}
(512,32)	1.74×10^{155}	1.331×10^{156}	1.330×10^{156}
(512,64)	2.81×10^{155}	2.661×10^{156}	2.660×10^{156}
(512,128)	4.96×10^{155}	5.321×10^{156}	5.320×10^{156}

Bibliography

[1] J. C. Maxwell, *Theory of heat*, 4th ed. Longmans, Green and co, 1875.

[2] L. Szilard, "On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings maxwell's demon 2 entropy," *Classical and quantum information computing*, pp. 840–856, 1929.

[3] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Dev.*, vol. 5, no. 3, pp. 183–191, Jul. 1961. [Online]. Available: http://dx.doi.org/10.1147/rd.53.0183

[4] C. H. Bennett, "Logical reversibility of computation," *IBM J. Res. Dev.*, vol. 17, no. 6, pp. 525–532, Nov. 1973. [Online]. Available: http://dx.doi.org/10.1147/rd.176.0525

[5] H. F. Chau and F. Wilczek, "Simple realization of the Fredkin gate using a series of two-body operators," *Phys. Rev. Lett.*, vol. 75, pp. 748–750, Jul. 1995. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.75.748

[6] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, Apr. 1965. [Online]. Available: http://dx.doi.org/10.1109/JPROC.1998.658762

[7] M. Nielsen and I. Chuang, *Quantum computation and quantum information.* New York, USA: Cambridge University Press, 2000.

[8] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1510–1523, Oct. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0097539796300933

[9] M. S. Islam, M. M. Rahman, Z. Begum, M. Z. Hafiz, and A. A. Mahmud, "Synthesis of fault tolerant reversible logic circuits," *CoRR*, vol. abs/1008.3340, pp. 1–4, 2010. [Online]. Available: http://arxiv.org/abs/1008.3340

[10] I. Koren and C. M. Krishna, "Fault-tolerant systems." Morgan Kaufmann Publishers Inc. San Francisco, USA, 2007.

[11] B. Parhami, "Fault-tolerant reversible circuits," in *Fortieth Asilomar Conference on Signals, Systems and Computers.*, 2006, pp. 1726 –1729.

[12] J. Mathew, J. Singh, A. A. Taleb, and D. K. Pradhan, "Fault tolerant reversible finite field arithmetic circuits," in *Proceedings of the 14th IEEE International On-Line Testing Symposium.* Washington, DC, USA: IEEE Computer Society, 2008, pp. 188–189.

[13] R. K. James, T. K. Shahana, K. P. Jacob, and S. Sasi, "Fault tolerant error coding and detection using reversible gates," in *IEEE TENCON*, 2007, pp. 1–4.

[14] M. P. Frank, "The physical limits of computing," *Computing in Science and Engg.*, vol. 4, no. 3, pp. 16–26, May 2002.

[15] A. K. Biswas, M. M. Hasan, A. R. Chowdhury, and H. M. Hasan Babu, "Efficient approaches for designing reversible binary coded decimal adders," *Microelectron. J.*, vol. 39, no. 12, pp. 1693–1703, Dec. 2008.

[16] G. Yang, F. Xie, W. N. N. Hung, X. Song, and M. A. Perkowski, "Realization and synthesis of reversible functions," *Theor. Comput. Sci.*, vol. 412, no. 17, pp. 1606–1613, 2011.

[17] M. Perkowski, "Reversible computation for beginners," lecture series, Portland state university, 2000. [Online]. Available: http://www.ee.pdx.edu/mperkows

[18] S. N. Mahammad and K. Veezhinathan, "Constructing online testable circuits using reversible logic," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, pp. 101–109, 2010.

[19] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. A. Perkowski, "Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 9, pp. 1652–1663, 2006.

