2011

Design, Synthesis and Test of Reversible Circuits for Emerging Nanotechnologies

Himanshu Thapliyal
University of South Florida, himanshu.thapliyal@gmail.com

Design, Synthesis and Test of Reversible Circuits for Emerging Nanotechnologies

by

Himanshu Thapliyal

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Nagarajan Ranganathan, Ph.D.
Srinivas Katkoori, Ph.D.
Hao Zheng, Ph.D.
Andrew M. Hoff, Ph.D.
Kaushal Chari, Ph.D.

Date of Approval:
April 29, 2011

Keywords: Reversible Logic, Quantum Computing, Quantum Dot Cellular Automata,
Conservative Logic, Datapath Functional Units

DEDICATION

This dissertation is lovingly dedicated to my mother for all the sacrifices she made to ensure that I obtain the best education possible.

ACKNOWLEDGEMENTS

I would like to thank my doctoral advisor Dr. Nagarajan Ranganathan to allow me to work on the burgeoning area of reversible logic and emerging nanotechnologies. I am also thankful to him for facilitating his great guidance and active discussions whenever key issues had to be discussed about research or personal front. I firmly believe that it is because of Dr. Ranganathan guidance, encouragement and confidence in me that has transformed me into a better researcher. I would also like to thank Dr. Srinivas Katkoori, Dr. Hao Zheng, Dr. Andrew M. Hoff, and Dr. Kaushal Chari for taking the time to be in my doctoral committee and providing valuable suggestions to improve this manuscript. I am extremely grateful to the CSE department faculties, especially Dr. Dmitry Goldgof and Dr. Larry Hall, and CSE department staff for all their help and support. I owe my sincere thanks to Dr. Hamid R. Arabnia, Dr. M.B Srinivas, Dr. Vishal Verma and Dr. Mark Zwolinski for providing me inspirational thoughts to dedicatedly pursue the research and for being the driving force since my inception to the thrilling life of research.

I thank all my friends, my present and past colleagues in Computer Architecture and Nano VLSI Systems (CANS) Research Group, Soumyaroop, Ransford, Michael, Saurabh, Yue, Koustav, Mali, Upavan and others for a friendly and intellectually stimulating ambiance in the lab. I would like to especially thank Soumyaroop, Ransford, Michael and Saurabh for being wonderful colleagues as well as great friends. Finally, I would like to thank my mother, brothers and sisters, brothers-in-law and sisters-in-law for their love, support and encouragement that have always inspired me to strive hard for my goals irrespective of good and bad times. My late father had always been a constant source of inspiration and was the one who inspired me to always think big. I would also like to thank my friends Rajnish, Rohit, Gourav, Gaurav and others for their helping hand at times of need. I would also like to thank my fiancee, Apeksha, for her understanding and wonderful support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Reversible circuits are similar to conventional logic circuits except that they are built from reversible gates. In reversible gates, there is a unique, one-to-one mapping between the inputs and outputs, not the case with conventional logic. Also, reversible gates require constant ancilla inputs for reconfiguration of gate functions and garbage outputs that help in keeping reversibility. Reversible circuits hold promise in futuristic computing technologies like quantum computing, quantum dot cellular automata, DNA computing, optical computing, etc. Thus, it is important to minimize parameters such as ancilla and garbage bits, quantum cost and delay in the design of reversible circuits.

The first contribution of this dissertation is the design of a new reversible gate namely the TR gate (Thapliyal-Ranganathan) which has the unique structure that makes it ideal for the realization of arithmetic circuits such as adders, subtractors and comparators, efficient in terms of the parameters such as ancilla and garbage bits, quantum cost and delay. The second contribution is the development of design methodologies and a synthesis framework to synthesize reversible data path functional units, such as binary and BCD adders, subtractors, adder-subtractors and binary comparators. The objective behind the proposed design methodologies is to synthesize arithmetic and logic functional units optimizing key metrics such as ancilla inputs, garbage outputs, quantum cost and delay. A library of reversible gates such as the Fredkin gate, the Toffoli gate, the TR gate, etc. was developed by coding in Verilog for use during synthesis. The third contribution of this dissertation is the set of methodologies for the design of reversible sequential circuits such as reversible latches, flip-flops and shift registers. The reversible designs of asynchronous set/reset D latch and the D flip-flop are attempted for the first time. It is shown that the designs are optimal in terms of number of garbage outputs while exploring the best possible values for quantum cost and delay.

The other important contributions of this dissertation are the applications of reversible logic as well as a special class of reversible logic called conservative reversible logic towards concurrent (online) and offline testing of single as well as multiple faults in traditional and reversible nanoscale VLSI circuits, based on emerging nanotechnologies such as QCA, quantum computing, etc. Nanoelectronic devices tend to have high permanent and transient faults and thus are susceptible to high error rates. Specific contributions include (i) concurrently testable sequential circuits for molecular QCA based on reversible logic, (ii) concurrently testable QCA-based FPGA, (iii) design of self checking conservative logic gates for QCA, (iv) concurrent multiple error detection in emerging nanotechnologies using reversible logic, (v) two-vectors, all 0s and all 1s, testable reversible sequential circuits.

CHAPTER 1

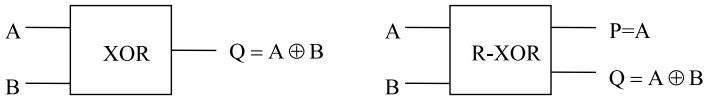
INTRODUCTION

Among the emerging computing paradigms, reversible logic appears to be promising due to its wide applications in emerging technologies. Some of the emerging nanotechnologies having applications of reversible logic are quantum computing, quantum dot cellular automata, optical computing, Spintronics, DNA computing, molecular computing and also in power-efficient nanocomputing, etc [5–13]. Researchers have also investigated optical computing based non-dissipative reversible gates [14, 15]. Reversible circuits are those circuits that do not lose information during computation and reversible computation in a system can be performed only when the system comprises of reversible gates. These circuits can generate unique output vector from each input vector, and vice versa, that is, there is a one-to-one mapping between the input and the output vectors. As a fundamental contribution in [16], Landauer has shown that during irreversible computation 1 bit of information lost results in KTln2 Joules of energy dissipation. Bennett in another seminal contribution [17], proved that this KTln2 joules of energy dissipation will not occur if computation is performed in a reversible manner. An $N \times N$ (N inputs and N outputs) reversible gate can be represented as

$$I_v = I_1, I_2, I_3, I_4, ..., I_N \tag{1.1}$$

$$O_v = O_1, O_2, O_3, O_4, ..., O_N \tag{1.2}$$

where I_v and O_v represent input and output vectors, respectively. Classical logic gates are irreversible since input vector states cannot be uniquely reconstructed from the output vector states. As an example, considering conventional irreversible XOR gate shown in Fig.1.1(a), it is obvious from the truth table illustrated in Table 1.1.a that a unique input vector cannot be constructed from the output vector. This is because for output 0, there are two input vectors AB = (00, 11) that give rise



(a) Conventional XOR gate (b) Reversible XOR gate

Figure 1.1. Conventional XOR and reversible XOR gates

to it. Considering Fig.1.1(b), a reversible XOR gate, it is evident from the truth table illustrated in Table 1.1.b that a unique input vector can be constructed for every output vector.

Table 1.1. Truth tables for conventional XOR and reversible XOR gates

(a). XOR gate

A	B		Q
0	0		0
0	1		1
1	0		1
1	1		0

(b). R-XOR gate

A	B		P	Q
0	0		0	0
0	1		0	1
1	0		1	1
1	1		1	0

1.1 Motivation

A quantum computer will be viewed as a quantum network (or a family of quantum networks) composed of quantum logic gates; each gate performing an elementary unitary operation on one, two or more two-state quantum systems called qubits. Each qubit represents an elementary unit of information; corresponding to the classical bit values 0 and 1. Any unitary operation is reversible and hence quantum networks must be built from reversible logic components [18]. Quantum computers is considered one of the most promising computing paradigm in which computation can be performed at an atomic level make it feasible to work beyond the existing limits of the semiconductor industry. The algorithms for quantum computers can potentially solve NP-complete problems that cannot be solved in traditional computers. This could mean that previously unsolvable problems can be solved once quatum computers are realized in critical fields such as biotechnology, nanomedicine, secure computing, etc. Thus, the feasibility of reversible logic circuits could critically impact the realization of quatum computing.

Further, when a computation is performed in an irreversible manner, 1 bit worth of lost logical information always leads to at least kTln2 amount of physical energy dissipation and is called as the von Neumann-Landauer (VNL) principle, after its discoverers [16]. From thermodynamic point of view, in order to avoid this limit, Bennett showed that kTln2 energy dissipation would not occur, if a computation is carried out in a reversible way [17]. Thus, from thermodynamic considerations, a firm lower limit on dissipation of $E_{diss} = KTln2 \approx 18meV$ (in room-temperature environment) is a necessity for conventional (irreversible) logic, even if reliability issues could be ignored. Reversible logic can be useful to design non-dissipative circuits if the physical implementation of the logic is also physically reversible. CMOS cannot be considered as a practical implementation platform as CMOS is not physically reversible. In modern CMOS technology, voltage-coded logic signals have an energy of $E_{sig} = (1/2)CV^2$, and whenever the node voltage is changed, it leads to dissipation of this energy and is orders of magnitude higher than the KTln2 factor. In contrast, there are emerging nanotechnologies such as Quantum Dot Cellular automata (QCA) computing, Optical Computing, and Superconductor Flux Logic (SFL) family, etc., where the energy dissipated due to information destruction will be a significant factor of the overall heat dissipation of the system [8, 19–22]. Thus, one of the primary motivation for adopting reversible logic lies in the fact that it can provide a logic design methodology for designing ultra-low power circuits beyond KTln2 limit for those emerging nanotechnologies in which the energy dissipated due to information destruction will be a significant factor of the overall heat dissipation. For example, in new Superconductor Flux Logic (SFL) family based on nSQUID gates, the energy dissipation in conventional logically irreversible architectures is close to few KTln2 per logic operation. By employing reversible logic, the energy dissipation per nSQUID gate per bit measured, at 4 K temperature is already below the thermodynamic threshold limit of KTln2 [21].

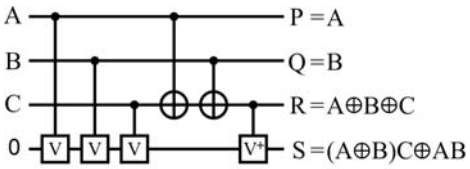
The performance of any computing system is given by the number of useful operations per unit time. Thus, performance can be written as $R = \frac{N_{ops}}{t} = \frac{N_{ops}}{E_{diss}} \times \frac{E_{diss}}{t} = F_E \times P_{diss}$ [10, 23] where R = performance, N_{ops} = number of useful operations performed during a job, t = total elapsed time to perform the job, E_{diss} = energy dissipated during the job, $F_E = N_{ops}/E_{diss}$ = energy efficiency, $P_{diss} = E_{diss}/t$ = average power dissipation during the job. It can be clearly understood

from the above equation that improving system performance requires increasing the average energy efficiency F_E of useful operations which can be done by minimizing the energy dissipated during the job. Thus, in emerging nanotechnologies in which the signal energy is few orders of magnitude higher than the KTln2 limit, further improvements can only be gained by going beyond the KTln2 limit which is only possible by adopting the reversible logic. This shows that reversible logic can be beneficial towards raising computer performance in emerging nanotechnologies to theoretically approach infinity.

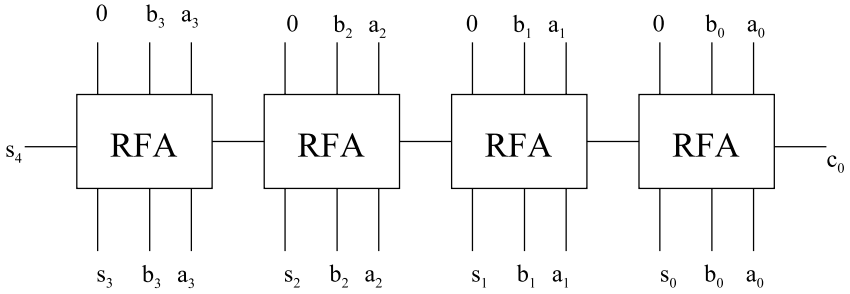
Reversible logic could also help to potentially recover and retain a fraction of the signal energy that can be reused for subsequent operations by doing the computation using the forward path and then undoing the computation using the backward path. These concepts have been implemented in CMOS to save significant amount of energy dissipation even close to 90% using the concepts such as reversible energy recovery logic (RERL) etc [24, 25]. Reversible logic have also promising applications in online and offline testing of faults. For example, it has been proved by researchers that for reversible logic circuits, the test set that detects all single stuck-at faults can also detect multiple stuck-at faults [26]. In summary, the above arguments constitute our main motivations to pursue further research into design, synthesis and test of reversible logic circuits having application in emerging nanotechnologies.

Several important metrics need to be considered in the design of reversible circuits the importance of which needs to be discussed. The constant input in the reversible quantum circuit is called the ancilla input qubit (ancilla input bit), while the garbage output refers to the output which exists in the circuit just to maintain one-to-one mapping but is not a primary or a useful output. Quantum computers of many qubits are extremely difficult to realize thus the number of qubits in the quantum circuits needs to be minimized. The importance of minimizing the garbage and ancilla bits could be best illustrated with an example. Suppose there is a need to realize a 6 inputs and 4 outputs function in a quantum computer and the design requires 6 additional garbage outputs (that is have the 4 constant inputs). This will result in a reversible function having 10-inputs and 10 outputs. Suppose the best realizable quantum computer due to technology limitations had only 7 qubits, thus we will not able implement the required design. This sets the major objective of optimizing the

number of ancilla input qubits and the number of the garbage outputs in the reversible logic based quantum circuits. Additionally, there are number of implementation platforms that are being explored for physical implementations for qubits and quantum gates. Some of these implementation platforms are trapped ions, spintronics, superconducting circuits, linear optics/photonics, quantum dots, etc, [5, 11, 21, 27–29]. There is no clear winner and it is not sure which implementation technology will be the future of the quantum computers. *Thus there is a need of technology independent design and synthesis of reversible logic circuits that are applicable to quantum computing*. The reversible circuit has other important parameters of quantum cost and delay which need to be optimized. The quantum cost of a design is the number of 1x1 and 2x2 reversible gates used in its design, thus can be considered equivalent to number of transistors needed in a conventional CMOS design. The delay of a reversible circuit can be computed by calculating its logical depth when it is designed from smaller 1x1 and 2x2 reversible gates. A synthesis framework in which a single parameter is optimized is inadequate since optimizing one parameter often could be resulting in the degradation of other important parameters. Further, the general synthesis methods proposed in the existing literature target combinational logic synthesis in general and are not suitable for synthesis of reversible arithmetic units as well as reversible sequential logic synthesis. This is because in arithmetic units such as adders, subtractors, comparators, etc, the choice of the hardware algorithm or the architecture has an impact on the performance and efficiency of the circuit. For example, the design of 1 bit reversible full adder having inputs A, B and carry input C is shown in Fig.1.2(a) which shows that the design of a 1 bit reversible full adder needs 1 ancilla input having constant value as 0. The sum and carry outputs are produced at outputs R and S, respectively (the regenerated inputs are not considered as garbage outputs). Thus, as shown in Fig.1.2(b) for a 4 bit reversible adder, the design of a n bit reversible adder based on the conventional ripple carry approach of cascading will need n ancilla inputs. This shows that the use of the conventional approach of design and synthesis to reversible logic circuits may result in significant overhead in terms of parameters such as ancilla and garbage bits. Further, in reversible logic based circuit design, parameters such as ancilla inputs, garbage outputs, quantum cost and delay need to be optimized that are completely different from the traditional parameters of speed, power and chip area used in conventional com-



(a) 1 bit reversible adder



(b) 4 bit reversible adder using conventional ripple carry approach

Figure 1.2. Circuit generation of reversible adder using conventional design methodology

puting. Thus, there is a need of research towards developing new design and synthesis methods for realization of reversible arithmetic circuits and a synthesis framework in which multiple parameters can be optimized.

Further, in the design of reversible logic circuits, research was primarily limited to the design of combinational circuits due to the convention that feedback is not allowed in reversible computing. However in one of the fundamental papers, Toffoli has shown that feedback can be allowed in reversible computing. According to Toffoli "a sequential network is reversible if its combinational part (i.e., the combinational network obtained by deleting the delay elements and thus breaking the corresponding arcs) is reversible". Further it is shown that in order to construct a reversible finite automata one can construct a reversible realization of its transition function and use it as a combinational part of the desired sequential circuit [30]. Thus, there is a need of research towards the design and synthesis of reversible sequential circuits to design complete reversible systems such as general purpose processor as well as application specific digital signal processors. The advancement in design and synthesis of reversible sequential can also be beneficial towards the design of zero power sequential machines [31]. Further, the emerging nanotechnologies such as quantum dot

cellular automata (QCA) and quantum computing are susceptible to high error rates. The nanoelectronic devices have significant increase in permanent as well as transient faults. The primary reason for the permanent defects at the nanoscale level is the small scale of devices and the bottom-up self-assembly process. Further, due to the nanometerscale of devices and small energy difference between ground and excited states, nanoelectronic transistors tend to be highly sensitive to environmental influences, such as temperature, cosmic ray particles and background noise-producing transient faults [6, 32, 33]. Thus, there is need of advancement of novel techniques and design methodologies that can address the concurrent (online) as well as offline testing of faults in circuits based on emerging nanotechnologies.

In summary, the motivations for this dissertation are: (i) to explore the design and synthesis of reversible logic circuits considering metrics of ancilla inputs, garbage outputs, quantum cost and the delay, and (ii) to explore the benefits of reversible logic towards concurrent and offline testing of faults in circuits based on emerging nanotechnologies.

1.2 Contributions and Significance

In reversible logic design and synthesis there are a few important points that need to be kept in mind for efficient and optimal reversible design,

- Minimize the quantum cost of the design. The quantum cost of a design is the number of 1x1 and 2x2 reversible gates used in its design.
- Minimize garbage outputs by using as many outputs of every gate as possible.
- Minimize delay by reducing the logic depth.
- Avoid constant inputs to gates unless necessary. The constant inputs are known as ancilla inputs.
- Avoid a fan-out of more than one, as each fan-out greater than one requires an additional copying circuit (fan-out is not allowed in reversible logic).

Thus, in the proposed dissertation we have designed reversible logic circuits and also proposed design methodology to address the above important points. The dissertation has the following contributions toward the design and synthesis of reversible logic circuits.

- *The first contribution of this dissertation is the design of a new reversible gate namely the TR gate (Thapliyal-Ranganathan) that has the unique structure by which it can realize the Boolean functions $A \cdot \bar{B} \oplus C$ and A $\oplus$ B with only one gate. Further, it can implement the functions such as $A \cdot \bar{B}$ when its input C is tied to 0. These properties of TR gate make it very useful in designing the reversible arithmetic circuits such as adders, subtractors and comparators, efficient in terms of the parameters such as ancilla and garbage bits, quantum cost and delay.*

- *The second contribution is the development of design methodologies and a framework to synthesize reversible data path functional units, such as binary and BCD adders, subtractors, adder-subtractors, and binary comparators. The objective behind the proposed design methodologies is to synthesize arithmetic and logic functional units optimizing key metrics such as ancilla inputs, garbage outputs, quantum cost and delay. A library of reversible gates such as the Fredkin gate, the Toffoli gate, the TR gate, etc. was developed by coding in Verilog Hardware Description Language for use during synthesis. .*

- *The third contribution of this dissertation is the set of methodologies for the design of reversible sequential circuits such as reversible latches, flip-flops and shift registers. The reversible designs of asynchronous set/reset D latch and the D flip-flop are attempted for the first time. It is shown that the designs are optimal in terms of number of garbage outputs while exploring the best possible values for quantum cost and delay.*

Further, the other major contributions are toward application of reversible logic as well as a special class of reversible logic called conservative reversible logic towards concurrent (online) and offline testing of single as well as multiple faults in traditional and reversible nanoscale VLSI circuits, based on emerging nanotechnologies such as QCA, quantum computing, etc. Nanoelectronic

devices tend to have high permanent and transient faults and thus are susceptible to high error rates. Specific contributions include:

- *Concurrently testable sequential circuits for molecular QCA based on reversible logic.*
- *Concurrently testable QCA-based FPGA.*
- *Design of self checking conservative logic gates for QCA..*
- *Concurrent multiple error detection in emerging nanotechnologies using reversible logic.*
- *Two vectors, all 0s and all 1s, testable reversible sequential circuits.*

The proposed contributions toward concurrent (online) and offline testing of faults illustrate that reversible logic can be of great advantage in reducing the cost of test pattern generation process for nanocircuits. Reversible circuits can greatly reduce the overall manufacturing cost by reducing the cost involved in testing. The proposed work designs the exhaustively testable reversible sequential circuits for unidirectional stuck-at faults, as well as, single/missing additional cell defect by using only two test vectors, all 0s and all 1s, thereby eliminating the need for any type of scan-path access to internal memory cells. We expect that the proposed work will encourage a new paradigm of fault testing of nanocircuits based on reversible logic.