[20] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Phys. Rev. A*, vol. 54, pp. 147–153, Jul 1996. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevA.54.147

[21] D. Maslov, G. W. Dueck, and N. Scott, "Reversible logic synthesis benchmarks page," 2005. [Online]. Available: http://webhome.cs.uvic.ca/~dmaslov

[22] E. P. A. Akbar, M. Haghparast, and K. Navi, "Novel design of a fast reversible wallace sign multiplier circuit in nano-technology," *Microelectronics Journal*, vol. 42, no. 8, pp. 973–981, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026269211001194

[23] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings of the 40th annual Design Automation Conference*, Anaheim, CA, USA, 2003, pp. 318–323.

[24] A. Gepp and P. Stocks, "A review of procedures to evolve quantum algorithms," *Genetic Programming and Evolvable Machines*, vol. 10, no. 2, pp. 181–228, Jun. 2009.

[25] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. Washington, DC, USA: IEEE Computer Society, 1994, pp. 124–134.

[26] V. Betz and J. Rose, "How much logic should go in an FPGA logic block?" in *IEEE Design and Test Magazine*, 1998, pp. 10–15.

[27] R. J. Francis, "A tutorial on logic synthesis for lookup-table based FPGAs," in *Proceedings of the IEEE/ACM international conference on Computer-aided design*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1992, pp. 40–47.

[28] T. Matsunaga, S. Kimura, and Y. Matsunaga, "Power and delay aware synthesis of multi-operand adders targeting LUT-based FPGAs," in *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*. Piscataway, NJ, USA: IEEE Press, 2011, pp. 217–222.

[29] J. Cong and Y. Ding, "Combinational logic synthesis for LUT based field programmable gate arrays," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 1, no. 2, pp. 145–204, Apr. 1996.

[30] N. B. Bhat and D. D. Hill, "Routable technologie mapping for LUT FPGAs," in *Proceedings of the 1991 IEEE International Conference on Computer Design on VLSI in Computer & Processors*. Washington, DC, USA: IEEE Computer Society, 1992, pp. 95–98. [Online]. Available: http://dl.acm.org/citation.cfm?id=645461.654583

[31] J. E. Vuillemin, P. Bertin, D. Roncin, M. Shand, H. H. Touati, and P. Boucard, "Readings in hardware/software co-design." Norwell, MA, USA: Kluwer Academic Publishers, 2002, ch. Programmable active memories: reconfigurable systems come of age, pp. 611–624.

[32] B. Heeb and C. Pfister, "Chameleon: A workstation of a different colour," in *Selected papers from the Second International Workshop on Field-Programmable Logic and Applications, Field-Programmable Gate Arrays: Architectures and Tools for Rapid Prototyping.* London, UK, UK: Springer-Verlag, 1993, pp. 152–161.

[33] Plessey, *Plessey Semiconductor ERA60100, Priliminary data sheet.* England: Swindon, 1989.

[34] V. Betz and J. Rose, "Cluster-based logic blocks for FPGAs: Area-efficiency vs. input sharing and size," *IEEE 1997 Custom Integrated Circuits Conference*, pp. 551–554, 1997.

[35] A. S. M. Sayem and S. K. Mitra, "Efficient approach to design low power reversible logic blocks for field programmable gate arrays," in *IEEE International Conference on Computer Science and Automation Engineering*, Shanghai, China, July 2011, pp. 251–255.

[36] A. S. M. Sayem, M. M. A. Polash, and H. M. H. Babu, "Design of a reversible logic block of field programmable gate array," in *Silver Jubilee Conference on Communication Technologies and VLSI design*, VIT University, Vellore, India, 2009, pp. 500–501.

[37] M. M. A. Polash and S. Sultana, "Design of a LUT-based reversible field programmable gate array," *J. Com.*, vol. 2, pp. 103–108, Oct. 2010.

[38] X. Han and C. Xu, "Design and characterization of high-voltage NMOS and PMOS devices in standard 0.25&#181;m CMOS technology," *Microelectron. J.*, vol. 38, no. 10-11, pp. 1038–1041, Oct. 2007.