1.3 Outline of Dissertation

The remainder of this dissertation is organized as follows: Chapter 2 describes the background and a comprehensive literature survey pertaining to our research. In Chapter 3, we design a new reversible gate namely the TR gate (Thapliyal-Ranganathan) that has the unique structure by which it can realize the reversible arithmetic functions efficiently compared to the existing reversible gates. In Chapter 4, we present class of new designs and design methodologies for reversible binary adders, subtractors, adder-subtractors and BCD adder circuits. The proposed designs are primarily optimized for the number of ancilla inputs and the number of garbage outputs, and they are designed to give the best possible values for quantum cost and delay. In chapter 5, we introduced novel designs of reversible sequential circuits which minimize the quantum cost, delay and number of garbage

outputs, and are more efficient compared to existing designs. The sequential circuits considered in this work are reversible designs of latches, such as D latch, T latch, JK latch, SR latch and their corresponding master-slave flip-flops. Complex reversible sequential circuits such as reversible universal shift register are also discussed. The simulation flow based on Verilog HDL, which is used to simulate the reversible sequential circuits is also illustrated. In Chapter 6, we present the application of reversible logic as well as a special class of reversible logic called conservative reversible logic towards the concurrent testing of faults in circuits based on emerging nanotechnologies such as QCA computing. A new self-checking conservative QCA gate is also presented. In Chapter 7, we present two vectors testable reversible sequential circuits. Reversible sequential circuits such as the latches, master slave flip-flops, double edge triggered flip-flops based on conservative reversible logic that can be tested by only two test vectors, all 0s and all 1s, to detect any unidirectional stuck-at faults, as well as, single missing/additional cell defect are designed. A new conservative logic gate called Multiplexer Conservative QCA gate (MX-cqca) is presented that is not reversible in nature but has similar properties as the Fredkin gate of working as 2:1 multiplexer and surpasses the Fredkin gate in terms of complexity(the number of majority voter), speed and area. The concluding remarks and the suggested future work in terms of extensions to the problems addressed in this dissertation, and other ideas for further refinements are given in Chapter 8.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Background on Reversible Logic

Several 3x3 reversible gates such as the Fredkin gate [34], the Toffoli gate [30], and the Peres gate [35] have been reported in the literature. Each reversible gate has a cost associated with it called the quantum cost [36]. The quantum cost of a reversible gate is the number of 1x1 and 2x2 reversible gates or quantum logic gates required in its design. The quantum costs of all reversible 1x1 and 2x2 gates are taken as unity [36–38]. Any reversible gate can be realized using the 1x1 NOT gate, and 2x2 reversible gates such as Controlled-V and Controlled-V^+ (V is a square-root-of NOT gate and V^+ is its hermitian) and the Feynman gate which is also known as the Controlled NOT gate (CNOT). Thus, in simple terms, the quantum cost of a reversible gate can be calculated by counting the numbers of NOT, Controlled-V, Controlled-V^+ and CNOT gates required in its implementation. A few cases as exceptions have been pointed out in [37].

2.1.1 The NOT Gate

A NOT gate is a 1x1 gate represented as shown in Fig. 2.1(a). Since it is a 1x1 gate, its quantum cost is unity.

2.1.2 The Controlled-V and Controlled-V^+ Gates

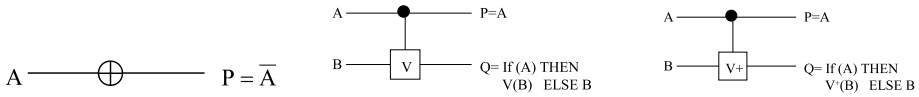
The controlled-V gate is shown in Fig. 2.1(b). In the controlled-V gate, when the control signal A=0 then the qubit B will pass through the controlled part unchanged, i.e., we will have Q=B. When A=1 then the unitary operation $V = \frac{i+1}{2}\begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$ is applied to the input B, i.e., Q=V(B). The controlled-V^+ gate is shown in Fig. 2.1(c). In the controlled-V^+ gate when the control signal A=0

then the qubit B will pass through the controlled part unchanged, i.e., we will have Q=B. When A=1 then the unitary operation $V^+ = V^{-1}$ is applied to the input B, i.e., Q=V^+(B).

The V and V^+ quantum gates have the following properties:

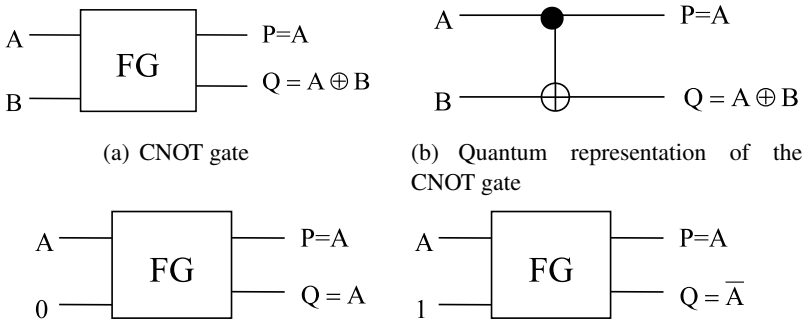
$$V \times V = NOT$$
$$V \times V^+ = V^+ \times V = I$$
$$V^+ \times V^+ = NOT$$

The properties above show that when two V gates are in series they will behave as a NOT gate. Similarly, two V^+ gates in series also function as a NOT gate. A V gate in series with V^+ gate, and vice versa, is an identity. For more details of the V and V^+ gates, the reader is referred to [5,37,38].



(a) NOT gate (b) Controlled-V gate (c) Controlled-V^+ gate

Figure 2.1. NOT, Controlled-V and Controlled-V^+ gates



(a) CNOT gate

(b) Quantum representation of the CNOT gate

(c) Feynman gate for avoiding the fanout

(d) Feynman gate for generating the complement of a signal

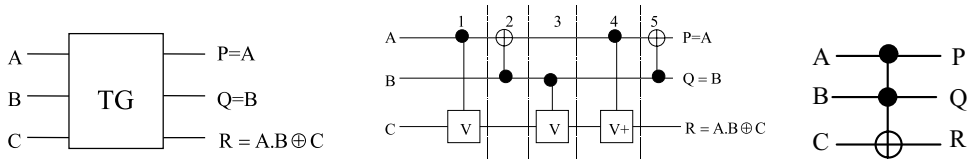
Figure 2.2. CNOT gate, its quantum implementation and its useful properties

2.1.3 The Feynman Gate (CNOT Gate)

The Feynman gate (FG) or the Controlled-NOT gate(CNOT) is a 2 inputs 2 outputs reversible gate having the mapping (A, B) to (P=A, Q=A⊕B) where A, B are the inputs and P, Q are the outputs, respectively. Since it is a 2x2 gate, it has a quantum cost of 1. Figures 2.2(a) and 2.2(b) show the block diagrams and quantum representation of the Feynman gate. The Feynman gate can be used for copying the signal thus avoiding the fanout problem in reversible logic as shown in Fig.2.2(c). Further, it can be also be used for generating the complement of a signal as shown in Fig.2.2(d).

2.1.4 The Toffoli Gate

The Toffoli Gate (TG) is a 3x3 two-through reversible gate as shown in Fig. 2.3(a). Two-through means two of its outputs are the same as the inputs with the mapping (A, B, C) to (P=A, Q=B, R=A·B⊕ C), where A, B, C are inputs and P, Q, R are outputs, respectively. The Toffoli gate is one of the most popular reversible gates and has the quantum cost of 5 as shown in Fig. 2.3(b) [30]. The quantum cost of Toffoli gate is 5 as it needs 2V gates, 1 V^+ gate and 2 CNOT gates to implement it.



(a) Toffoli gate

(b) Quantum implementation of Toffoli gate

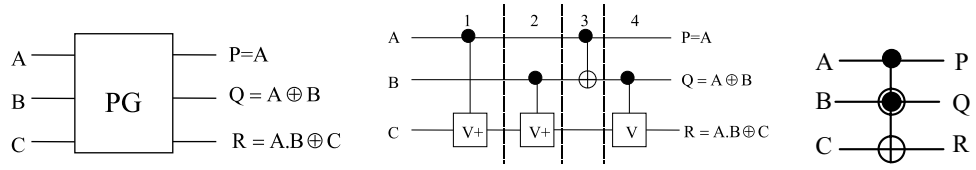
(c) Graphical notation of Toffoli gate

Figure 2.3. Toffoli gate and its quantum implementation

2.1.5 The Peres Gate

The Peres gate is a 3 inputs 3 outputs (3x3) reversible gate having the mapping (A, B, C) to (P=A, Q=A⊕B, R= A·B⊕C), where A, B, C are the inputs and P, Q, R are the outputs, respectively

[35]. Figure 2.4(a) shows the Peres gate and Fig. 2.4(b) shows the quantum implementation of the Peres gate (PG) with quantum cost of 4 [37]. The quantum cost of Peres gate is 4 since it requires 2 V^+ gates, 1 V gate and 1 CNOT gate in its design. In the existing literature, among the 3x3 reversible gates, the Peres gate has the minimum quantum cost. The graphical notation of the Peres gate is shown in Fig.2.4(c).



(a) Peres gate

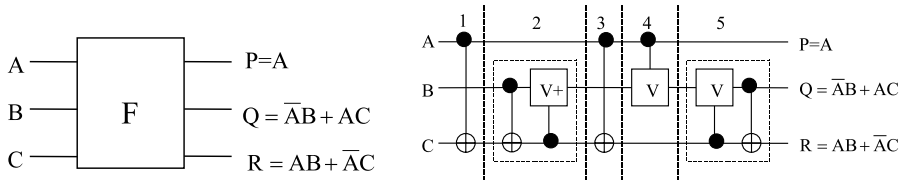
(b) Quantum implementation of Peres gate

(c) Graphical notation of Peres gate

Figure 2.4. Peres gate and its quantum implementation

2.1.6 Fredkin Gate

Fredkin gate is a (3x3) conservative reversible gate, having the mapping (A, B, C) to (P=A, Q=A'B+AC, R=AB+A'C), where A, B, C are the inputs and P, Q, R are the outputs, respectively [34]. It is called a 3x3 gate because it has three inputs and three outputs. Figure 2.5(a) shows the Fredkin gate and Figure 2.5(b) shows its quantum implementation with quantum cost of 5 [37]. Please note that each dotted rectangles in Fig. 2.5(b) is equivalent to a 2x2 Feynman gate and so the quantum cost of each dotted rectangle is 1 [36]. This assumption is used in [37] for calculation of the quantum cost of the Fredkin gate. Hence Fredkin gate cost consists of 2 dotted rectangles, 1 Controlled-V gate and 2 CNOT gates resulting in its quantum cost as 5. For calculating the quantum cost of the reversible sequential circuits, we have also followed the assumption of [36] and used the quantum cost of the Fredkin gate as 5 in our calculations.



(a) Fredkin gate (b) Quantum representation of the Fredkin gate

Figure 2.5. Fredkin gate and its quantum implementation

2.1.7 Delays of the Reversible Gates

In this section, we discuss the importance of delay as a parameter in the design of reversible sequential circuits. In much of the earlier works such as [39, 40] the delay of 2x2, 3x3 and 4x4 reversible gates are considered to be of unit delay irrespective of their computational complexity. The unit delay model does not aid fair comparisons as delay will vary according to the complexity of a reversible gate. In our delay calculations, we use the logical depth as measure of the delay [41]. The delay of each 1x1 gate and 2x2 reversible gate is taken as unit delay called Δ. Any 3x3 reversible gate can be designed from 1x1 reversible gates and 2x2 reversible gates, such as CNOT gate, Controlled-V and Controlled-V+ gates. Thus, the delay of a 3x3 reversible gate can be computed by calculating its logical depth when it is designed from smaller 1x1 and 2x2 reversible gates.

The Toffoli gate, the Peres gate and the Fredkin gate are the three major 3x3 reversible gates used for designing the reversible circuits, thus in this section we calculate the delays of the Toffoli gate, the Peres gate and the Fredkin gate. Figure 2.3(b) shows the logic depth in the quantum implementation of the Toffoli gate. Thus, it can be seen that the Toffoli gate has the delay of 5 Δ. Each 2x2 reversible gate in the logic depth contributes to 1 Δ delay. Similarly, the Peres gate and the Fredkin gate shown in Fig. 2.4(b) and Fig. 2.5(b), respectively, have the logic depth of 4 and 5, respectively, that results in their delay as 4 Δ and 5 Δ, respectively.

For delay calculations in the reversible sequential circuits such as reversible latches, we determine the reversible gates in the critical path, replace the reversible gates with corresponding quantum gates, now sum up the corresponding delay values for the quantum gates to arrive at the

path delay. The path or paths with the maximum path delay value will constitute the critical path. It should be noted that the path delay corresponds to the logical depth of the reversible circuit. In other words, the logical depth corresponds to the number of quantum gates in the path. Thus, the quantum cost of the circuit is the total number of quantum gates in the circuit while the delay is the number of quantum gates in the critical path of the circuit and thus both will be different values.

2.2 Related Work on Reversible Logic

The research on reversible logic is expanding towards both design and synthesis. Several researchers have been exploring techniques for synthesis of reversible logic circuits and many interesting contibutions have been made [42–49]. The synthesis of reversible circuits that employ a minimum number of gates and contain no redundant input-output line-pairs (temporary storage channels) is investigated in [43]; Researchers in [42] have used the positive-polarity Reed-Muller expansion of a reversible function to synthesize the function as a network of Toffoli gates; The work in [44] has illustrated the number of garbage outputs that must be added to a multiple output function to make it reversible. Further a new reversible design method that uses the minimum number of garbage outputs is also proposed; The authors in [45] investigate the problem of optimally synthesizing 4-bit reversible circuits using an enhanced bi-directional synthesis approach. The researchers have also addressed the optimization of reversible logic circuits from the perspective of quantum cost and the number of garbage outputs. Recently, in [50, 51], interesting contributions have been made towards deriving exact minimal elementary quantum gate realization of reversible combinational circuits. Thus, in synthesis of reversible logic circuits, the optimization in terms of number of ancilla input bits and also the delay are not yet addressed except in the recent work [52] which discusses about the post synthesis method for reducing the number of lines (qubits) in the reversible circuits.

Reversible arithmetic units such as adders, subtractors, multipliers,dividers which form the essential components of a computing system have also been designed in binary as well as ternary logic as in [40, 53–55] in which parameters such as the number of reversible gates, number of garbage outputs, quantum cost, number of transistors, etc are considered for optimization. In [56],

researchers have designed the quantum ripple carry adder having no input carry with one ancilla input bit. In [57, 58], researchers have investigated new designs for the quantum ripple carry adder with no ancilla input bit as well as improved delay. The work in [59] present the measurement based design of a carry look-ahead adder while in [60], the concept of arithmetic on a distributed-memory quantum multicomputer is introduced. A comprehensive survey of quantum arithmetic circuits is presented in [61]. The design of BCD adders and subtractors have also been attempted in [62], and in [4]. The researchers have investigated the design of BCD adders and subtractors in which parameters such as the number of reversible gates, number of garbage outputs, quantum cost, number of transistors, etc are considered for optimization [4, 39, 62–65].

The design of reversible sequential circuits was first introduced in [34], in which the design of the JK latch was discussed. Later, the design of the RS latch was introduced in [66]. The design uses two cross-coupled reversible NOR gates as used in conventional logic for designing the RS latch. The design was clock less in nature, i.e., there was no enable signal. The NOR gates were designed from the reversible Fredkin gate. The work was limited to the design of RS latch only. In [67], the authors introduced reversible latches such as D latch, T latch, etc., along with their corresponding flip-flops. The flip-flops were designed using master-slave strategy in which one reversible latch works as a master latch and the other works as a slave latch. The work in [67] was the first to discuss the complete design of reversible sequential circuits, however their implementation was expensive as each irreversible logic gate was mapped to its corresponding reversible counterpart. In [68], the reversible RS latch that avoids the fanout problem in the design proposed by [66] was proposed. In [68], the author had primarily focussed on the RS latch. All the other latches were designed as the sub-units from reversible RS latch as a part of master-slave flip-flops. In [69], the authors proposed the designs of reversible latches and flip flops. The proposed designs were shown to be better than the designs presented in [68] in terms of the number of reversible gates and garbage outputs. The transistor implementations of the reversible gates were also addressed. Recently, in [70] a more detailed analysis of the reversible RS latch design is presented. In [71], a discussion on how a universal reversible computer could be constructed from reversible logic elements and reversible sequential machines is given from the standpoint of computation theory, but no actual hardware

design was presented. Other designs of reversible sequential circuits are given in [72, 73]. Recently in [74], all reversible latch(except the SR latch) and their corresponding master-slave flip-flops were proposed that are better than the designs in [67] and [68]. The parameters used in comparison were the number of reversible gates and garbage outputs. Thus, from a careful survey of the existing works on reversible sequential circuits, it can be concluded that most of these works considered the optimization of number of reversible gates and garbage outputs, while ignoring the important parameters of quantum cost and delay.

Any nanotechnology having applications of reversible logic such as based on nano-CMOS devices, NMR based quantum computing, or low power molecular QCA computing, all are susceptible to high error rates due to permanent and transient faults. This has attracted the attention of researchers towards testing of reversible logic circuits. In [26], it has been proved that for reversible logic circuits, the test set that detects all single stuck-at faults can also detect multiple stuck-at faults. In [75], four fault models for reversible circuits, viz., single missing gate fault, the repeated-gate fault, the multiple missing gate fault and the partial missing-gate fault are proposed based on ion-trap quantum computing at logical level. In [76], a new fault model called crosspoint fault model is proposed along with the ATPG method. In [77], a universal test set is proposed for detection of missing-gate faults in reversible circuits. In [78], a DFT methodology for detecting bridging faults in reversible logic circuits is proposed. Recently, the design of reversible finite field arithmetic circuits with error detection is also proposed [79]. An online testing methodology for reversible circuits using a combination of R1 gate along with R2 gate (a 4*4 Feynman Gate) is proposed in [80] while in [81] an automatic conversion of any given reversible circuit into an online testable circuit that can detect online any single-bit errors, including soft errors in the logic blocks is presented. The online testing methodology of reversible logic circuits are also addressed in [82]. Further, it has been proved in [83–85] that the combinational circuits based on reversible conservative logic gates outperforms all the circuits implemented in classical gates in the area of testing. Any combinational circuit based on conservative logic gates can be tested for classical unidirectional stuck-at faults using only two test vectors, all 0s and all 1s.

2.3 Dissertation Context in Light of Past Works on Reversible Logic

Among the existing reversible gates, there is no such reversible gate that can help in mapping the equations $A \cdot \bar{B} \oplus C$ and A $\oplus$ B, singly. The equations $A \cdot \bar{B} \oplus C$ and A $\oplus$ B are useful in mapping many arithmetic functions. As an example, the half subtractor performs A-B operation. The output functions of the half subtractor are $Borrow = \bar{A} \cdot B$; $Difference = A \oplus B$. This dissertation advances the field by proposing a new reversible gate called the TR gate that realizes the functions $A \cdot \bar{B} \oplus C$ and A $\oplus$ B, singly. Using the TR gate, the half subtractor equations can be mapped on a single gate. Further, to the best of our knowledge researchers have not yet addressed the design of the binary and BCD arithmetic units primarily focusing on optimizing the number of ancilla input bits and the garbage outputs along with optimizing the parameters of the quantum cost and the delay. In this work, we present a class of new designs for reversible binary and BCD adder circuits. The proposed designs are primarily optimized for the number of ancilla inputs and the number of garbage outputs and are designed for possible best values for the quantum cost and delay.

In the existing works on reversible sequential circuits, the primary design focus has been on optimizing the number of reversible gates and the garbage outputs. The number of reversible gates is not a good metric of optimization as each reversible gate is of different type and computational complexity, and thus will have a different quantum cost and delay. The computational complexity of a reversible gate can be represented by its quantum cost. Further, delay constitutes an important metric, which has not been addressed in prior works on reversible sequential circuits as a design metric to be optimized. In this work, we present novel designs of reversible sequential circuits that are optimized in terms of quantum cost, delay and the garbage outputs. The optimized designs of several reversible sequential circuits are presented. We also introduce a novel strategy of cascading a Fredkin gate at the outputs of a reversible latch to realize the designs of the Fredkin gate based asynchronous set/reset D latch and the master-slave D flip-flop.

Further, to the best of our knowledge the offline testing of faults in reversible sequential circuits is not addressed in the literature except in [74]. In this work, we advance the state of the art by presenting the design of reversible sequential circuits based on conservative reversible Fredkin gate

that can be tested by only two test vectors, all 0s and all 1s, for any unidirectional stuck-at-faults. Further, the approach of fault testing based on conservative logic is extended towards the design of non-reversible sequential circuits based on a new conservative logic gate called multiplexer conservative QCA gate (Mx-cqca). The design of testable reversible double edge triggered flip-flop is discussed for the first time in the literature.

2.4 Background on Quantum Dot Cellular Automata (QCA) Computing

CMOS technology is reaching closer to the limits beyond which the feature size cannot be downscaled further without compromising the proper functioning of the device. CMOS devices suffer from thermal effects, as they have to discharge all the stored energy when flipping from 1 to 0. Quantum dot cellular automata (QCA) is one of the emerging nanotechnologies which make it possible to achieve circuit densities and clock frequencies beyond the limits of existing CMOS technology. QCA has significant advantage in terms of power dissipation, as it does not have to dissipate all its signal energy during transition. Hence, QCA is considered as one of the promising technologies to achieve the thermodynamic limit of computation [86, 87].

2.4.1 Basic QCA Cell

A QCA cell is a coupled dot system in which four dots are at the vertices of a square. The cell has two extra electrons that occupy the diagonals within the cell due to electrostatic repulsion. The cell polarization P measures the charge distribution along diagonal axes and is given by equation 1 (here Pi denotes the electronic charge at dot i). When electrons are in dots 1 and 3, P = -1 (Logic '0') and when electrons in dots 2 and 4, P = +1 (Logic '1') [86]. Figures 2.6(a) and 2.6(b) show the 4 quantum dots in a QCA cell, and the implementation of logic '0' and logic '1' in a QCA cell, respectively.

$$P = \frac{(P2 + P4) - (P1 + P3)}{P1 + P2 + P3 + P4} \tag{2.1}$$

The basic QCA device is the majority voter or majority gate, and has the output function as F=AB+BC+AC, where F is the majority of the inputs A, B and C. The majority voter can be made to work as an AND gate or as an OR gate, by setting one of the inputs as '0' and '1', respectively

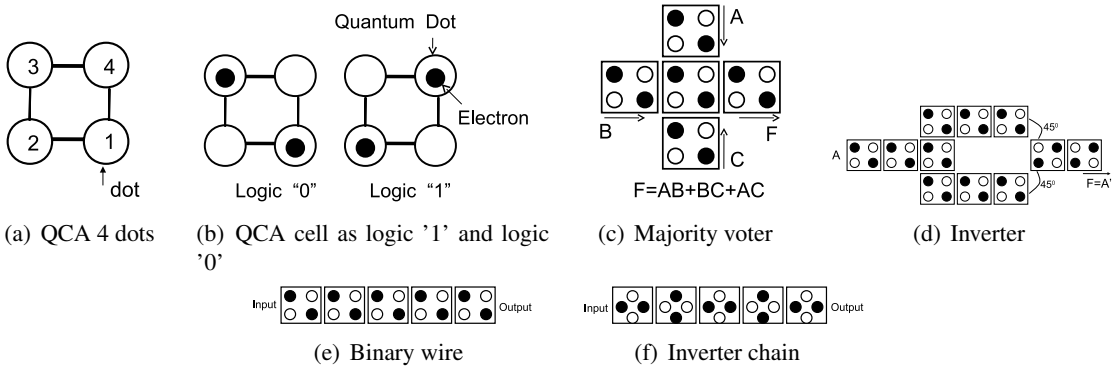
(For example, if C=0 we will get F=A.B. Similarly if C=1, we will get F=A+B). Another important gate in QCA is the inverter, which is formed when a QCA cell, say cell-1 is placed 45 degrees to another QCA cell, for example cell-0, cell-1 gets the inverse value of cell-0. There can be many ways of designing the QCA inverter, one of which is shown in Fig. 2.6(d). In QCA computing, signal transfer is made through wires that are of two types (i) Binary wire, (ii) Inverter chain.

2.4.2 Binary Wire

The electrons in adjacent QCA cells interact with each other resulting in propagation of the polarization from one cell to another. Thus, a QCA wire can be formed by arranging the QCA cells in a series in which all the neighboring cells will get the polarization of the driver cell (input). The binary wire is shown in Fig. 2.6(e).

2.4.3 Inverter Chain

Two wires in QCA can cross without interaction. This is because QCA provides an inverter chain of QCA cells, in which the dots in each cell are rotated by 45 degrees (This is not the same as in QCA inverter). Each cell in this arrangement has opposite polarization of their neighbors as they interact inversely. The inverter chain is shown in Fig. 2.6(f). In QCA, when a binary wire crosses the inverter chain, there is no interaction between the two; hence the signals in the inverter chain and binary wire can pass over each other.



(a) QCA 4 dots (b) QCA cell as logic '1' and logic '0' (c) Majority voter (d) Inverter

(e) Binary wire (f) Inverter chain

Figure 2.6. QCA cell and basic QCA devices

2.4.4 QCA Clocking

In QCA computing, the clock helps in the synchronization of circuits and provides the power required for functionality. QCA clocking consists of four phases: switch, hold, release and relax, as shown in Fig. 2.7 [88, 89]. During the switch phase, the barriers are raised and the cells become polarized, depending on the state of its adjacent cell. The states of the cells are fixed during this stage. During the hold phase, the barriers are maintained at a high value. This helps the outputs to drive the inputs of the next stage, which is in the switching phase. In the release phase, the barriers are lowered and the cells are allowed to relax to an unpolarized state. During the relaxed phase, the cells remain in an unpolarized neutral state. The cells in QCA are connected to 4 clocking zones, each lagging behind by 90 degrees in phase. QCA clocking helps in the successive transfer of information from one clock zone to the next. To gain a better understanding of QCA clocking, consider a QCA wire as shown in Fig. 2.8 driven by the fixed input, in which the cells are connected to clock zone 0 to zone 3 successively. Initially, array 1(connected to clock 0) is switching driven by the fixed input. Next, array 1 enters the hold phase and array 2 starts switching. Since we have array 3 in the relaxed state, it will not affect the computational state of array 2. Array 1 then moves to the release phase, and array 2 to the hold state. Since array 2 is in the hold phase, it will drive the input to array 3, which is in the switch phase. After that, array 3 progresses to the hold phase and array 4 goes to the switching phase [88, 89]. Therefore, we have information flow from the input to the output in a pipelined fashion [90].

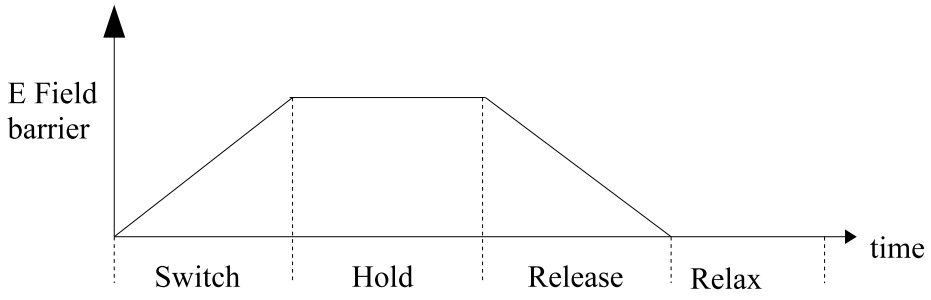


Figure 2.7. QCA 4 phase clocking

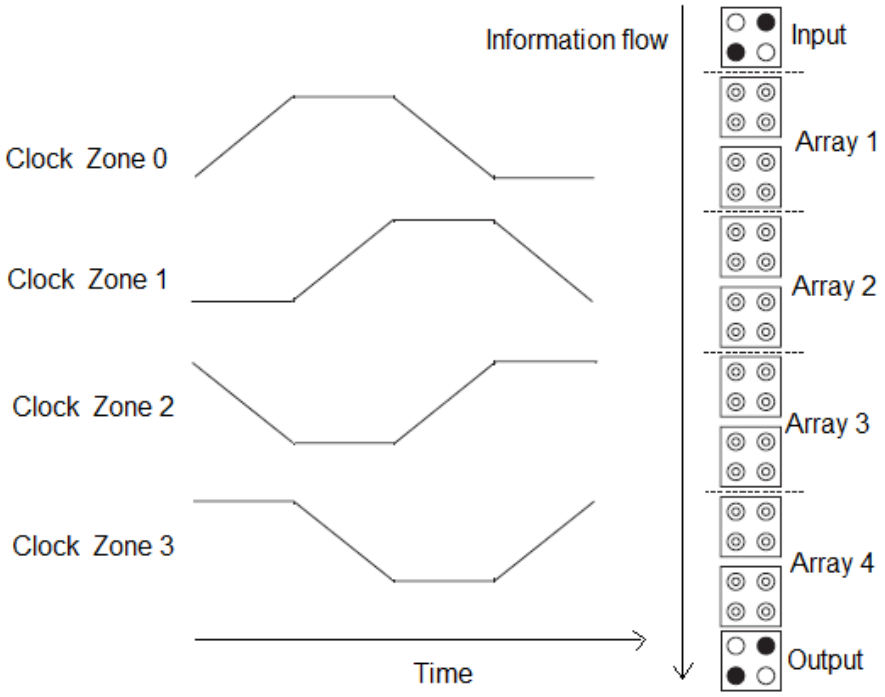


Figure 2.8. Information flow in a QCA wire

2.5 Related Work on QCA Computing

The QCA cell can be implemented using metal, semiconductor, magnetic, or molecular technologies. In molecular QCA, molecules such as 1,4-diallyl butane radical cation; 1,5,9-decatriene cation; 1,10,19-eicosatriene cation and DNA are used for QCA cell implementation [91–93]. Molecular QCA has many advantages compared to other implementation methods, such as room temperature operation, ultra small devices (i.e., the density of a device can be very high), fast switching (i.e., a device can operate in the Gigahertz range), and low power. Due to the potential advantages of molecular QCA computing, the University of Notre Dame is conducting significant research into the fabrication of molecular QCA [94]. In QCA manufacturing, defects can occur during the synthesis and deposition phases, although defects are most likely to take place during the deposition phase [32]. Researchers assume that QCA cells have no manufacturing defect; in metal QCA, faults occur due to cell misplacement. These defects can be characterized as cell displacement, cell misalignment and cell omission [95]. Researchers have shown that molecular QCA cells are more susceptible to missing and additional QCA cell defects [96]. The additional cell defect is because

of the deposition of an additional cell on the substrate. The missing cell defect is due to the missing of a particular cell.

In their seminal work, Tahoori et al. [32] address the subject of QCA testing for the first time. They investigate the defect characterization of QCA devices, and show how QCA testing is different from conventional CMOS. For instance, the unwanted complementation fault is observed at the logic level due to cell omission defects. For combinational circuits, Momenzadeh et al. modeled QCA defects at the molecular level. They perform fault characterization for single missing/ additional cell defect on QCA devices such as MV, INV, fan-out, Crosswire and L-shape wire [96]. Gupta et al. [95] present the test generation framework for QCA. They show that additional test vectors can be generated for detecting QCA defects that are undetected by the stuck-at fault model. Bridging fault on QCA wires is also addressed. In [6], reversible logic is proposed as a means to detect single missing/additional cell defects. It is shown that reversible 1D array is C-testable. Wei et al. [97] present fault-tolerant QCA designs using triple modular redundancy with shifted operands, while Dysart et al. [98] analyze how various triple redundancy schemes can impact QCA system reliability. In [89, 99], the defect characterization and tolerance of the QCA SR latch and the sequential circuit based on it are presented. Bhanja et al. address the robust coplanar crossing in QCA, proving that wires having rotated cells are thermally more stable [100]. Sultana et al. perform exhaustive testing of single stuck-at faults (SSF) in combinational logic [101].

2.6 Dissertation Context in Light of Past Works on QCA Computing

As is evident from the existing works on the fault testing of QCA circuits, there is lack of research towards the online (concurrent) testing of faults in QCA circuits. In this work, we propose a class of novel designs for the implementation of concurrently testable circuits such as latches, flip-flops, as well as complex circuits such as FPGA, for QCA computing based on a special type of reversible logic called conservative reversible logic. In conservative reversible logic, in addition to one-to-one mapping, there would be an equal number of 1s in the outputs as there would be on the inputs. Thus, by comparing the number of 1s in the inputs and to the number of 1s at outputs, any permanent or transient fault in QCA circuits can be concurrently detected. Further, we developed

novel conservative logic gate for QCA computing called the 'MV-cqca'(Majority voter conservative QCA gate) to design concurrently testable QCA circuits. This dissertation also address the offline testing of faults in QCA computing based on conservative logic. We presented two vectors, all 0s and all 1s, testable QCA layout of the reversible Fredkin gate for any single missing/additional cell defect. The testable QCA layout of the Fredkin gate can be used to design two vectors testable sequential circuits. We also proposed a new conservative logic gate called Multiplexer Conservative QCA gate (MX-cqca) that is not reversible in nature but has similar properties as the Fredkin gate of working as 2:1 multiplexer and surpasses the Fredkin gate in terms of complexity (the number of majority voter), speed and area.

CHAPTER 3

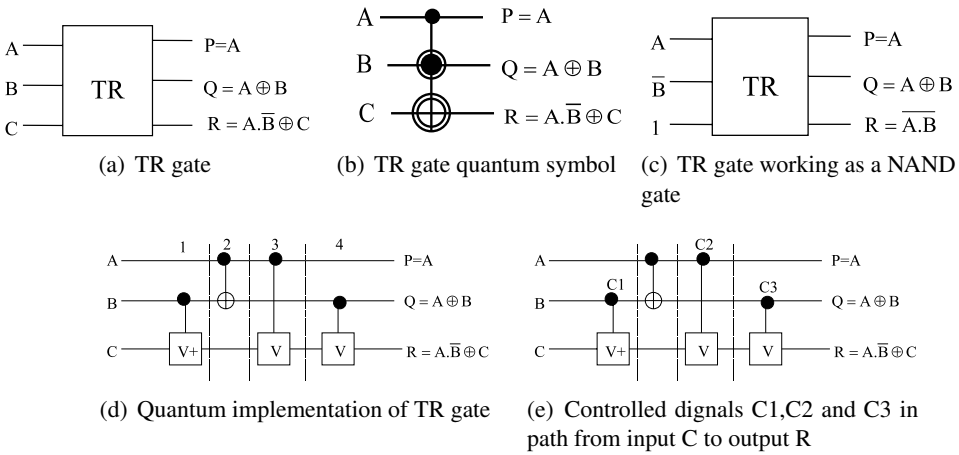
THE REVERSIBLE TR GATE AND ITS SIGNIFICANCE

In this work, we propose a novel reversible gate called the TR gate as an alternative to the Fredkin and Feynman gates, specifically to be used in the design of reversible arithmetic circuits [102]. The reversible TR gate is designed from 2x2 reversible gates such as CNOT and Controlled-V and Controlled-V+ gates. We present the designs of the reversible half subtractor, reversible full subtractor and the reversible binary comparator based on the TR gate [103, 104]. The proposed designs are shown to be better than the existing designs presented in the literature in terms of metrics such as the quantum cost, delay and garbage outputs. The reversible subtractor and comparator proposed in this work will be useful in a number of digital signal processing applications based on reversible computing where dedicated subtractor and comparator units are required.

3.1 Proposed Reversible TR Gate

The proposed reversible TR gate is a 3 inputs 3 outputs gate having inputs to outputs mapping as (P=A, Q=A $\oplus$ B, $R = A \cdot \bar{B} \oplus C$). Figure 3.1(a) shows the proposed TR gate, while Fig.3.1(b) shows the graphical notation of the TR gate of the TR gate. Table 3.1 shows the truth table of the TR gate. The proposed TR gate can be used to realize the NAND gate as shown in Fig. 3.1(c) with input B passed in complemented form by using a NOT gate. Since NAND is a universal gate, it demonstrates the universal nature of the proposed TR gate. We present the quantum implementation of the TR gate with 2x2 quantum gates in Fig. 3.1(d). The TR gate is designed from 1 Controlled V gate, 1 CNOT gate, and 2 Controlled V^+ gates resulting in its quantum cost as 4. Further, the logic depth of the quantum implementation of the TR gate is 4 resulting in its propagation delay as 4 Δ. The TR gate can realize the Boolean functions $A \cdot \bar{B} \oplus C$ and A $\oplus$ B singly. Further, it can

implement the functions such as $A \cdot \bar{B}$ when its input C is tied to 0. These properties of TR gate make it very useful in designing the reversible binary arithmetic circuits.



(a) TR gate

(b) TR gate quantum symbol

(c) TR gate working as a NAND gate

(d) Quantum implementation of TR gate

(e) Controlled dignals C1,C2 and C3 in path from input C to output R

Figure 3.1. The TR gate, its symbol, quantum implementation and its working in various modes

Table 3.1. Truth table for the TR gate

A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	1	0
0	1	1		0	1	1
1	0	0		1	1	1
1	0	1		1	1	0
1	1	0		1	0	0
1	1	1		1	0	1

3.2 Functional Verification of the Quantum Implementation of the TR gate

We have functionally verified the working of the proposed quantum implementation of the TR gate. The output P of the TR gate is equal to A and the output Q is equal to $A \oplus B$ thus the functionality of the outputs P and Q is easy to verify. The path of the outputs Q consists of V^+ and V gates thus cannot be directly verified. In order to verify the output R, we use the truth table of the TR gate, and compare the expected output with the output produced. For the path from input

C to output R, the controlled signals of V^+ and V gates are labelled as C1, C2 and C3. A small illustration of the verification of the output R is shown below for two input combinations in which the inversion and identity properties of V and V^+ gates will be utilized (the properties of V and V^+ gates working as an NOT gate and identity gate are discussed earlier in chapter 2):

- Consider the case when inputs ABC to have value 101. Now we have A=1, B=0 and C=1 thus control signals C1,C3 and C3 will have values as C1=0, C2=1 and C3=1, thus first V^+ gate will not play the controlling role and will just work as a wire transferring the input C. The second and third V gates will be active playing the controlling role resulting in a NOT gate (two V gates in series work as a NOT gate). Thus at output R we will have the inverted value of C resulting in the value at output R as '0'. From the logic equation the output R is $R = A \cdot \bar{B} \oplus C$ which also produce the value as 0. This verifies the working of the quantum implementation of the TR gate for inputs ABC to have value 101.

- Consider the case when inputs ABC to have value 111, we have A=1, B=1 and C=1 thus control signals C1,C3 and C3 will have values as C1=1, C2=1 and C3=0. The second V gate will not play control role since control signal C3 is 0, the first V^+ and third V^+ gate will form an identity resulting in value of input c passed to output R producing R=C. Thus the output R will be 1. From the logic equation R is $R = A \cdot \bar{B} \oplus C$ which also produce the value as 1. This verifies the working of the quantum implementation of the TR gate for inputs ABC to have value 111. Similarly, the proposed design is tested for all 8 inputs combinations and it matches the expected output.

3.3 TR Gate as a Reversible Half Subtractor

Before discussing the existing design of reversible half subtractor, the basic working of a half subtractor is illustrated. Let A and B are two binary numbers. The half subtractor performs A-B operation. Table 3.2 shows the truth table of the half subtractor. The output of the XOR gate produces the difference between A and B. The output of the AND gate $\bar{A} \cdot B$ produces a Borrow. Thus, the output function will be $Borr = \bar{A} \cdot B$; $Diff = A \oplus B$. In the existing literature,

the reversible half subtractor as shown in Fig. 3.2 is designed from 2 CNOT gates (2 Feynman gates) and 1 Toffoli gate [1]. The design in [1] is the most widely used design of quantum half subtractor [3, 105]. The existing design of the reversible half subtractor in [1] has the quantum cost of 7 and delay of 7 Δ, while the existing design in [102] has the quantum cost of 6 and delay of 6 Δ. In this work, we propose the reversible half subtractor design based on a new quantum implementation of the reversible TR gate.