[39] M. Fadlallah, G. Ghibaudo, M. Bidaud, O. Simonetti, and F. Guyader, "Stress-induced leakage current at low field in NMOS and PMOS devices with ultra-thin nitrided gate Oxide," *Microelectron. Eng.*, vol. 72, no. 1-4, pp. 241–246, May 2004.

[40] M. Ashouei, A. D. Singh, and A. Chatterjee, "Reconfiguring CMOS as pseudo N/PMOS for defect tolerance in nano-scale CMOS," in *Proceedings of the 21st International Conference on VLSI Design.* Washington, DC, USA: IEEE Computer Society, 2008, pp. 27–32.

[41] H. M. Hasan Babu, R. Islam, A. R. Chowdhury, and S. M. A. Chowdhury, "Reversible logic synthesis for minimization of full-adder circuit," in *Euromicro Symposium on Digital Systems Design.* Los Alamitos, CA, USA: IEEE Computer Society, 2003, pp. 50–53.

[42] Z. Guan, Z. Bao, and W. Jing, "The cascade of the reversible gate network-based the dynamic binary spanning tree," in *Proceedings of the 2009 Second International Workshop on Computer Science and Engineering.* Washington, DC, USA: IEEE Computer Society, 2009, pp. 403–407. [Online]. Available: http://dx.doi.org/10.1109/WCSE.2009.697

[43] P. O. Boykin and V. P. Roychowdhury, "Reversible fault-tolerant logic," in *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, ser. DSN '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 444–453.

[44] J. Donald and N. K. Jha, "Reversible logic synthesis with Fredkin and Peres gates," *J. Emerg. Technol. Comput. Syst.*, vol. 4, no. 1, pp. 2:1–2:19, Apr. 2008. [Online]. Available: http://doi.acm.org/10.1145/1330521.1330523

[45] J. B. Altepeter, "A tale of two qubits: how quantum computers work." [Online]. Available: http://arstechnica.com/science/guides/2010/01

[46] Z. Li, H. Chen, B. Xu, X. Song, and X. Xue, "An algorithm for synthesis of optimal 3-qubit reversible circuits based on bit operation," in *Proceedings of the 2008 Second International Conference on Genetic and Evolutionary Computing*, ser. WGEC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 455–458.

[47] Y. Takahashi and N. Kunihiro, “A quantum circuit for shor’s factoring algorithm using 2n + 2 qubits,” *Quantum Info. Comput.*, vol. 6, no. 2, pp. 184–192, Mar. 2006.

[48] M. Ross and M. Oskin, “Quantum computing,” *Commun. ACM*, vol. 51, no. 7, pp. 12–13, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1364782.1364787

[49] M. Mosca and A. Ekert, “The hidden subgroup problem and eigenvalue estimation on a quantum computer,” in *Selected papers from the First NASA International Conference on Quantum Computing and Quantum Communications*, ser. QCQC ’98. London, UK, UK: Springer-Verlag, 1998, pp. 174–188. [Online]. Available: http://dl.acm.org/citation.cfm?id=645812.670797

[50] M. Lukac, M. Perkowski, P. Kerntopf, M. Pivtoraiko, M. Folgheraiter, D. Lee, H. Kim, W. Hwuangbo, J. wook Kim, Y.W. Choi, “A hierarchical approach to computer-aided design of quantum circuits,” *6th International Symposium on Representations and Methodology of Future Computing Technology*, pp. 201–209, 2003.

[51] L. Jamal, M. Shamsujjoha, and H. M. Hasan Babu, “Design of optimal reversible carry look-ahead adder with optimal garbage and quantum cost,” *International Journal of Engineering and Technology*, vol. 2, pp. 44–50, 2012. [Online]. Available: http://iet-journals.org/archive/2012/jan_vol_2_no_1/349421324456832.pdf

[52] D. Maslov, G. W. Dueck, and D. M. Miller, “Simplification of Toffoli networks via templates,” in *Proceedings of the 16th Annual Symposium on Integrated Circuits and Systems Design.* IEEE Computer Society, Sept. 2003, pp. 53–58. [Online]. Available: http://dx.doi.org/10.1109/SBCCI.2003.1232806

[53] L. Singhal, E. Bozorgzadeh, and D. Eppstein, "Interconnect criticality-driven delay relaxation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, pp. 1803–1817, Octobor 2007.