Table 3.2. Truth table of half subtractor

A	B		Borr	Diff
0	0		0	0
0	1		1	1
1	0		0	1
1	1		0	0

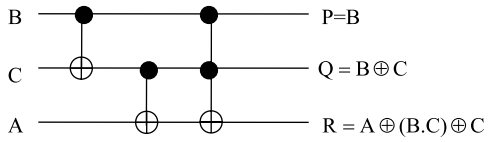
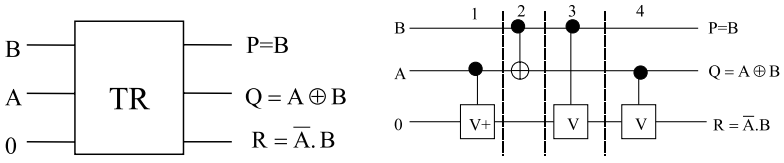


Figure 3.2. Existing design for quantum half subtractor [1]

Figure 3.3(a) shows the working of the TR gate as a reversible half subtractor. As shown in Fig.3.3(b), the TR gate implements the reversible half subtractor with quantum cost of 4, delay of 4 Δ and 0 garbage outputs (the inputs regenerated at the outputs are not considered as garbage outputs). A comparison of the reversible half subtractors is shown in Table 3.3. Thus proposed design achieves 43% reduction in terms of quantum cost (QC) and delay compared to design presented in [1], while the improvement compared to design presented in [102] is 33% both in terms of the quantum cost (QC) and the delay.

3.4 Design of Reversible Full Subtractor

To subtract three binary numbers, one can use a full subtractor which realizes the operation Y=A-B-C. The truth table of the full subtractor is shown in Table 3.4. This gives the equation of



(a) TR gate as a reversible half subtractor

(b) Quantum implementation of TR gate based reversible half subtractor

Figure 3.3. Improved design of TR gate based reversible half subtractor

Table 3.3. A comparison of reversible half subtractors

	QC	Delay
Design proposed in [1]	7	7
Design proposed in [102]	6	6
Proposed design	4	4
Improvement in % w.r.t [1]	43	43
Improvement in % w.r.t [102]	33	33

the borrow and difference as follows: $Diff = A \oplus B \oplus C$; $Borr = A \cdot \bar{B} \oplus \overline{A \oplus B} \cdot C$. In the existing literature, the reversible full subtractor is designed with 2 Toffoli gates, 3 Feynman gates and 2 NOT gates [2]. The existing design of reversible full subtractor is shown in Fig.3.4. Thus, the existing reversible full subtractor has the quantum cost of 15, delay of 15 Δ. In this work, we propose the design of the reversible full subtractor in Fig.3.5. It requires two TR gates to design a reversible full subtractor with no garbage output and 1 ancilla input. The quantum realization of the TR gate based reversible full subtractor is shown in Fig. 3.6(a). From Fig. 3.6(a), we can see that the TR gate based reversible full subtractor has the quantum cost of 8 with delay of 8 Δ. As can be seen in the Fig. 3.6(a) the fourth gate (V gate) and the fifth gate (V^+ gate) are in series thus forming an identity and can be removed. This results in a new optimized design of TR gate based reversible full subtractor with quantum cost of 6 and delay of 6 Δ as shown in Fig.3.6(b). Further, in the optimized design shown in Fig.3.6(b) few of the gates can be moved by applying the move rules discussed in [106], without affecting the functionality of the circuit. This results in two pairs of V and CNOT gates operating in parallel, and generate a new design of the reversible full subtractor as shown in Fig.3.6(c) that has the delay of 4 Δ. Thus, compared to the existing design [2], the

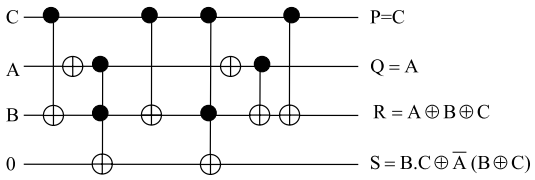


Figure 3.4. Existing design of quantum full subtractor [2]

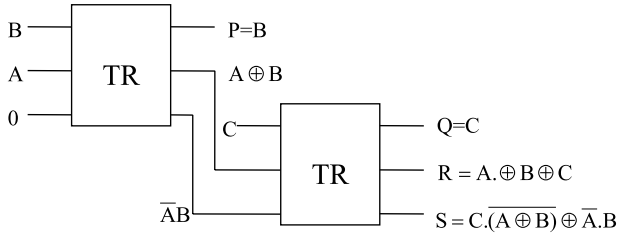
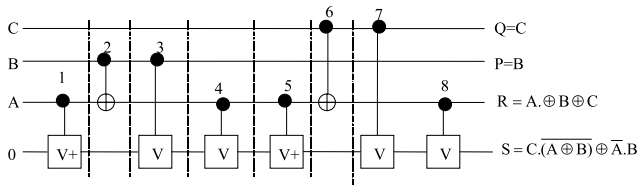


Figure 3.5. The TR gate as a full subtractor

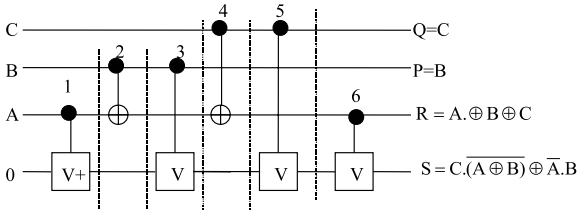
proposed reversible full subtractor design based on TR gate has an improvement ratios of 60% and 73.33% in terms of quantum cost (QC) and delay, respectively. The improvements compared to existing design [102] are 50% and 66.67% in terms of the quantum cost and the delay. All the existing designs of the reversible full subtractor also have no garbage outputs and 1 ancilla input. The results are summarized in 3.5.

Table 3.4. Truth table of full subtractor

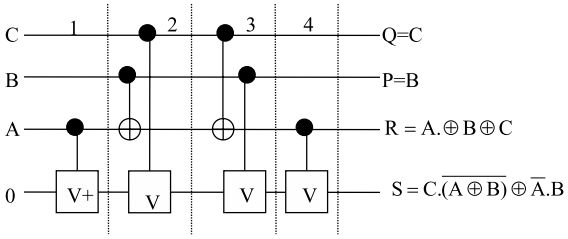
A	B	C		Borr	Diff
0	0	0		0	0
0	0	1		1	1
0	1	0		1	1
0	1	1		1	0
1	0	0		0	1
1	0	1		0	0
1	1	0		0	0
1	1	1		1	1



(a) Quantum implementation of TR gate based reversible full subtractor



(b) Optimized quantum implementation of TR gate based reversible full subtractor



(c) Delay optimization of TR gate based reversible full subtractor

Figure 3.6. Optimization of TR gate based reversible full subtractor

Table 3.5. A comparison of reversible full subtractors

	QC	Delay
Design proposed in [2]	15	15
Design proposed in [102]	12	12
Proposed Design	6	4
Improvement in % w.r.t [2]	60	73.33
Improvement in % w.r.t [102]	50	66.67

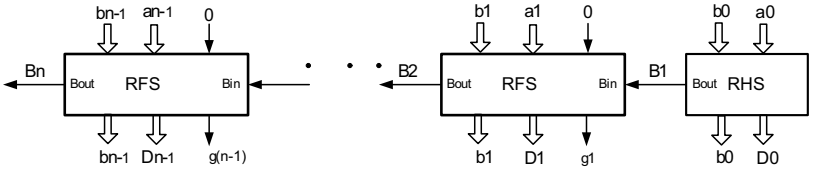


Figure 3.7. n bit reversible full subtractor

3.5 Design of n Bit Reversible Subtractor

Once we have designed the 1 digit reversible full subtractor, the n bit reversible subtractor can be designed by cascading a reversible half subtractor followed by a reversible full subtractor, followed by a reversible full subtractor and so on using the ripple borrow approach. A n bit reversible subtractor subtracting n bit numbers a_i and b_i where $0 \leq i \leq n-1$ is shown in Fig. 3.7. The design of n bit reversible subtractor using the ripple borrow approach has $n-1$ ancilla inputs, $n-1$ garbage outputs, quantum cost of $6n-2$ and delay of $4n$ Δ. Table 3.6 illustrates that the proposed design is better than the existing design in terms of the quantum cost and the delay.

Table 3.6. A comparison of n bit reversible full subtractors

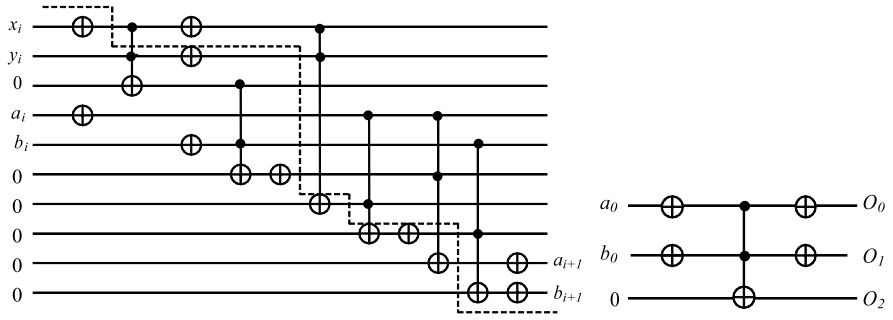
	QC	Delay
Design proposed in [2]	15n	15n
Design proposed in [102]	12n	12n
Proposed Design	6n-2	4n
Improvement in % w.r.t [2]	60	73.33
Improvement in % w.r.t [102]	50	66.67

3.6 Design of Binary Comparator Based on TR Gate

The existing reversible design of the binary comparator is a serial architecture [3] that has the latency of O(n). In the existing design approach, the comparator consists of a chain of reversible comparison cells that performs the operation of comparing a bit of the first number say x with the corresponding bit of the second number say y. In [3], a 1 bit comparator cell is designed using reversible Toffoli gates, Feynman gates and the NOT gates. The 1 bit reversible comparator cell is shown in Fig.3.8(a) while the output circuit is shown in Fig.3.8(b). The output circuit takes the two inputs a_0(x<y) and b_0(x>y) to generate three signals O_0(x<y), O_2(x>y) and O_3(x=y). The reversible output circuit is designed from 1 Toffoli gate and 4 NOT gates. In the design presented in Fig.3.8(a), the reversible 1 bit comparator cell has 8 garbage outputs and has the quantum cost of 39 and delay of 24 Δ (the delay is marked by dashed line in Fig. 3.8(a)). The reversible output circuit has the quantum cost of 9 and delay of 7 Δ. The 8 bit and 64 bit reversible comparators can be designed by cascading 8 copies and 64 copies of the 1 bit comparator cell, respectively, followed by the reversible output circuitry. To illustrate an example of the existing reversible binary comparator, the design of the 3 bit comparator which compares two numbers x and y is shown in Fig.3.9 in which the x_2 and b_2 are the most significant bits. The outputs a_0 and b_0 are passed to the output circuit to generate three signals O_0(x<y), O_1(x>y) and O_2(x=y). Using the serial design approach presented in [3], the 8 bit reversible comparator will have the quantum cost of 321, delay of 199 Δ and generates 64 garbage outputs. The reversible design of the 64 bit reversible binary comparator will have the quantum cost of 2505, delay of 1543 Δ and generates 512 garbage outputs.

3.6.1 Proposed Comparator Design

We present a new reversible design of the binary comparator using the binary tree structure. The proposed design has the latency of O($log_2(n)$). The proposed reversible binary tree comparator has a binary tree structure in which each node consists of a 2 bit reversible binary comparator that can compare two 2 bit numbers x(x_i, x_{i-1}) and y(y_i, y_{i-1}), to generate 2 bit outputs indicating whether x(x_i, x_{i-1})>y(y_i, y_{i-1}) or x(x_i, x_{i-1})<y(y_i, y_{i-1}). Thus, we propose a novel design of the 2 bit reversible comparator efficient in terms of quantum cost, delay and the number of garbage



(a) Reversible implementation of 1 bit comparator Cell (b) Reversible implementation of output circuit

Figure 3.8. Reversible designs of comparator cell and output circuit [3]

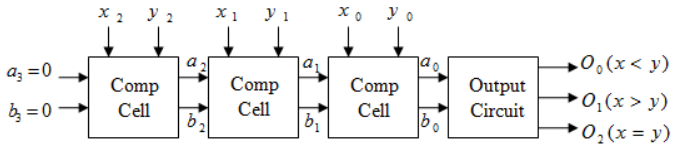


Figure 3.9. Serial design of reversible 3 bit comparator [3]

outputs. Each internal node (2 bit reversible binary comparator) of the binary tree receives the partial comparison results from the left and the right children and propagates the 2 bit outputs of comparison to its parent. Finally, the root node which is also a 2 bit reversible binary comparator generates the 2 bit comparison results of the two n-bit numbers x and y to evaluate whether x>y or x<y. The 2 bit results of the root node is passed to the output circuit which was shown in Fig.3.8(b) to generate three signal O_0(x<y), O_1(x>y) and O_2(x=y). The proposed approach is illustrated with the design of 8 bit and 64 bit reversible comparators.

3.6.2 Design of the 2 Bit Reversible Binary Comparator

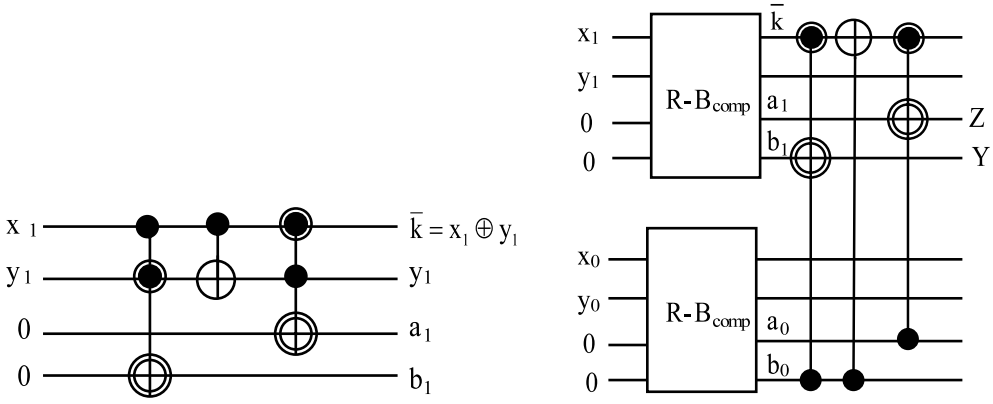
For designing the 2 bit reversible binary comparator, consider two 2 bit binary numbers x(x_1, x_0) and y(y_1, y_0). The condition for x>y: ($x_1 > y_1$) or ($x_1 = y_1$ and $x_0 > y_0$). Thus $Y = x_1\bar{y_1} + kx_0\bar{y_0}$ should be 1 for x>y, where $k = \overline{x_1 \bigoplus y_1}$ is 1 when $x_1 = y_1$. Similarly, the condition for x<y: ($x_1 < y_1$) or ($x_1 = y_1$ and $x_0 < y_0$). Thus $Z = \bar{x_1}y_1 + k\bar{x_0}y_0$ should be 1 for x<y, where $k = \overline{x_1 \bigoplus y_1}$ is 1 when $x_1 = y_1$. From the above equations of Y and Z we observed that to design

the 2 bit reversible binary comparator we need to have the reversible module that can generate the outputs as $x_1\bar{y_1}$ and $\bar{x_1}y_1$, the module is called R-Bcomp. Once we have R-Bcomp module ready it can also generate $x_0\bar{y_0}$ and $\bar{x_0}y_0$ by changing the inputs. We propose the reversible design of the R-Bcomp module using the reversible TR gate. TR gate is very useful as when its input C=0 we will have $R = A \cdot \bar{B}$ which can implement the logic functions such as $x_1\bar{y_1}$ and $\bar{x_1}y_1$. Thus we used TR gate to design the R-Bcomp module as shown in Fig.3.10(a). In the R-Bcomp module the outputs $\bar{x_1}y_1$ and $x_1\bar{y_1}$ are labelled as a1 and b1, respectively. Further it can observed that R-Bcomp also produces $\bar{k} = \overline{x_1 \bigoplus y_1}$ which is beneficial in the design of 2 bit comparator.

After careful analysis, we modified the logic equations of $Y = x_1\bar{y_1} + kx_0\bar{y_0}$ and $Z = \bar{x_1}y_1 + k\bar{x_0}y_0$ to $Y = x_1\bar{y_1} \oplus kx_0\bar{y_0}$ and $Z = \bar{x_1}y_1 \oplus k\bar{x_0}y_0$, respectively. This is because since any function F=A+BC will produce the same output as the function F=$A \oplus BC$ except when the variables A,B,C have the values as A=1,B=1 and C=1. In the equation of Y and Z explained above when k=1 then $x_1\bar{y_1}$ and $\bar{x_1}y_1$ will be 0 and vice versa, hence we are able to replace + operator with $\oplus$ operator without affecting the functionality of the design. This helps in mapping of the equations of Y and Z on the third output of the TR gate which is R=$A\bar{B} \oplus C$. Since our R-Bcomp circuit shown in Fig. 3.10(a) produces the $\bar{k}$ it can be passed as the B input of the TR gates to produce Y and Z signals without the need of the NOT gates. We use the R-Bcomp shown in Fig.3.10(a) along with CNOT gate and TR gate to design the 2 bit reversible comparator as shown in Fig.3.10(b). The 2 bit reversible comparator has 6 garbage outputs(the unused outputs in the design) and has the quantum cost of 18 and delay of 18 Δ.

3.6.3 Design of 8 Bit Reversible Binary Comparator

The proposed design of the 8 bit reversible comparator using reversible 2 bit comparator is shown in Fig.3.11 *(the garbage outputs in the design are not shown)*. The design contains 2 bit reversible binary comparators as the nodes of the binary tree. Since n=8, the tree will have $log_2(8)$= 3 levels. The design requires seven 2 bit binary comparators along with a reversible output circuitry (R-O/P Ckt). The reversible output circuit (R-O/P Ckt) is similar to the output circuit shown in Fig.3.8(b) and is primarily used to generate O_2(x=y) signal from x>y and x<y outputs. The quan-



(a) R-Bcomp design (b) 2 bit reversible comparator

Figure 3.10. Proposed design of 2 bit reversible comparator

tum cost of the proposed 8 bit reversible binary comparator is: { 7 * quantum cost of 2 bit binary reversible comparator+ quantum cost of the reversible output circuitry =7*18+9 =135 }. The delay of the 8 bit reversible binary comparators is:{ 3 * delay of 2 bit binary reversible comparator+ delay of the reversible output circuitry = 3 * 18 Δ+ 7 Δ= 61 Δ }. The number of garbage outputs of the 8-bit reversible binary comparator is: {7 * garbage outputs of the 2 bit binary reversible comparator = 7* 6 = 42 }. The improvement compared to the design presented in [3] for 8 bit comparator is shown in Table 3.7.

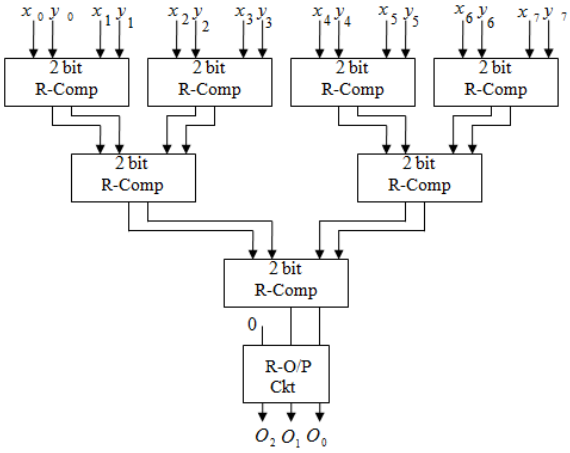


Figure 3.11. Proposed reversible 8 bit comparator

Table 3.7. A comparison of reversible 8 bit reversible comparator

	QC	Delay	GOs
Serial Design [3]	321	199	64
Proposed Design	135	61	42
Improvement in %	57.94	69.34	34.37

3.6.4 Design of 64 Bit Reversible Binary Comparator

We also designed the 64 bit reversible comparator based on the binary tree based approach. The design will contain nine 8 bit reversible binary comparator as illustrated in Fig.3.12 and the reversible output circuitry. The eight 8 bit reversible binary comparator are designed from 2 bit reversible binary comparators as illustrated in the Fig.3.11 with the difference that they will not contain the reversible output circuitries at the outputs. The proposed design of 64 bit reversible binary comparator will have the quantum cost of { 9 * quantum cost of 8 bit binary reversible comparator without the reversible output circuitry at the output + quantum cost of the reversible output circuitry=9 * 126+9 =1143 }. The delay of the 64 bit reversible binary comparators is:{ 2 * delay of 8 bit binary reversible comparator without the reversible output circuitry + delay of the reversible output circuitry = 2 * 54 Δ+ 7 Δ= 115Δ }. The number of garbage outputs of the 8 bit reversible binary comparator is: {9 * garbage outputs of the 8 bit binary reversible comparator without the reversible output circuitry = 9 * 42 = 378 }. Using the similar approach, the reversible n-bit tree based comparator can be designed. A comparison of the proposed reversible tree based comparator with the existing serial based approach [3] is illustrated in Table 3.8 for 64 bit binary comparator. We can see from Table 3.8 that the proposed design has an improvement of 92.7% compared to the design presented in [3] in terms of delay because of its logarithmic latency.

Table 3.8. A comparison of reversible 64 bit comparator

	QC	Delay	GOs
Serial Design [3]	2505	1543	512
Proposed Design	1143	115	378
Improvement in %	54.37	92.5	26.17

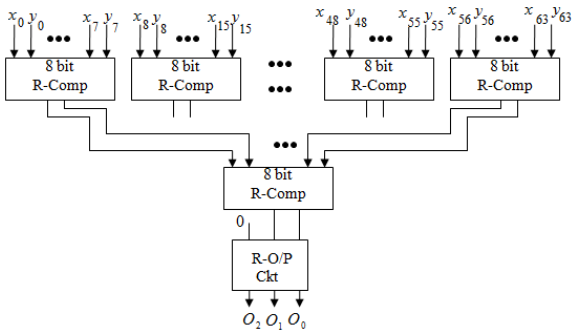


Figure 3.12. Proposed reversible 64 bit comparator

3.7 Conclusions

In this work, we have presented a new reversible TR gate and efficient designs of reversible subtractors and binary comparators based on it. The proposed reversible subtractor designs are shown to be better than the existing designs in terms of the quantum cost and delay while maintaining the minimal number of garbage outputs. We conclude that the design of a specific reversible gate for a particular combinational function can be very much beneficial in minimizing the quantum cost, delay and the garbage outputs. Further, we observe a special property of the reversible TR gate in relation to the popular Peres gate. We derive the inverse of the TR gate since a reversible gate can be combined with its inverse reversible gate to minimize the garbage outputs [34]. In order to derive the logic equations of the inverse TR gate, we performed the reverse mapping of the TR gate outputs working as inputs to generate the inputs of the TR gate. We observe that the inverse of TR gate is same as the existing Peres gate having inputs to outputs mapping as P=A, Q=A $\oplus$ B, $R = A \cdot B \oplus C$, where A, B and C are the inputs, and P, Q and R are the outputs, respectively. Among the existing reversible gates, the Peres gate has the minimum quantum cost of 4. Thus, the proposed TR gate can be combined with its inverse reversible gate (Peres gate) to design minimal quantum cost and garbageless reversible circuits. The proposed efficient designs of reversible subtractors will find applications in quantum/reversible computing requiring dedicated subtractors units. We have also presented reversible binary comparator based on binary tree based approach having logarithmic latency. The design is based on the useful properties of the TR gate suitable for mapping the comparator Boolean equations. The design presented is shown better than the existing

serial design of reversible binary comparator in terms of quantum cost, garbage outputs and the propagation delay. Due to logarithmic latency, the proposed design achieves improvement of 92.7% in terms of delay compared to the existing serial design of 64-bit binary reversible comparator. The comparator designs proposed in this work can be useful in realizing the hardware design of the quantum algorithms such as Shors' factoring algorithm.

CHAPTER 4

REVERSIBLE BINARY AND BCD ADDER/SUBTRACTOR CIRCUITS

The proposed work focuses on the design of reversible binary adder circuits which are further used to design reversible binary subtractor, reversible adder-subtractor and reversible BCD adder circuits primarily optimized for number of ancilla input bits and the garbage outputs [107, 108]. As the optimization of ancilla input bits and the garbage outputs may impact the design in terms of the quantum cost and the delay, thus quantum cost and the delay parameters are also considered for optimization with primary focus towards the optimization of number of ancilla input bits and the garbage outputs. To the best of our knowledge this is the first attempt in the literature that explores the reversible BCD adder designs with the goal of optimizing the number of ancilla input bits and the garbage outputs. First, we propose two new designs for the reversible ripple carry adder: (i) one with no input carry c_0 and no ancilla input bits, and (ii) one with input carry c_0 and no ancilla input bits. The proposed reversible ripple carry adder designs with no ancilla input bits have less quantum cost and logic depth (delay) compared to their existing counterparts in the literature. In these designs, the quantum cost and delay are reduced by deriving designs based on the reversible Peres gate and the TR gate. Next, the two adder designs, the one with no input carry and the one with input carry, are used to design reversible binary subtractor using the properties that the subtraction operation can be defined as $a - b$=$\overline{\bar{a} + b}$ and $a - b$=$a + \bar{b} + 1$, respectively. Next, the proposed design methodologies for the design of reversible binary subtractor are adapted to design the reversible binary adder-subtractor that can perform addition as well as subtraction operation depending on the value of the control signal. Next, four new designs for the reversible BCD adder are presented based on the following two approaches: (i) the addition is performed in binary mode and correction is applied to convert to BCD when required through detection and correction, and (ii)

the addition is performed in binary mode and the result is always converted using a binary to BCD converter. The various reversible components needed in the BCD adder design are optimized in parameters of number of ancilla input bits/qubits and the number of garbage outputs and explore the possible best values for the quantum cost and delay. The comparison of the proposed designs with the existing designs is also illustrated. Finally, a new synthesis framework for automatic generation of reversible arithmetic circuits optimizing the metrics of ancilla inputs, garbage outputs, quantum cost and the delay based on the user specifications is presented.

4.1 Design Methodology of Proposed Reversible Ripple Carry Adder With No Input Carry

We present the design of reversible ripple carry adder with no input carry(c_0) and is designed without any ancilla inputs and the garbage outputs. *The proposed method improves the quantum cost and the delay of the reversible ripple carry adder compared to the existing design approaches which have optimized the adder design in terms of number of ancilla inputs*. Consider the addition of two n bit numbers a_i and b_i stored at memory locations A_i and B_i, respectively, where $0 \leq i \leq n-1$. Further, consider that memory location A_n is initialized with z $\in \{0, 1\}$. At the end of the computation, the memory location B_i will have s_i, while the location A_i keeps the value a_i. The additional location A_n that initially stores the value z will have the value $z \oplus s_n$ at the end of the computation. Thus A_n will have the value of s_n when z=0. Here, s_i is the sum bit produced and is defined as:

$$s_i = \begin{cases} a_i \oplus b_i \oplus c_i & \text{if } 0 \leq i \leq n-1 \\ c_n & \text{if } i = n \end{cases}$$

where c_i is the carry bit and is defined as:

$$c_i = \begin{cases} 0 & \text{if } i = 0 \\ a_{i-1}b_{i-1} \oplus b_{i-1}c_{i-1} \oplus c_{i-1}a_{i-1} & \text{if } 1 \leq i \leq n \end{cases}$$

The proposed design methodology of generating the reversible ripple carry adder with no input carry minimizes the garbage outputs by producing the carry bits c_i based on the inputs a_{i-1}, b_{i-1} and the carry bit c_{i-1} from the previous stage. Once all the carry bits c_i are generated they are stored at memory location A_{i-1} which was initially used for storing the input a_{i-1} for $0 \leq i \leq n-1$.

After the generated carry bits are used for further computation, the location A_i are restored to the value a_i while the location B_i stores the sum bit s_i for $0 \leq i \leq n-1$. Thus restoring of location A_i to the value a_i helps in minimizing the garbage outputs. Since no constant input having the value as 0 is needed in the proposed approach, it saves the ancilla inputs. The proposed methodology of generating the reversible ripple adder circuit without input carry is referred as methodology 1 in this work. The proposed methodology is generic in nature and can design the reversible ripple carry adder circuit with no input carry of any size. The steps involved in the proposed methodology is explained for addition of two n bit numbers a_i and b_i, where $0 \leq i \leq n-1$. An illustrative example of generation of reversible ripple carry adder circuit that can perform the addition of two 8 bit numbers a=$a_0...a_7$ and b=$b_0...b_7$ is also shown. The steps involved are as follows:

Step 1: For i=1 to n-1:

At pair of locations A_i and B_i apply the CNOT gate such that the location A_i will maintain the same value, while location B_i transforms to ($*A_i \oplus *B_i$), where $*A_i$ and $*B_i$ represent the values stored at location A_i and B_i. The step 1 is shown for reversible ripple carry adder circuit that can perform the addition of two 8 bit numbers in Fig.4.1(a).

Step 2: For i=n-1 to 1:

At pair of locations A_i and A_{i+1} apply the CNOT gate such that the location A_i will maintain the same value, while the location A_{i+1} transforms to ($*A_i \oplus *A_{i+1}$). The step 2 is shown for reversible 8 bit adder circuit in Fig.4.1(b).

Step 3: For i=0 to n-2:

At locations B_i, A_i and and A_{i+1} apply the Toffoli gate such that B_i, A_i and and A_{i+1} are passed to the inputs A, B, C, respectively, of the Toffoli gate. The step 3 is shown for reversible 8 bit adder circuit in Fig.4.1(c).

Step 4: For i=n-1 to 0:

At locations A_i, B_i and and A_{i+1} apply the Peres gate such that A_i, B_i and and A_{i+1} are passed to the inputs A, B, C, respectively, of the Peres gate. The step 4 is shown for reversible 8 bit adder circuit in Fig.4.2(a).

Step 5: For i=1 to n-2:

At pair of locations A_i and A_{i+1} apply the CNOT gate such that the location A_i will maintain the same value, while location B_i transforms to the value ($*A_i \oplus *B_i$). The step 5 is shown for reversible 8 bit adder circuit in Fig.4.2(b).

Step 6: For i=1 to n-1 :

At pair of locations B_i and A_i apply the CNOT gate such that the location A_i will maintain the same value, while location B_i transforms to the value ($*A_i \oplus *b_i$). This final step will result in a reversible adder circuit that can perform the addition of two n bit numbers. For reversible 8 bit adder circuit, the design is shown in Fig.4.2(c).

Thus, the proposed methodology implements the reversible ripple carry adder with no input carry, without any ancilla input bit. Since in the design of the reversible BCD adder, the 4 bit reversible ripple carry adder will be used, thus its design is also illustrated in Fig.4.3

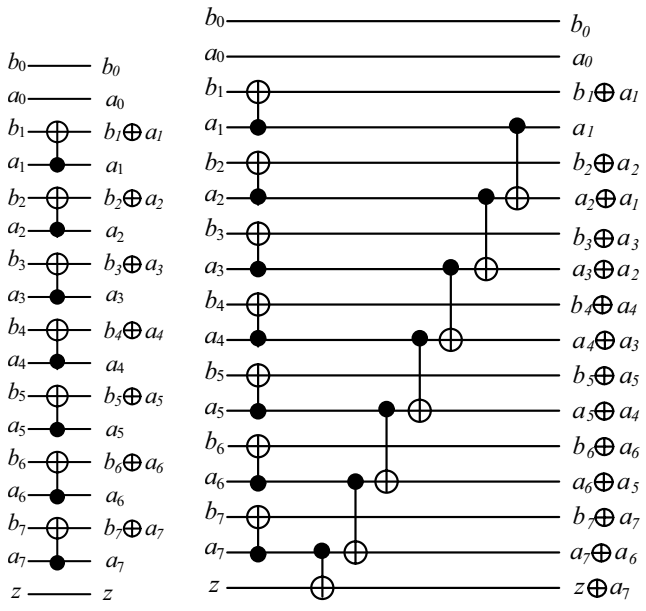
Theorem 1: *Let a and b are two n bit binary numbers represented as* a_i *and* b_i *and* $z \in \{0, 1\}$ *is another 1 bit input, where* $0 \leq i \leq n-1$*, then the proposed design steps of methodology 1 result in the ripple carry adder circuit that works correctly. The proposed design methodology designs an n bit adder circuit that produces the sum output* s_i *at the memory location where* b_i *is stored, while restores the location where* a_i *is initially stored to the value* a_i *for* $0 \leq i \leq n-1$*. Further, the proposed design methodology transforms the memory location where z is initially stored to* $z \oplus s_n$*, and restores the memory location where the input carry* c_0 *is initially stored to the value* c_0.

Proof: The proposed approach will make the following changes on the inputs that are illustrated as follows:

Step 1: The step 1 of the proposed approach transforms the input states to

$$|b_0\rangle |a_0\rangle \left(\bigotimes_{i=1}^{n-1} |b_i \oplus a_i\rangle |a_i\rangle \right) |z\rangle$$

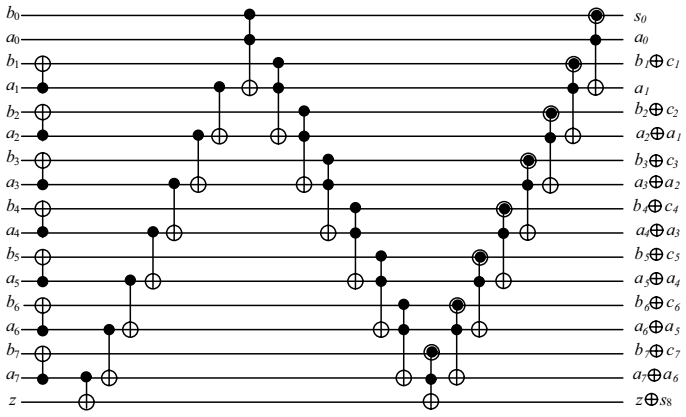
An example of the transformation of the input states after step 1 is illustrated for 8 bit reversible ripple carry adder circuit in Fig.4.1(a).



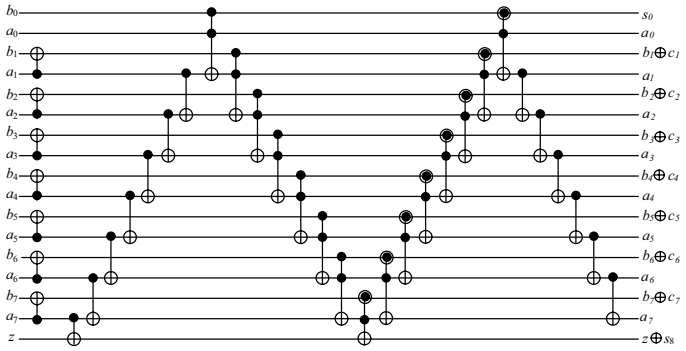
(a) After Step 1 (b) After Step 2

(c) After Step 3

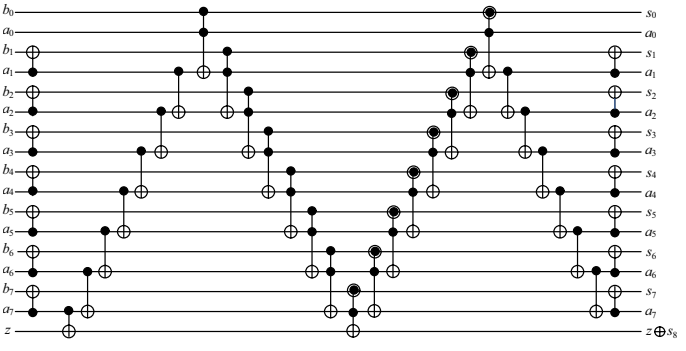
Figure 4.1. Circuit generation of reversible 8 bit adder with no input carry: Steps 1-3



(a) After Step 4



(b) After Step 5



(c) After Step 6

Figure 4.2. Circuit generation of reversible 8 bit adder with no input carry: Steps 4-6

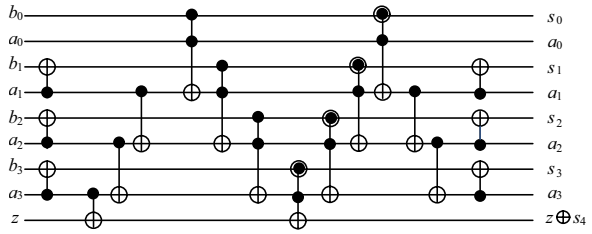


Figure 4.3. Proposed reversible 4 bit adder without input carry

Step 2: The step 2 of the proposed approach transforms the input states to

$$|b_0\rangle |a_0\rangle |b_1 \oplus a_1\rangle |a_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus a_i\rangle |a_i \oplus a_{i-1}\rangle \right) |z \oplus a_{n-1}\rangle$$

An example of the transformation of the input states after step 2 is illustrated for 8 bit reversible ripple carry adder circuit in Fig.4.1(b).

Step 3: The step 3 has n-1 Toffoli gates. The first Toffoli gate takes the inputs as b_0,a_0 and a_1 and produces the output as b_0,a_0 and $a_1 \oplus c_1$. The third output of the Toffoli gate produces $a_1 \oplus c_1$ because c_1=$a_0 \cdot b_0$ where c_1 represents the generated output carry after addition of a_0 and b_0. The remaining n-2 Toffoli gates take the inputs as $b_i \oplus a_i$, $a_i \oplus c_i$, $a_i \oplus a_{i+1}$ and produces the outputs as $b_i \oplus a_i$, $a_i \oplus c_i$, $a_{i+1} \oplus c_{i+1}$ where $1 \leq i \leq n-1$. Thus, after the step 3, the input states is transformed to

$$|b_0\rangle |a_0\rangle \left(\bigotimes_{i=1}^{n-1} |b_i \oplus a_i\rangle |a_i \oplus c_i\rangle \right) |z \oplus a_{n-1}\rangle$$

An example of the transformation of the input states after step 3 is illustrated for 8 bit reversible ripple carry adder circuit in Fig.4.1(c).

Step 4: The step 4 has n Peres gates. The n-1 Peres gate take the inputs as $a_i \oplus c_i$, $b_i \oplus a_i$, $a_{i+1} \oplus c_{i+1}$ to produce the outputs as $a_i \oplus c_i$, $b_i \oplus c_i$, $a_i \oplus a_{i+1}$. The third outputs of the Peres gate are $a_i \oplus a_{i+1}$ because it realizes the function $A \cdot B \oplus C$ where A, B and C are the inputs of the Peres gate. Hence the Peres gates will have the third outputs as $a_i \oplus c_i \cdot b_i \oplus c_i \oplus a_i \oplus a_{i+1}$= $a_i \oplus a_{i+1}$. The nth Peres gate takes the inputs as a_0, b_0, $a_1 \oplus c_1$ to produce the outputs as a_0,

$a_0 \oplus b_0$, a_1. Please note that $s_0 = a_0 \oplus b_0$. Thus the step 4 transforms the input states to

$$|s_0\rangle |a_0\rangle |b_1 \oplus c_1\rangle |a_1\rangle \left(\bigotimes_{i=2}^{n-1} |b_i \oplus c_i\rangle |a_i \oplus a_{i-1}\rangle \right) |z \oplus s_n\rangle$$

An example of the transformation of the input states after step 4 is illustrated for 8 bit reversible ripple carry adder circuit in Fig.4.2(a).

Step 5: The step 5 of the proposed approach transforms the input states to

$$|s_0\rangle |a_0\rangle \left(\bigotimes_{i=1}^{n-1} |b_i \oplus c_i\rangle |a_i\rangle \right) |z \oplus s_n\rangle$$

An example of the transformation of the input states after step 5 is illustrated for 8 bit reversible ripple carry adder circuit in Fig.4.2(b).

Step 6: The step 6 of the proposed approach transforms the input states to

$$\left(\bigotimes_{i=0}^{n-1} |s_i\rangle |a_i\rangle \right) |z \oplus s_n\rangle$$

An example of the transformation of the input states after step 6 is illustrated for 8 bit reversible ripple carry adder circuit in Fig.4.2(c).

Thus, the proposed six steps transform the memory location where b_i is initially stored to the sum output s_i, while the location where a_i is initially stored will be restored to the value a_i for $0 \leq i \leq n-1$ after the generation of the output carries and their subsequent use to produce the sum outputs. The memory location where z is stored will have $z \oplus s_n$ and the memory location where the input carry c_0 was stored initially will be restored to the value c_0. In summary, the proposed design methodology 1 generates the n bit reversible ripple carry adder that is functionally correct.