[54] L. Singhal and E. Bozorgzadeh, "Fast timing closure by interconnect criticality driven delay relaxation," in *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 792–797. [Online]. Available: http://dl.acm.org/citation.cfm?id=1129601.1129713

[55] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[56] M. S. Islam, M. M. Rahman, Z. Begum, and M. Z. Hafiz, "Fault tolerant reversible logic synthesis: Carry look-ahead and carry-skip adders," *CoRR*, vol. abs/1008.3311, 2010. [Online]. Available: http://arxiv.org/abs/1008.3311

[57] S. G. Younis, J. Knight, and T. F., "Practical implementation of charge recovering asymptotically zero power CMOS," in *Proceedings of the 1993 symposium on Research on integrated systems*. Cambridge, MA, USA: MIT Press, 1993, pp. 234–250. [Online]. Available: http://dl.acm.org/citation.cfm?id=163429.163468

[58] S. G. Younis, "Asymptotically zero energy computing using split-level charge recovery logic," Cambridge, MA, USA, Tech. Rep., 1994.

[59] J. Lim, K. Kwon, and S.-I. Chae, "Reversible energy recovery logic circuit without non-adiabatic energy loss," *Electronics Letters*, vol. 34, no. 4, pp. 344–346, 1998.

[60] J. Lim, K. Kwon, and S. I. Chae, "Reversible energy recovery logic circuits and its 8-phase clocked power generator for ultra-low-power applications," *IEICE Trans. Electron*, vol. E82-C, pp. 646–653, April 1999.

[61] J. Lim, S.-I. Chae, and D.-G. Kim, "nMOS reversible energy recovery logic for ultra-low-energy applications," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 865–875, 2000.

[62] P. D. Picton, "Optoelectronic multi-valued conservative logic," *Int. Journal of Optical Computing*, vol. 2, pp. 19–29, 1991.

[63] S. Bandyopadhyay, "Nanoelectric implementations of recversible and quantum logic," *Supperlattices and Microstructures*, vol. 23, pp. 445–464, 1998.

[64] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, "Online testable reversible logic circuit design using nand blocks," *Defect and Fault-Tolerance in VLSI Systems, IEEE International Symposium on*, vol. 0, pp. 324–331, 2004.

[65] H. Thapliyal and A. P. Vinod, "Design of reversible sequential elements with feasibility of transistor implementation," in *International Symposium on Circuits and Systems (ISCAS 2007).* IEEE, 2007, pp. 625–628. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ISCAS.2007.378815

[66] H. M. H. Babu, M. I. Zaber, M. M. Rahman, and M. R. Islam, "Implementation of multiple-valued flip-flips using pass transistor logic," in *2004 Euromicro Symposium on Digital Systems Design (DSD 2004), Architectures, Methods and Tools, 31 August - 3 September 2004, Rennes, France.* IEEE Computer Society, 2004, pp. 603–606.

[67] H. Thapliyal and A. P. Vinod, "Transistor realization of reversible TSG gate and reversible adder architectures," in *IEEE Asia Pacific Conference on Circuits and Systems 2006, APCCAS 2006, Singapore, 4-7 December 2006.* IEEE, 2006, pp. 418–421.

[68] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, "CMOS realization of online testable reversible logic gates," in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI: New Frontiers in VLSI Design*, ser. ISVLSI '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 309–310. [Online]. Available: http://dx.doi.org/10.1109/ISVLSI.2005.23

[69] R. Feynman, "Quantum mechanical computers," *Foundations of Physics*, vol. 16, pp. 507–531, 1986. [Online]. Available: http://dx.doi.org/10.1007/BF01886518

[70] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3, pp. 219–253, Apr. 1982. [Online]. Available: http://dx.doi.org/10.1007/BF01857727

[71] E. Fredkin and T. Toffoli, "Collision-based computing," *C.L*, pp. 47–81, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=644307.644311

[72] DSCH:, "Microwind and dsch information page." [Online]. Available: http://www.microwind.org/

[73] J.-B. Note and E. Rannaud, "From the bitstream to the netlist," in *Proceedings of the 16th international ACM/SIGDA symposium on field programmable gate arrays*. New York, NY, USA: ACM, 2008, pp. 264–264. [Online]. Available: http://doi.acm.org/10.1145/1344671.1344729

[74] S. D. Corey and A. T. Yang, "Automatic netlist extraction for measurement-based characterization of off-chip interconnect," in *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*. Washington, DC, USA: IEEE Computer Society, 1996, pp. 24–29.

[75] D. Thomas and P. Moorby, *The VERILOG Hardware Description Language*, 5th ed. Springer Publishing Company, Incorporated, 2008.

[76] P. Frey and D. O'Riordan, "Verilog-ams: Mixed-signal simulation and cross domain connect modules," in *Proceedings of the 2000 IEEE/ACM international workshop on Behavioral modeling and simulation*, ser. BMAS '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 103–. [Online]. Available: http://dl.acm.org/citation.cfm?id=555919.791445

[77] V. Sagdeo, *The Complete VERILOG Book.* Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[78] A. D. Vos, B. Desoete, A. Adamski, P. Pietrzak, M. Sibínski, and T. Widerski, "Design of reversible logic circuits by means of control gates," in *Proceedings of the 10th International Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation*, ser. PATMOS '00. London, UK: Springer-Verlag, 2000, pp. 255–264. [Online]. Available: http://dl.acm.org/citation.cfm?id=646948.712572

[79] M. Mohammadi and M. Eshghi, "On figures of merit in reversible and quantum logic designs," *Quantum Information Processing*, vol. 8, no. 4, pp. 297–318, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11128-009-0106-0

[80] I. Kuon, R. Tessier, and J. Rose, "FPGA architecture : survey and challenges," *Found. Trends Electron. Des. Autom.*, vol. 2, pp. 135–253, February 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1454695.1454696

[81] I. Kuon, A. Egier, and J. Rose, "Design, layout and verification of an FPGA using automated tools," in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays.* New York, NY, USA: ACM, 2005, pp. 215–226. [Online]. Available: http://doi.acm.org/10.1145/1046192.1046220

[82] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate arrays," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 277–282, Mar 1991.

[83] F. Sharmin, M. M. A. Polash, M. Shamsujjoha, L. Jamal, and H. M. Hasan Babu, "Design of a compact reversible random access memory," in *4th IEEE International Conference on Computer Science and Information Technology*, vol. 10, Chengdu, China, Jun. 2011, pp. 103–107.

[84] A. S. M. Sayem and M. Ueda, "Optimization of reversible sequential circuits," *Journal of Computing*, vol. 2, no. 6, pp. 208–214, Jun. 2010. [Online]. Available: http://arxiv.org/abs/1006.4570

[85] M.-L. Chuang and C.-Y. Wang, "Synthesis of reversible sequential elements," *J. Emerg. Technol. Comput. Syst.*, vol. 3, pp. 1–19, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1324177.1324181

[86] B. A. B. Sarif and M. Abd-El-Barr, "The use of multiple connected pseudo minterms in the synthesis of MVL functions," in *Proceedings of the 2009 39th International Symposium on Multiple-Valued Logic*, ser. ISMVL '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 145–150.