4.1.1 Delay and Quantum Cost

- At step 1 the proposed methodology needs $n-1$ CNOT gates working in parallel thus this step has the quantum cost of $n-1$ and delay of 1Δ.
- At step 2 the proposed methodology needs n CNOT gates working in series thus this step has the quantum cost of n and delay of $n\Delta$.
- The step 3 needs $n-1$ Toffoli gates working in series thus this step has the quantum cost of $5(n-1)$ and delay of $5(n-1)\Delta$.
- The step 4 needs n Peres gates working in series thus this step has the quantum cost of $4n$ and delay of $4n\Delta$.
- The step 5 needs $n-1$ CNOT gates working in series thus this step has the quantum cost of $n-1$ and delay of $(n-1)\Delta$.
- The step 6 needs $n-1$ CNOT gates working in parallel thus this step has the quantum cost of $n-1$ and delay of 1Δ.

Thus, the total quantum cost of n bit reversible ripple carry adder is $n-1+n+5(n-1)+4n+n-1+n-1 = 13n-8$. The propagation delay will be $1\Delta+n\Delta+5(n-1)\Delta+4n\Delta+(n-1)\Delta+1\Delta = (11n-4)\Delta$. A comparison of the proposed design with the existing designs is illustrated in Table 4.1. In Table 4.1, for [57] the quantum cost and the delay values are valid for $n \geq 2$, and for n=1 the design has the quantum cost of 8 and delay of 8 Δ. Among the existing designs of the reversible ripple carry adder with no input carry, the designs in [57] and [58] are designed with no ancilla input bits and the garbage outputs, while the design presented in [56] has 1 ancilla input and 1 garbage output. In this work we have compared our proposed design of the reversible carry adder with the designs in [56], [57] and [58] for values of n varying from 8 bits to 512 bits. Table 4.2 shows the comparison in terms of quantum cost which shows that the proposed design of the reversible carry adder with no input carry achieves the improvement ratios ranging from 22.5% to 23.51%, 46.36% to 49.95%, and 13.37% to 13.33% compared to the design presented in [56], [57] and [58], respectively. From Table 4.3, it can be seen that the proposed design of reversible ripple carry adder

achieves the improvement ratios ranging from 49.09% to 54.09%, and 13.40% to 15.35% in terms of delay compared to the designs presented in [57] and [58], respectively, while the design presented in [56] is faster than the proposed design by 4.7% to 9.02%.

Table 4.1. A comparison of reversible ripple carry adder with no input carry

	1	2	3	Proposed
Ancilla Inputs	1	0	0	0
Garbage Outputs	1	0	0	0
Quantum Cost	17n-12	26n-29	15n-9	13n-8
Delay Δ	10n	24n-27	13n-7	11n-4
1 is the design in [56] 2 is the design in [57] 3 is the design in [58]				

Table 4.2. Quantum cost comparison of reversible ripple carry adders (no input carry)

Bits	1	2	3	Proposed	% Impr. w.r.t 1	% Impr. w.r.t 2	% Impr. w.r.t 3
8	124	179	111	96	22.5	46.36	13.51
16	260	387	231	200	23	48.32	13.41
32	532	803	471	408	23.3	49.19	13.37
64	1076	1635	951	824	23.42	49.6	13.35
128	2164	3299	1911	1656	23.47	49.8	13.34
256	4340	6627	3831	3320	23.5	49.9	13.33
512	8692	13283	7671	6648	23.51	49.95	13.33
1 is the design in [56] 2 is the design in [57] 3 is the design in [58]							

4.2 Design Methodology of Proposed Reversible Ripple Carry Adder With Input Carry

The reversible ripple carry adder with input carry(c_0) is designed without any ancilla inputs and the garbage outputs, and with less quantum cost and reduced delay compared to the existing design approaches which have optimized the adder design in terms of number of ancilla inputs. Consider the addition of two n bit numbers a_i and b_i stored at memory locations A_i and B_i, respectively, where $0 \leq i \leq n-1$. The input carry c_0 is stored at memory location A_{-1}. Further, consider that memory location A_n is initialized with z $\in \{0, 1\}$. At the end of the computation, the memory location B_i will have s_i, while the location A_i keeps the value a_i for $0 \leq i \leq n-1$. Further, at the end of the computation, the additional location A_n that initially stores the value z will have the

Table 4.3. Delay(in Δ) comparison of reversible ripple carry adders (no input carry)

Bits	1	2	3	Proposed	% Impr. w.r.t 1	% Impr. w.r.t 2	% Impr. w.r.t 3
8	80	165	97	84	-	49.09	13.40
16	160	357	201	172	-	51.8	14.42
32	320	741	409	348	-	53.03	14.91
64	640	1509	825	700	-	53.61	15.15
128	1280	3045	1657	1404	-	53.89	15.26
256	2560	6117	3321	2812	-	54.02	15.32
512	5120	12261	6649	5628	-	54.09	15.35
1 is the design in [56] 2 is the design in [57] 3 is the design in [58]							

value $z \oplus s_n$, and the memory location A_{-1} keeps the input carry c_0. Thus A_n will have the value of s_n when z=0. Here, s_i is the sum bit produced and is defined as:

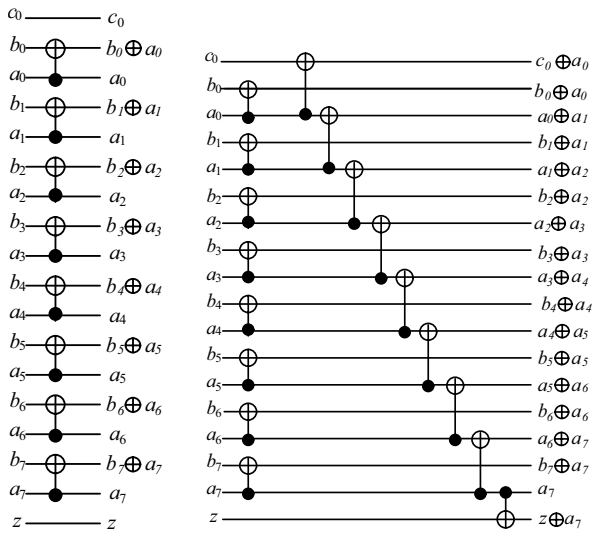
$$s_i = \begin{cases} a_i \oplus b_i \oplus c_i & \text{if } 0 \leq i \leq n-1 \\ c_n & \text{if } i = n \end{cases}$$

where c_i is the carry bit and is defined as:

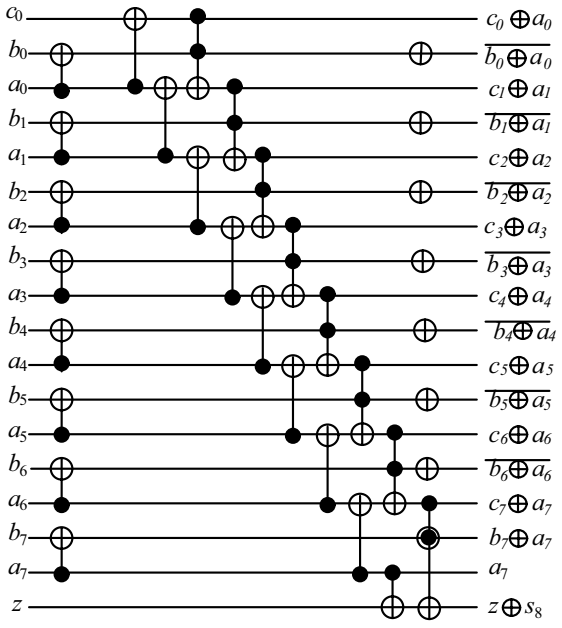
$$c_i = \begin{cases} c_0 & \text{if } i = 0 \\ a_{i-1}b_{i-1} \oplus b_{i-1}c_{i-1} \oplus c_{i-1}a_{i-1} & \text{if } 1 \leq i \leq n \end{cases}$$

As shown above c_i is the carry bit and is generated by using a_{i-1}, b_{i-1} and c_{i-1}. In our proposed approach firstly all the carry bits are generated and are saved in the memory location A_{i-1} which was initially used for storing a_{i-1} for $1 \leq i \leq n-1$. Once the generated carry bits are used, the location A_i are restored to the value a_i while the location B_i will have s_i for $0 \leq i \leq n-1$. Thus restoring of location A_i to the value a_i helps in minimizing the garbage outputs. Since no constant input having the value as 0 is needed in the proposed approach, it saves the ancilla inputs. The details of the proposed approach to minimize the garbage outputs and the ancilla inputs can be understood by following the steps of the proposed design methodology.

The proposed method improves the delay and the quantum cost by selectively using the Peres gate and the TR gate at the appropriate places. The generalized methodology of designing the n bit reversible ripple carry adder with input carry is explained below along with an illustrative example of 8 bit reversible ripple carry adder. The illustrative example of 8 bit reversible ripple carry adder

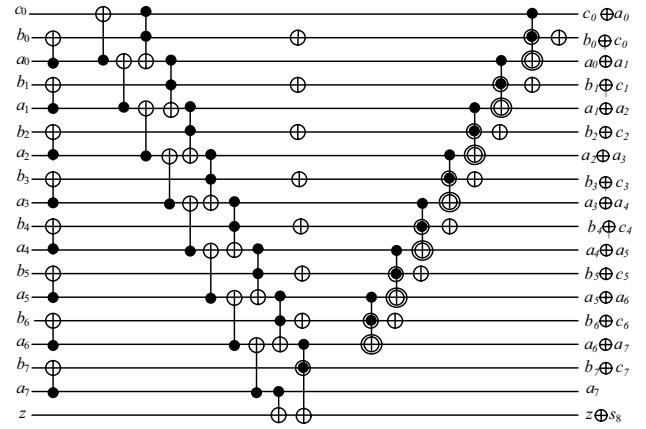


(a) After Step 1 (b) After Step 2

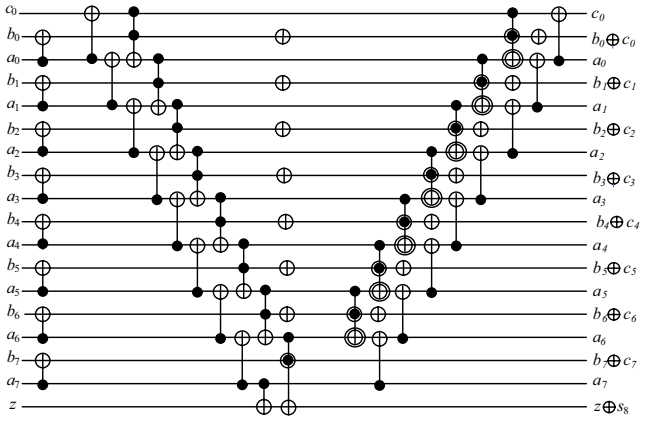


(c) After Step 3

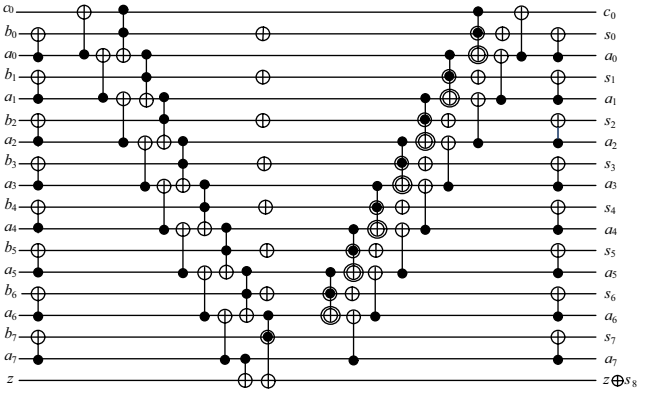
Figure 4.4. Circuit generation of reversible 8 bit adder with input carry: Steps 1-3



(a) After Step 4



(b) After Step 5



(c) After Step 6

Figure 4.5. Circuit generation of reversible 8 bit adder with input carry: Steps 4-6

is shown in Figs. 4.4 and 4.5 that can perform the addition of two 8 bit numbers a=$a_0...a_7$ and b=$b_0...b_7$, and has the input carry c_0. The proposed methodology will be referred as methodology 2 further in this work and is explained in the following steps:

Step 1: For i=0 to n-1:

At pair of locations A_i and B_i apply the CNOT gate such that the location A_i will maintain the same value, while location B_i transforms to the value $*A_i \oplus *B_i$, where $*A_i$ and $*B_i$ represent the values stored at location A_i and B_i. For illustrative purpose, the circuit of reversible ripple carry adder with input carry c_0 after step 1 is shown for addition of 8 bit numbers in Fig.4.4(a).

Step 2: For i= -1 to n-2:

At pair of locations A_{i+1} and A_i apply the CNOT gate such that the location A_{i+1} will maintain the same value, while the value at location A_i transforms to $*A_{i+1} \oplus *A_i$. Further, apply a CNOT gate at pair of locations A_{n-1} and A_n such that the value at location A_{n-1} will remain same, while the value at location A_n transforms to $*A_{n-1} \oplus *A_n$. The reversible 8 bit adder circuit after the step 2 is illustrated in Fig. 4.4(b).

Step 3: The step 3 has the following sub-steps:

- For i=0 to n-2:

  At locations A_{i-1}, B_i and A_i apply the Toffoli gate such that A_{i-1}, B_i and A_i are passed to the inputs A, B, C, respectively, of the Toffoli gate. Apply a Peres gate at location A_{n-2}, B_{n-1} and A_n such that A_{n-2}, B_{n-1} and A_n are passed to the inputs A, B, C, respectively, of the Peres gate.

- For i=0 to n-2: Apply a NOT gate at location B_i.

The reversible 8 bit adder circuit based on the proposed design methodology after the step 3 is illustrated in Fig. 4.4(c)

Step 4: The step 4 has the following two sub-steps:

- For i=n-2 to 0:

  At locations A_{i-1}, B_i and A_i apply the TR gate such that A_{i-1}, B_i and A_i are passed to the inputs A, B, C, respectively, of the TR gate.

- For i=0 to n-2: Apply a NOT gate at location B_i.

The reversible 8 bit adder circuit based on the proposed methodology after step 4 is illustrated in Fig. 4.5(a).

Step 5: For i=n-1 to 0:

At pair of locations A_i and A_{i-1} apply the CNOT gate such that the location A_i will maintain the same value, while value at location A_{i-1} transforms to the value $*A_i \oplus *A_{i-1}$. The reversible 8 bit adder circuit after the step 5 is shown in Fig.4.5(b).

Step 6: For i=0 to n-1:

At pair of locations A_i and B_i apply the CNOT gate such that the location A_i will maintain the same value, while the value at location B_i transforms to the value $*A_i \oplus *B_i$. After this step we will have the complete working design of the reversible adder an example of which is shown for addition of 8 bit numbers in Fig.4.5(c).

Thus, the proposed methodology is able to design the reversible ripple carry adder with an input carry without any ancilla and garbage bits. As in the design of the reversible BCD adder, the 4 bit reversible ripple carry adder will be used, thus its design is also illustrated in Fig.4.6

Theorem 2: *Let a and b are two n bit binary numbers represented as a_i and b_i, c_0 is the input carry (c_0), and z $\in \{0, 1\}$ is the another 1 bit input, where $0 \leq i \leq n-1$, then the proposed design steps of methodology 2 result in the ripple carry adder circuit that works correctly. The proposed design methodology designs an n bit adder circuit that produces the sum output s_i at the memory location where b_i is initially stored, while the location where a_i is initially stored is restored to the value a_i for $0 \leq i \leq n-1$. Further, the memory location where z is initially stored transforms to*

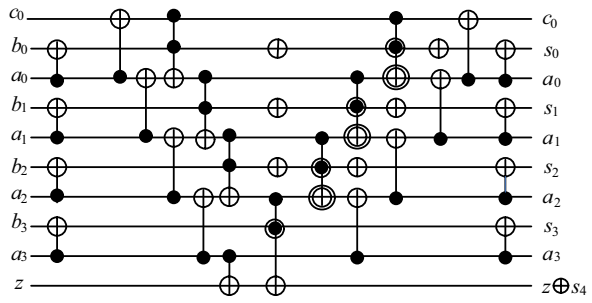


Figure 4.6. Proposed reversible 4 bit adder with input carry

$z \oplus s_n$, and the memory location where the input carry c_0 is initially stored is restored to the value c_0.

Proof: The proposed approach will make the following changes on the inputs that are illustrated as follows:

Step 1: The step 1 of the proposed approach transforms the input states to

$$|c_0\rangle \left(\bigotimes_{i=0}^{n-1} |b_i \oplus a_i\rangle |a_i\rangle \right) |z\rangle$$

For illustrative purpose, the transformation of the input states of a 8 bit reversible adder circuit after step 1 is shown in Fig.4.4(a).

Step 2: The step 2 of the proposed approach transforms the input states to

$$|c_0 \oplus a_0\rangle \left(\bigotimes_{i=0}^{n-2} |b_i \oplus a_i\rangle |a_i \oplus a_{i+1}\rangle \right) |b_{n-1} \oplus a_{n-1}\rangle |a_{n-1}\rangle |z \oplus a_{n-1}\rangle$$

For illustrative purpose, the transformation of the input states of a 8 bit reversible adder circuit after step 2 is shown in Fig.4.4(b).

Step 3: The step 3 has the two sub-steps:

Step 3.a has the Toffoli gates which take the inputs as $c_i \oplus a_i$, $b_i \oplus a_i$, and $a_i \oplus a_{i+1}$ for $0 \leq i \leq n-2$. The Toffoli gates will produce the outputs as $c_i \oplus a_i$, $b_i \oplus a_i$, and $c_{i+1} \oplus a_{i+1}$. The third outputs of the Toffoli gates are $c_{i+1} \oplus a_{i+1}$ because of the fact that the Toffoli gate has the logic equation as $A \cdot B \oplus C$ where A, B and C are the inputs of a Toffoli gate, thus will

produce the output as $c_i \oplus a_i \cdot b_i \oplus a_i \oplus a_i \oplus a_{i+1}$= $c_{i+1} \oplus a_{i+1}$. Finally we have a Peres gate having the inputs $c_{n-1} \oplus a_{n-1}$, $b_{n-1} \oplus a_{n-1}$, and $a_{n-1} \oplus z$ which produces the outputs as $c_{n-1} \oplus a_{n-1}$, $c_{n-1} \oplus b_{n-1}$, and $s_n \oplus z$. Thus after the step 3.a the input states are transformed to:

$$|c_0 \oplus a_0\rangle \left(\bigotimes_{i=0}^{n-2} |b_i \oplus a_i\rangle \, |c_{i+1} \oplus a_{i+1}\rangle \right) |b_{n-1} \oplus c_{n-1}\rangle \, |a_{n-1}\rangle \, |z \oplus s_n\rangle$$

The step 3.b applies the NOT operation to the location B_i having the value as $b_i \oplus a_i$ for $0 \leq i \leq n-2$, thus the input states are transformed to

$$|c_0 \oplus a_0\rangle \left(\bigotimes_{i=0}^{n-2} |\overline{b_i \oplus a_i}\rangle \, |c_{i+1} \oplus a_{i+1}\rangle \right) |b_{n-1} \oplus c_{n-1}\rangle \, |a_{n-1}\rangle \, |z \oplus s_n\rangle$$

For illustrative purpose, the transformation of the input states of a 8 bit reversible adder circuit after step 3 is shown in Fig.4.4(c).

Step 4: The step 4 has the TR gates which take the inputs as $\{c_i \oplus a_i$, $\overline{b_i \oplus a_i}$, and $c_{i+1} \oplus a_{i+1}$ for i=n-2 to 0. The TR gates will produce the outputs as $c_i \oplus a_i$, $\overline{b_i \oplus c_i}$, and $a_i \oplus a_{i+1}$ for i=n-2 to 0. Thus after the application of TR gates the input states transform to:

$$|c_0 \oplus a_0\rangle \left(\bigotimes_{i=0}^{n-2} |\overline{b_i \oplus c_i}\rangle \, |a_i \oplus a_{i+1}\rangle \right) |b_{n-1} \oplus c_{n-1}\rangle \, |a_{n-1}\rangle \, |z \oplus s_n\rangle$$

Next, the NOT gates are applied to the TR gates outputs $\overline{b_i \oplus c_i}$ for i=n-2 to 1. Thus the step 4 of the proposed approach transforms the input states to

$$|c_0 \oplus a_0\rangle \left(\bigotimes_{i=0}^{n-2} |b_i \oplus c_i\rangle \, |a_i \oplus a_{i+1}\rangle \right) |b_{n-1} \oplus c_{n-1}\rangle \, |a_{n-1}\rangle \, |z \oplus s_n\rangle$$

For illustrative purpose, the transformation of the input states of a 8 bit reversible adder circuit after step 4 is shown in Fig.4.5(a).

Step 5: The step 5 of the proposed approach transforms the input states to

$$|c_0\rangle \left(\bigotimes_{i=0}^{n-2} |b_i \oplus c_i\rangle \, |a_i\rangle \right) |b_{n-1} \oplus c_{n-1}\rangle \, |a_{n-1}\rangle \, |z \oplus s_n\rangle$$

For illustrative purpose, the transformation of the input states of a 8 bit reversible adder circuit after step 5 is shown in Fig.4.5(b).

Step 6: The step 6 of the proposed approach transforms the input states to

$$|c_0\rangle \left(\bigotimes_{i=0}^{n-1} |s_i\rangle \, |a_i\rangle \right) |z \oplus s_n\rangle$$

For illustrative purpose, the transformation of the input states of a 8 bit reversible adder circuit after step 6 is shown in Fig.4.5(c).

Thus we can see that the proposed six step will produce the sum output s_i at the memory location where b_i is stored initially, while the location where a_i is stored initially will be restored to the value a_i for $0 \leq i \leq n-1$. The memory location where z is stored will have $z \oplus s_n$ and the memory location where the input carry c_0 was stored initially will be restored to the value c_0.This proves the correctness of the proposed methodology of designing the reversible ripple carry adder with input carry.

4.2.1 Delay and Quantum Cost

- The step 1 of the proposed methodology needs n CNOT gates working in parallel thus this step has the quantum cost of n and delay of 1Δ.
- The step 2 of the proposed methodology needs $n+1$ CNOT gates working in series thus this step has the quantum cost of $n+1$. The delay of this stage will be only 2Δ as it has $n-1$ CNOT gates work in parallel with the Toffoli gates of the next stage thus only 2 CNOT gates contributes to the delay.
- The step 3 needs $n-1$ Toffoli gates working in series thus contributing to the quantum cost of $5(n-1)$ and delay of $5(n-1)$ Δ. There is a Peres gate contributing to the quantum cost of 4 and delay of 4Δ. There are $n-1$ NOT gates working in parallel with the Peres gate thus contributing to quantum cost of $n-1$ and zero delay. The total quantum cost of this stage is $5(n-1)+4+n-1$ while the delay contribution of this stage is $5(n-1)\Delta + 4\Delta$.
- The step 4 needs $n-1$ TR gates working in series thus contributing to the quantum cost by $4(n-1)$ and delay of $4(n-1)\Delta$. Further, there are $n-1$ NOT gates, which all work in

parallel with the TR gates except the last NOT gate. Thus, it contributes to quantum cost of $n-1$ and delay of 1Δ. Thus this step has the quantum cost of $4(n-1)+n-1$ and the delay of $4(n-1)\Delta$ +1 Δ.

- The step 5 needs n CNOT gates working in parallel with the TR gates and the NOT gates, except the last one. Thus this step has the quantum cost of n and delay of 1Δ.

- The step 6 needs n CNOT gates working in parallel thus this step has the quantum cost of n and delay of 1Δ.

Thus the total quantum cost of n bit reversible ripple carry adder is $n+n+1+5(n-1)+4+n-1+4(n-1)+n-1+n+n=15n-6$. The propagation delay will be $1\Delta+2\Delta+5(n-1)\Delta+4\Delta+4(n-1)\Delta+1\Delta+1\Delta+1\Delta=(9n+1)\Delta$.

A comparison of the proposed design with the existing designs is illustrated in Table 4.4 which shows that the proposed design of reversible ripple carry adder with input carry is designed with no ancilla input bit and has less quantum cost and delay compared to its existing counterparts. Table 4.5 shows the comparison in terms of quantum cost which shows that the proposed design of the reversible carry adder with input carry achieves the improvement ratios ranging from 12.3% to 11.77% and 0% to 11.61% compared to the designs presented in [56]. From Table 4.6, it can be seen that the proposed design of reversible ripple carry adder achieves the improvement ratios ranging from 10.97% to 10.01%, and 0% to 9.83% in terms of delay compared to the designs presented in [56], respectively.

Table 4.4. A comparison of reversible ripple carry adder with input carry

	1	2	Proposed
Ancilla Inputs	0	0	0
Garbage Outputs	0	0	0
Quantum Cost	17n-6	17n-22	15n-6
Delay Δ	10n+2	10n-8	9n+1
1 is the design 1 in [56] 2 is the design 2 in [56]			

Table 4.5. Quantum cost comparison of reversible ripple carry adders (with input carry)

Bits	1	2	Proposed	% Impr. w.r.t 1	% Impr. w.r.t 2
8	130	114	114	12.30	-
16	266	250	234	12.03	6.4
32	538	522	474	11.89	9.19
64	1082	1066	954	11.82	10.50
128	2179	2154	1914	11.79	11.14
256	4346	4330	3834	11.78	11.45
512	8698	8682	7674	11.77	11.61
1 is the design 1 in [56] 2 is the design 2 in [56]					

Table 4.6. Delay (in Δ) comparison of reversible ripple carry adders (with input carry)

Bits	1	2	Proposed	% Impr. w.r.t 1	% Impr. w.r.t 2
8	82	72	73	10.97	-
16	162	152	145	10.49	4.6
32	322	312	289	10.24	7.37
64	642	632	577	10.12	8.7
128	1282	1272	1153	10.06	9.35
256	2562	2552	2305	10.03	9.67
512	5122	5112	4609	10.01	9.83
1 is the design 1 in [56] 2 is the design 2 in [56]					

4.3 Design of Reversible Subtractor Circuit Based on Reversible Adder With No Input Carry

An n bit reversible subtractor can be designed based on the n bit reversible adder using the property that $a - b$=$\overline{\bar{a} + b}$. Using this approach $a - b$=$\overline{\bar{a} + b}$, an n bit subtractor can be designed by using the n bit ripple carry adder that performs the operation $a + b$. As shown above in Section 4.1, we have proposed the efficient design of the n bit ripple carry adder with no input carry that performs the addition of two n bit numbers a and b without any ancilla input and the garbage output, and has the quantum cost of $13n - 8$ and delay of $(11n - 9)$ Δ. The design of 4 bit reversible ripple carry adder with no input carry is illustrated in Fig.4.3. From Fig.4.3 it can be observed that the proposed ripple carry adder transforms the input b to the sum output while the input a is regenerated at the output. The reversible n bit subtractor can be designed based on the proposed ripple carry adder by complementing the input a at the start, and finally complementing a and the sum produced at the end. The design of the proposed n bit reversible subtractor based on the proposed reversible ripple

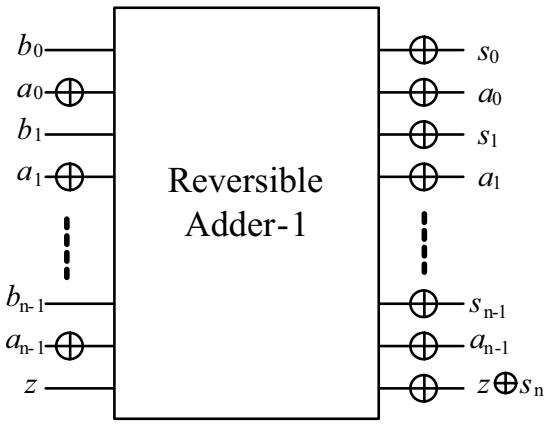


Figure 4.7. Proposed reversible n bit subtractor based on reversible ripple carry adder with no input carry

carry adder with no input carry is illustrated in Fig.4.7. The proposed n bit reversible subtractor has the quantum cost of $13n - 8$+$3n$=$16n - 8$ while the delay is $11n - 9$+n+n=$(13n - 9)$ Δ.

4.4 Design of Reversible Subtractor Circuit Based on Reversible Adder With Input Carry

Another possible approach to design an n bit subtractor that can subtract two n bit numbers a and b is to use the property $a - b$=$a + \bar{b} + 1$. Thus, an n bit subtractor can be designed by using the n bit ripple carry adder with input carry that performs the operation $a + b + c_0$ where a and b are n bit numbers and c_0 is the input carry hardwired as $c_0 = 1$. As shown above in Section 4.2, we have proposed an efficient design of the n bit ripple carry adder with input carry that performs the addition of two n bit numbers a and b and input carry c_0. For n bit addition, the approach presented designs the n bit reversible ripple carry adder with input carry with quantum cost of $15n - 6$, while the propagation delay of the design is $(9n + 1)\Delta$. The design of 4 bit reversible ripple carry adder with input carry based on the proposed approach is illustrated in Fig.4.6. From Fig.4.6 it can be observed that the proposed ripple carry adder transforms the input b to the sum output while the input a and carry input c_0 is regenerated at the outputs. The reversible n bit subtractor circuit based on the ripple carry adder with input carry can be designed by complementing the input b at the start and hardwiring the input $c_0 = 1$. The design of the proposed n bit reversible subtractor

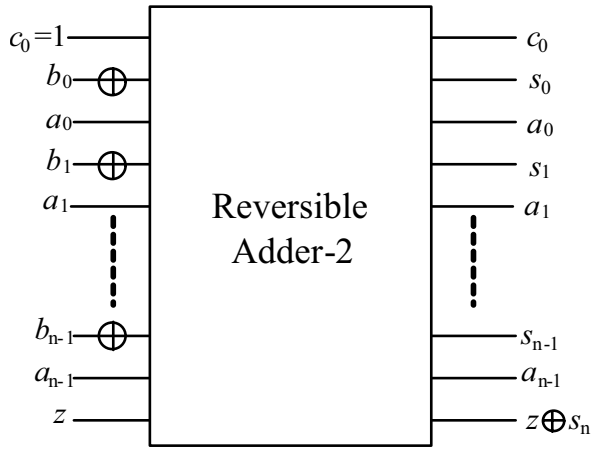


Figure 4.8. Proposed reversible n bit subtractor based on reversible ripple carry adder with input carry

based on the ripple carry adder with input carry is illustrated in Fig.4.8. The proposed design of the n bit reversible subtractor has the quantum cost of $15n - 6$+n=$16n - 6$, while the delay is $(9n + 1)$+1=$(9n + 2)$ Δ.

4.5 Comparison of n Bit Reversible Subtractor

As illustrated in Section 3.5, the first design of the n bit reversible subtractor based on the conventional ripple borrow approach has the quantum cost of 6n-2, delay of 4n Δ, n-1 ancilla inputs and n-1 garbage outputs. The second design of the n bit reversible subtractor based on proposed reversible ripple carry adder with no input carry has the quantum cost of $16n - 8$, delay of $13n - 9$ Δ, 0 ancilla inputs and 0 garbage outputs. The third design of the n bit reversible subtractor based on the reversible ripple carry adder with input carry has the quantum cost of $16n-6$, delay of $9n+2$ Δ, 1 ancilla input and 1 garbage output. Further, in the existing literature there are designs of reversible n bit subtractor which are based on property $a - b$=$\overline{\bar{a} + b}$. The n bit reversible subtractor proposed in [56] has 1 ancilla input, 1 garbage output, quantum cost of 20n-12 and delay of 12n Δ. The n bit reversible subtractor proposed in [57] has the quantum cost of 29n-29 and delay of (26n-27) Δ and is designed without any ancilla input and garbage output. The reversible subtractor based on adder

Table 4.7. A comparison of n bit reversible ripple borrow subtractors

	Ancilla input	Garbage outputs	Quantum cost	Delay Δ
This work 1	n-1	n-1	6n-3	6n-3
This work 2	0	0	16n-8	13n-9
This work 3	1	1	16n-6	9n+2
Cuccaro et al. 2004 [56]	1	1	20n-12	12n
Takahashi and Kunihiro 2005 [57]	0	0	29n-29	26n-27
Takahashi et al. 2009 [58]	0	0	18n-9	15n-7

1 represents proposed design of the proposed n bit reversible subtractor based on conventional ripple borrow approach presented in Section 3.5

2 represents proposed design of the proposed n bit reversible subtractor based on reversible ripple carry adder with no input carry presented in Section 4.1

3 represents proposed design of the proposed n bit reversible subtractor based on reversible ripple carry adder with input carry presented in Section 4.2

1 is the most efficient design in terms of quantum cost and the delay but has overhead in terms of ancilla inputs and the garbage outputs

2 has only 0 ancilla input and 0 garbage output and is better than its existing counterparts in [57] and [58] (which are also designed with 0 ancilla input and 0 garbage output) in terms of quantum cost and the delay

3 has only 1 ancilla input and 1 garbage output and is better than its existing counterpart in [56] (which is also designed with 1 ancilla input and 1 garbage output) in terms of quantum cost and the delay

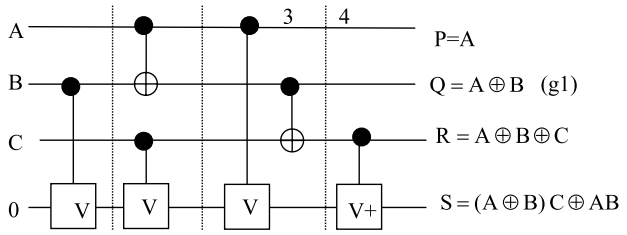
presented in [58] has the quantum cost of 18n-9 and delay of (15n-7) Δ and is also designed without any ancilla input and the garbage output. Table 4.7 summarizes the comparison of all reversible n bit subtractor. The comparison shows the proposed designs of the n bit reversible subtractor excels the existing design of reversible subtractor in various parameters. The proposed first design that is based on the conventional ripple borrow approach is most efficient in terms of delay and quantum cost while sacrificing the ancilla inputs and the garbage outputs. The proposed design based on reversible ripple carry adder with input carry having 1 ancilla input and 1 garbage output is better than its existing counterpart in [56] in terms of quantum cost and delay. The proposed design based on reversible ripple carry adder with no input carry and without any ancilla inputs and the garbage outputs is better than its existing counterparts proposed in [57] and [58] in terms of quantum cost and delay.

4.6 Design of Unified Reversible Adder-Subtractor

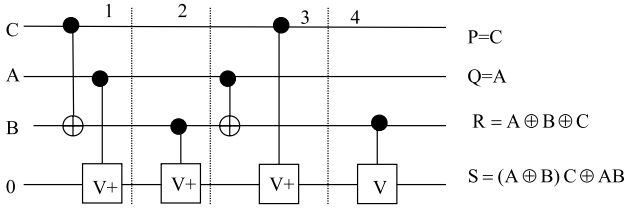
In this section, we discuss the design of unified reversible adder-subtractor that can work as an adder as well as a subtractor depending on the value of the control signal (*ctrl*).

4.6.1 Design of Reversible Adder-Subtractor Based on Conventional Ripple Carry/Borrow Approach

Before presenting the design of an n bit reversible adder-subtractor we propose a new design of the 1 bit reversible full adder that forms the basic building block in the complete design of the adder-subtractor. In the existing literature the design of 1 bit full adder presented in [106] is one of the most efficient design in terms of delay. As shown in Fig.4.9(a), the design presented in [106] has the sum and carry outputs produced at R and S outputs, respectively. The reversible full adder in [106] has delay of 4 Δ, quantum cost of 6, needs 1 ancilla input and 1 garbage output (g1). The output Q that produces the function Q=$A \oplus B$ is considered as the garbage output. Even though the design in [106] is efficient in terms of delay, quantum cost and the ancilla input, it has the drawback of producing a garbage output. Thus, in this work we present a new efficient design of a 1 bit reversible full adder with quantum cost of 6, delay of 4 Δ, 1 ancilla input and without any garbage output as illustrated in Fig.4.9(b). Thus, the proposed design of the 1 bit reversible full adder is better than the existing design presented in [106]. The output P=C is not considered as the garbage output because in the full adder inputs A,B and C can be exchanged with each other without affecting the functionality of the design. Thus in Fig.4.9(b) by passing input B in place of input C, the output P will have value P=B and will not be considered as a garbage output (the regenerated inputs are not considered as the garbage outputs [34]). Once we have an efficient design of 1 bit reversible full adder (RFA), an n bit unified adder-subtractor can be designed as illustrated in Fig.4.10. The design uses n 1 bit reversible carry adder (RFA) cascaded as a chain and functions as an adder or a subtractor depending on the values passed by n CNOT gates controlled by the signal labelled as *ctrl*. When the signal *ctrl=0* the design works as an n bit reversible adder because the CNOT gates pass the input b in its normal form. When *ctrl=1*, the design works as an n bit reversible subtractor because the value of the input b is complemented by the CNOT gates thus performing the operation $a - b$=$a + \bar{b} + 1$. The proposed design of unified reversible adder-subtractor based on conventional ripple carry/borrow approach has the quantum cost of 7n+1 and delay of (4n+1) Δ, while there are n+1 ancilla inputs and no garbage outputs.



(a) Design of 1 bit reversible full adder proposed in [106] with 1 garbage output



(b) Proposed design of 1 bit reversible full adder without any garbage output

Figure 4.9. Designs of 1 bit reversible full adder

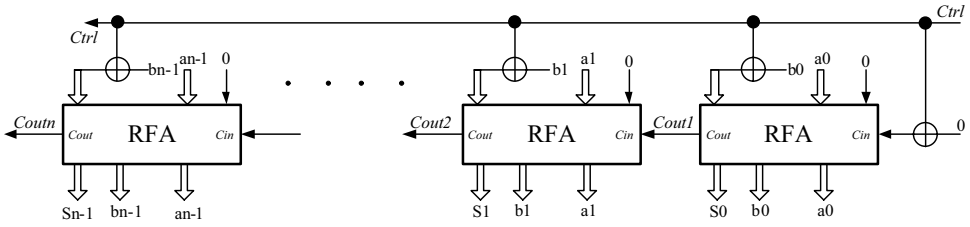


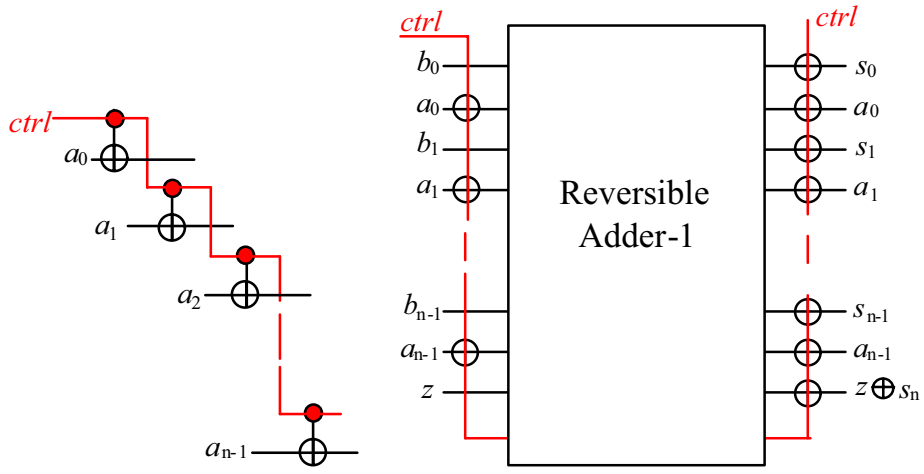
Figure 4.10. Proposed design of n bit reversible adder-subtractor based on conventional ripple carry/borrow approach

4.6.2 Design of Reversible Adder-Subtractor Based on Reversible Ripple Carry Adder With No Input Carry

As discussed in Section 4.3, the second design of an n bit reversible subtractor uses the property $a - b$=$\overline{\bar{a} + b}$. The design of n bit reversible subtractor based on reversible ripple carry adder with no input carry is shown earlier in Fig.4.7. The design of the n bit reversible full subtractor shown in Fig.4.7 can be converted to an n bit adder-subtractor by using a control signal that controls the complementing of the input a at the start, and also the complementing of the input a and the output sum produced at the end. An example of this strategy that shows how CNOT gates can be used for controlling the signals is illustrated in Fig. 4.11(a). Fig.4.11(a) illustrates that if *ctrl=1* the design will complement the controlled input a otherwise the controlled input is passed as such. Using this strategy of using the control signal *ctrl* to define the add or subtract operation, the proposed design of an n bit reversible adder-subtractor is illustrated in Fig. 4.11(b). As the CNOT gate has the quantum cost of 1 and delay of 1 Δ, the proposed design of n bit reversible adder-subtractor based on ripple carry adder with no input carry has quantum cost and delay as same as reversible n bit subtractor based on reversible ripple carry adder with no input carry. The quantum cost of n bit reversible adder-subtractor using reversible ripple carry adder with no input carry is $13n - 8$+$3n$=$16n - 8$ while the delay is $11n - 9$+n+n=$(13n - 9)$ Δ.