[87] S. J. Weber and K. Keutzer, "Using minimal minterms to represent programmability," in *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, Jersey City, NJ, USA, 2005, pp. 63–68. [Online]. Available: http://doi.acm.org/10.1145/1084834.1084854

[88] R. Wisniewski, *Synthesis of compositional microprogram control units for programmable devices, ISBN: 978-83-7481-293-1*, ser. Lecture Notes in Control and Computer Science, Vol. 14. Zielona Gra: University of Zielona Gra Press, 2009.

[89] S. Wee, J. Casper, N. Njoroge, Y. Tesylar, D. Ge, C. Kozyrakis, and K. Olukotun, "A practical FPGA-based framework for novel CMP research," in *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays.* New York, USA: ACM, 2007, pp. 116–125. [Online]. Available: http://doi.acm.org/10.1145/1216919.1216936

[90] W.-K. Mak and D. F. Wong, "Board-level multiterminal net routing for FPGA-based logic emulation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 2, no. 2, pp. 151–167, Apr. 1997. [Online]. Available: http://doi.acm.org/10.1145/253052.253136

[91] N.-C. Chou, L.-T. Liu, C.-K. Cheng, W.-J. Dai, and R. Lindelof, "Circuit partitioning for huge logic emulation systems," in *Proceedings of the 31st annual Design Automation Conference*, ser. DAC '94. New York, USA: ACM, 1994, pp. 244–249. [Online]. Available: http://doi.acm.org/10.1145/196244.196365

[92] D. Yin, D. Unnikrishnan, Y. Liao, L. Gao, and R. Tessier, "Customizing virtual networks with partial FPGA reconfiguration," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 1, pp. 125–132. [Online]. Available: http://doi.acm.org/10.1145/1925861.1925882

[93] M. Saldaña, L. Shannon, and P. Chow, "The routability of multiprocessor network topologies in FPGAs," in *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays.* New York, NY, USA: ACM, 2006, pp. 232–232. [Online]. Available: http://doi.acm.org/10.1145/1117201.1117253

[94] M. French, L. Wang, M. Wirthlin, and P. Graham, "Reducing power consumption of radiation mitigated designs for FPGAs," in *9th Annual International Conference on Military and Aerospace Programmable Logic Devices*, Sep. 2006.

[95] S. Coric, M. Leeser, E. Miller, and M. Trepanier, "Parallel-beam backprojection: an FPGA implementation optimized for medical imaging," in *Proceedings of the ACM/SIGDA tenth international symposium on Field-programmable gate arrays.* New York, USA: ACM, 2002, pp. 217–226. [Online]. Available: http://doi.acm.org/10.1145/503048.503080

[96] Y.-K. Choi, K. You, J. Choi, and W. Sung, "A real-time FPGA-based 20 000-word speech recognizer with optimized DRAM access," *Trans. Cir. Sys. Part I*, vol. 57, no. 8, pp. 2119–2131, Aug. 2010. [Online]. Available: http://dx.doi.org/10.1109/TCSI.2010.2041501

[97] E. C. Lin, K. Yu, R. A. Rutenbar, and T. Chen, "A 1000-word vocabulary, speaker-independent, continuous live-mode speech recognizer implemented in a single FPGA," in *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays.* New York, NY, USA: ACM, 2007, pp. 60–68. [Online]. Available: http://doi.acm.org/10.1145/1216919.1216928

[98] S. Brown, R. Francis, J. Rose, and Z. Vranesic, "Field-programmable gate arrays." *Springer/Kluwer Academic Publishers. ISBN: 978-0-7923-9248-4*, 1992.

[99] Voyiatzis, D. Gizopoulos, and A. Paschalis, "Accumulator-based test generation for robust sequential fault testing in DSP cores in near-optimal time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, pp. 1079–1086, 2005.

[100] Y. Lin, F. Li, and L. He, "Power modeling and architecture evaluation for FPGA with novel circuits for vdd programmability," in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays.* New York, USA: ACM, 2005, pp. 199–207. [Online]. Available: http://doi.acm.org/10.1145/1046192.1046218