4.6.3 Design of Reversible Adder-Subtractor Based on Reversible Ripple Carry Adder With Input Carry

An n bit subtractor designed by using the n bit ripple carry adder with input carry that performs the operation $a + b + c_0$ where a and b are n bit number and c_0 is the input carry hardwired as $c_0 = 1$ is illustrated earlier in Fig.4.8. The design of the n bit reversible full subtractor based on reversible ripple carry adder with input carry as shown in Section 4.4 can be converted to an n bit adder-subtractor by using a control signal *ctrl* that controls the complementing of the input b at the start and can also add 1 to the adder to perform the n bit subtraction. The design of n bit adder-subtractor based on reversible ripple carry adder with input carry is illustrated in Fig. 4.10. As illustrated in Fig. 4.10 if *ctrl=1* the design will complement the controlled input b and add 1 and will work as



(a) CNOT gates based controlling of the signals

(b) Proposed design of n bit reversible adder-subtractor based on reversible ripple carry adder with no input carry

Figure 4.11. Illustration of CNOT gates based controlling of inputs, and the design of n bit reversible adder-subtractor based on reversible ripple carry adder with no input carry.

n bit reversible subtractor. Otherwise when *ctrl=0* the controlled input b is passed as such and the design will work as n bit reversible full adder. The proposed n bit reversible adder-subtractor has the quantum cost and delay as same as n bit reversible subtractor based on reversible ripple carry adder with input carry and is designed without any ancilla input and the garbage output.

4.6.4 Comparison of n Bit Reversible Adder-Subtractor

As discussed above in Section 4.6.1, the n bit reversible adder-subtractor based on conventional ripple carry/borrow approach has the quantum cost of 7n+1, delay of (4n+1) Δ, n+1 ancilla inputs and 0 garbage outputs. As shown in Section 4.6.2, the design of n bit reversible adder-subtractor based on reversible ripple carry adder with no input carry has the quantum cost of $16n - 8$, delay of $(13n - 9)$ Δ, 0 ancilla inputs and 0 garbage outputs. As shown in Section 4.6.3, the design of n bit reversible adder-subtractor based on reversible ripple carry adder with input carry has the quantum cost of $16n - 6$, delay of $(9n + 2)$ Δ, 0 ancilla inputs and 0 garbage outputs. Thus the design of the reversible adder-subtractor based on conventional ripple carry/borrow approach is most efficient in terms of delay. The results shows that the design of the reversible adder-subtractor based on

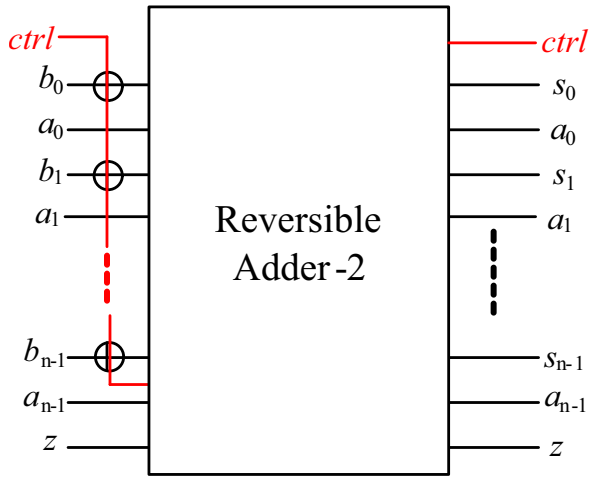


Figure 4.12. Proposed reversible n bit adder-subtractor based on reversible ripple carry adder with input carry

Table 4.8. A comparison of n bit reversible adder-subtractors

	Ancilla input	Garbage outputs	Quantum cost	Delay Δ
This work 1	n+1	0	7n+1	4n+1
This work 2	0	0	16n-8	13n-9
This work 3	0	0	16n-6	9n+2

1 represents proposed design of the n bit reversible adder-subtractor based on conventional ripple carry/borrow approach as shown in Section 4.6.1

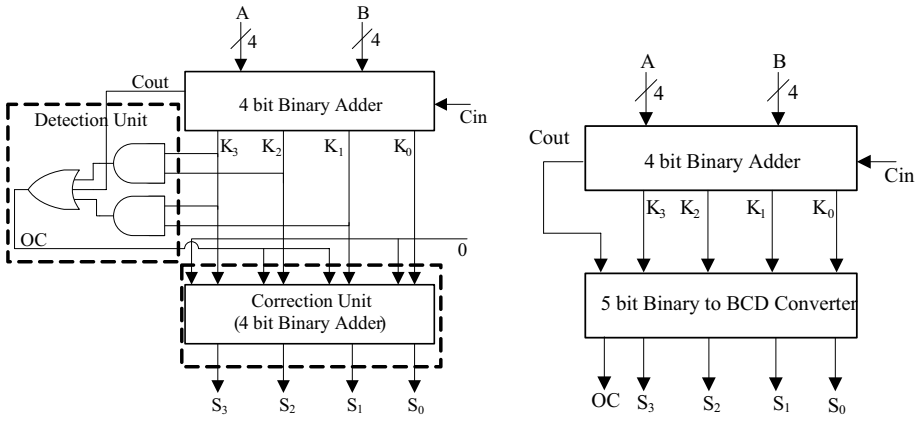
2 represents proposed design of the n bit reversible adder-subtractor based on reversible ripple carry adder with no input carry as shown in Section 4.6.2

3 represents proposed design of the n bit reversible adder-subtractor based on reversible ripple carry adder with input carry as shown in Section 4.6.3

the reversible ripple carry adder with input carry is most efficient as it has 0 ancilla inputs and 0 garbage outputs and has less quantum cost and delay compared to the design based on approach 2. The results shows that the design of the reversible adder-subtractor based on the reversible ripple carry adder with input carry will provide an efficient way of designing an n bit reversible adder-subtractor. All the results are summarized in Table 4.8

4.7 Design of Reversible BCD Adder

A BCD adder is a circuit that adds two BCD digits in parallel and produces a sum digit, also in BCD. We are illustrating two different approaches of designing the conventional BCD adder.



(a) Approach 1 for design of conventional 1 digit BCD adder

(b) Approach 2 for design of conventional 1 digit BCD adder

Figure 4.13. Design approaches of conventional BCD adder

4.7.1 Basics

Figure 4.13(a) shows the first approach of designing the 1 digit conventional BCD adder, which also includes the detection and the correction logic in its internal construction. The two decimal digits A and B, together with the input carry Cin, are first added in the top 4-bit binary adder to produce the 4 bit binary sum (K_3 to K_0) and the carry out (Cout). In the BCD addition, when the binary sum of A and B is less than 1001, the BCD number is same as the binary number thus no conversion is needed. But when the binary sum of A and B is greater than 1001, 0110 is added to convert the binary number into an equivalent BCD number. The condition that summation of numbers A, B and Cin is greater than 1001 is detected through a detection unit. The detection unit works on the condition that can be expressed by the Boolean function $OC = Cout + K_3 \cdot K_2 + K_3 \cdot K_1$. When OC (output carry) is equal to zero, nothing is added to the binary sum. When it is equal to one, binary 0110 is added to the binary sum using the correction unit (another 4-bit binary adder).

As illustrated above in Fig.4.13(a), the binary adder produces a result that may not be in correct BCD format and need to be converted to BCD format through the use of detection and correction unit. Instead of using the detection and the correction unit to convert the result of summation to

the BCD format, the outputs of the binary adder can be passed to a binary to BCD converter to have the result of the binary addition in the BCD format [4]. This approach is illustrated in the Fig. 4.13(b) where the 5 bit binary to BCD converter produces the desired output in the BCD format. In this work, we have proposed the equivalent reversible design of these two approaches to design the reversible BCD adder optimized for the number of ancilla input bits and the number of garbage outputs.

4.7.2 Design of Reversible BCD Adder Based on Approach 1

We present two designs of reversible BCD adders based on approach 1 with and without input carry c_0 (the conventional irreversible design of the 1 digit BCD adder based on approach 1 is illustrated in Fig. 4.13(a)).

4.7.2.1 Design 1 of Reversible BCD adder With Input Carry

As can be observed from Fig.4.13(a) that in order to have this design, we need the design of 4 bit reversible adder with input carry, the design of which is already illustrated in Fig.4.6. Next we present the new reversible design of the detection unit. As illustrated above, the detection unit uses the Boolean function $OC = Cout + K_3 \cdot K_2 + K_3 \cdot K_1$ as the checking condition. It can be written as $OC = Cout + K_3(K_2 + K_1)$. On careful observation it can be reduced to $OC = Cout \oplus K_3(K_2 + K_1)$ as Cout and $K_3(K_2 + K_1)$ cannot be true at the same time [39]. We have designed the reversible detection unit based on the modified Boolean equation $OC = Cout \oplus K_3(K_2 + K_1)$ using NOT, CNOT, the Peres gate and the TR gate. Before explaining the reversible design of the detection unit, we would like to emphasize a very useful property of the TR gate in relation to the popular Peres gate. We derive the inverse of the TR gate since a reversible gate can be combined with its inverse reversible gate to minimize the garbage outputs [34]. In order to derive the logic equations of the inverse TR gate, we performed the reverse mapping of the TR gate outputs working as inputs to generate the inputs of the TR gate. We observe that the inverse of the TR gate is same as the existing Peres gate having inputs to outputs mapping as (P=A, Q=A $\oplus$ B, $R = A \cdot B \oplus C$. Thus, the TR gate and the Peres gate are inverse of each other.

The design of the reversible detection unit is illustrated in Fig.4.14(a) in which by using the TR gate as the inverse of the Peres gate we are able to regenerate K_1, K_2, K_3 to be used further in the correction unit. Further, the ancilla input bit having the constant value as '0' is regenerated to be used further in the correction unit. In the design, firstly with the help of the NOT gate and the Peres gate the output $\bar{K_1} \cdot \bar{K_2}$ is generated which is passed to the TR gate to generate the output $OC = Cout \oplus K_3(K_2 + K_1)$. Then with the help of the CNOT gate cascaded at the inputs A and B of the TR gate, the function $\bar{K_1} \cdot \bar{K_2}$ is regenerated. Finally, TR gate combined with NOT gates is used to regenerate $K_1, K_2, 0$ outputs (here the final TR gate works as the inverse of the Peres gate). The reversible detection unit has the quantum cost of 17 and the delay of 15 Δ.

The design of the reversible correction unit is illustrated in Fig.4.14(b). The design of the correction unit is a 2 bit binary adder based on the methodology of the proposed reversible ripple carry adder without input carry as illustrated earlier in section IV. In the design the 2 bits inputs of the adders are $a_0 = K_1, b_0 = OC$, $a_1 = K2, b_1 = OC$. In order to generate S_3 we have passed OC at the location z (a_3) of the reversible ripple carry adder as S3 can be generated as $K_3 \oplus C_3$. Since 0 needs to be added to K_0 to produce S_0, thus S_0 will be same as K_0 and hence a wire connection. The proposed design of the reversible correction unit does not need any ancilla input bit. The reversible correction unit has the quantum cost of 16 and delay of 16 Δ. The modules designed above can be integrated together to design the 1 digit reversible BCD adder as illustrated in Fig.4.15. The design Fig.4.15 will be used with name RBCD-1 further in this work. The proposed design contains the 4 bit reversible adder with input carry, reversible detection unit and reversible correction unit. A Feynman gate is used to avoid the fanout of OC signal as fanout is not allowed in reversible logic. It can be observed that the proposed reversible BCD adder design uses two ancilla input bits, and generates 1 garbage output labelled as g1 in the Fig.4.15 which is the copy of the output carry (OC) that will not be used further in the computation *(the inputs regenerated at the outputs are not considered as garbage outputs)*. The design has the quantum cost of 88 which is the summation of the quantum cost of 4 bit reversible adder with input carry, reversible detection unit, 1 Feynman gate and reversible correction unit. Further, the design has the delay of 73 Δ which

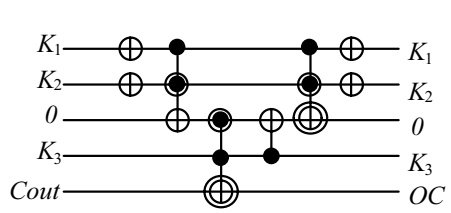
is the summation of the propagation delay of the 4 bit reversible adder with input carry, reversible detection unit, 1 Feynman gate and reversible correction unit.

Once we have designed the 1 digit reversible BCD adder, the n digit reversible BCD adder with input carry can be designed by cascading of the 1 digit reversible BCD adder (RBCD-1) in the ripple carry fashion as illustrated in Fig.4.17(a). Thus, the design 1 of n digit reversible BCD adder with input carry has $2n$ ancilla inputs bits, $2n-1$ garbage outputs, quantum cost of $88n$ and delay of $73n\Delta$. As shown in Fig.4.17(a) the design 1 of the n digit reversible BCD adder with input carry has $2n-1$ garbage outputs because the first 1 digit BCD adder will have 1 garbage output(extra OC output), while the remaining $n-1$ 1 digit BCD adders each will have two garbage outputs. The extra one garbage output in each $n-1$ 1 digit reversible BCD adder is from the ripple carry regenerated at the outputs, for example the second 1 digit BCD adder in Fig.4.17(a) has two garbage outputs labeled as g2 and g3, the extra garbage output g2 is the output carry OC1 of the first 1 digit BCD adder that is passed to the second 1 digit BCD adder as input carry and is regenerated at one of its outputs.

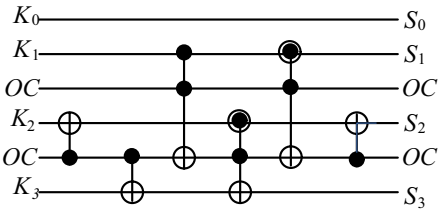
Table 4.9 illustrates that the proposed design is better than the existing design in terms of ancilla input bits and garbage outputs while also being efficient in terms of the quantum cost and the delay.

4.7.2.2 Design 2 of Reversible BCD Adder With No Input Carry

We have also designed 1 digit reversible BCD adder based on approach 1 having no input carry. For achieving the design efficient in terms of number of ancilla input bits and the garbage outputs, we have used the 4 bit reversible input adder without any input carry based on the methodology proposed in this work (the design can be referred in Fig. 4.3). The rest of the design is same as the design of the reversible BCD adder with input carry. The complete design of the 1 digit reversible BCD adder with no input carry is illustrated in Fig.4.16, and will be used with name RBCD-2 further in this work. The design has only 2 ancilla input bits and needs 1 garbage output. The design has the quantum cost of 80 and delay of 80 Δ. Once we have designed the 1 digit reversible BCD adder with no input carry (RBCD-2) , the n digit reversible BCD adder with no input carry can be designed by utilizing the 1 digit reversible BCD adder with no input carry (RBCD-2) to add



(a) Proposed reversible detection unit with one ancilla input bit. K_1, K_2, K_3 and Cout here are the outputs of the top 4 bit binary adder needed in the design of 1 digit BCD adder. OC represents the output carry. The details of K_1, K_2, K_3, Cout and OC signals can be found in Fig.4.13(a)



(b) Proposed reversible correction unit with no ancilla input bit. K_0 here represents the least significant sum bit generated by the 4 bit binary adder in the BCD adder. K_1, K_2, K_3 and OC are the outputs of the reversible detection unit. The details of these signals can also be found in Fig.4.13(a)

Figure 4.14. Proposed detection and correction unit of reversible BCD adder

the least significant digit and then cascading $n-1$ 1 digit reversible BCD adder with input carry (RBCD-1) in the ripple carry fashion. The design of n digit reversible BCD adder with no input carry is illustrated in Fig.4.17(b). Thus, the design 2 of n digit reversible BCD adder without input carry has $2n$ ancilla inputs bits, $2n-1$ garbage outputs, quantum cost of $88n-18$ and delay of $73n-1\Delta$. Table 4.9 illustrates that the proposed design is better than the existing design in terms of number of ancilla input bits and the garbage outputs while also being efficient in terms of the quantum cost and the delay.

4.7.3 Design of Reversible BCD Adder Based on Approach 2

In order to design the reversible BCD adder based on approach 2, we need a 4 bit reversible adder and a 5 bit reversible binary to BCD converter as illustrated in Fig.4.13(b).

4.7.3.1 Design 3 of Reversible BCD Adder With Input Carry

We first present the design of the 1 digit reversible BCD adder with input carry. The 4 bit reversible ripple carry adder with input carry shown in Fig.4.6 is used in the design. Recently, an efficient reversible binary to BCD converter without any ancilla bit is proposed in [4] which we have used in our design. The reversible binary to BCD converter proposed in [4] is illustrated in Fig.4.18 and has the quantum cost of 16 and delay of 16 Δ. The proposed design of the 1 digit reversible

Table 4.9. A comparison of n digit reversible BCD adders

	Ancilla input	Garbage outputs	Quantum cost	Delay Δ
[63]	17n	18n	110n	-
[39]	7n	6n	55n	-
[62]	4n	4n	169n	-
[64]	14n	16n	84n	-
[4] (Design 3*)	2n	6n	103n	-
This work 1 (with input carry)	2n	2n-1	88n	73n
This work 2 (without input carry)	2n	2n-1	88n-18	73n-1
This work 3 (with input carry)	n	n-1	70n	57n
This work 4 (without input carry)	n	n-1	70n-8	57n-3

1 represents proposed Design 1 of Reversible BCD adder with input carry

2 represents proposed Design 2 of Reversible BCD adder without input carry

3 represents proposed Design 3 of Reversible BCD adder without input carry

4 represents proposed Design 4 of Reversible BCD adder without input carry

*In [4], 6 designs of the BCD adders are proposed varying in parameters of the number of ancilla inputs, garbage outputs, quantum cost and the delay. Among 6 designs, design 3 has the minimum number of ancilla inputs and the garbage outputs, thus we have compared to our work with design 3 of the [4].

Table 4.10. Ancilla inputs comparison of n digit reversible BCD adders

Digits	1	2	Proposed*	% Impr. w.r.t 1	% Impr. w.r.t 2
8	32	16	8	75	50
16	64	32	16	75	50
32	128	64	32	75	50
64	256	128	64	75	50
128	512	256	128	75	50
256	1024	512	256	75	50
512	2048	1024	512	75	50

1 is the design in [62]

2 is the design 3 in [4]

* is our design 3 proposed in this work. In improvement calculation all the existing designs are compared with the proposed Design 3 of the proposed reversible BCD adder as among the proposed design it has minimal number of anicilla inputs , garbage outputs, quantum cost and the delay.

Table 4.11. Garbage outputs comparison of n digit reversible BCD adders

Digits	1	2	Proposed*	% Impr. w.r.t 1	% Impr. w.r.t 2
8	32	48	7	78.12	85.4
16	64	96	15	76.56	84.37
32	128	192	31	75.78	83.85
64	256	384	63	75.39	83.59
128	512	768	127	75.19	83.46
256	1024	1536	255	75.09	83.39
512	2048	3072	511	75.04	83.36

1 is the design in [62]

2 is the design 3 in [4]

* is our design 3 proposed in this work. In improvement calculation all the existing designs are compared with the proposed Design 3 of the proposed reversible BCD adder as among the proposed design it has minimal number of anicilla inputs , garbage outputs, quantum cost and the delay.

Table 4.12. Quantum cost comparison of n digit reversible BCD adders

Digits	1	2	Proposed*	% Impr. w.r.t 1	% Impr. w.r.t 2
8	1352	824	560	58.57	34.88
16	2704	1648	1120	58.57	32.03
32	5408	3296	2240	58.57	32.03
64	10816	6592	4480	58.57	32.03
128	21632	13184	8960	58.57	32.03
256	43264	26368	17920	58.57	32.03
512	86528	52736	35840	58.57	32.03

1 is the design in [62]

2 is the design 3 in [4]

* is our design 3 proposed in this work. In improvement calculation all the existing designs are compared with the proposed Design 3 of the proposed reversible BCD adder as among the proposed design it has minimal number of anicilla inputs , garbage outputs, quantum cost and the delay.

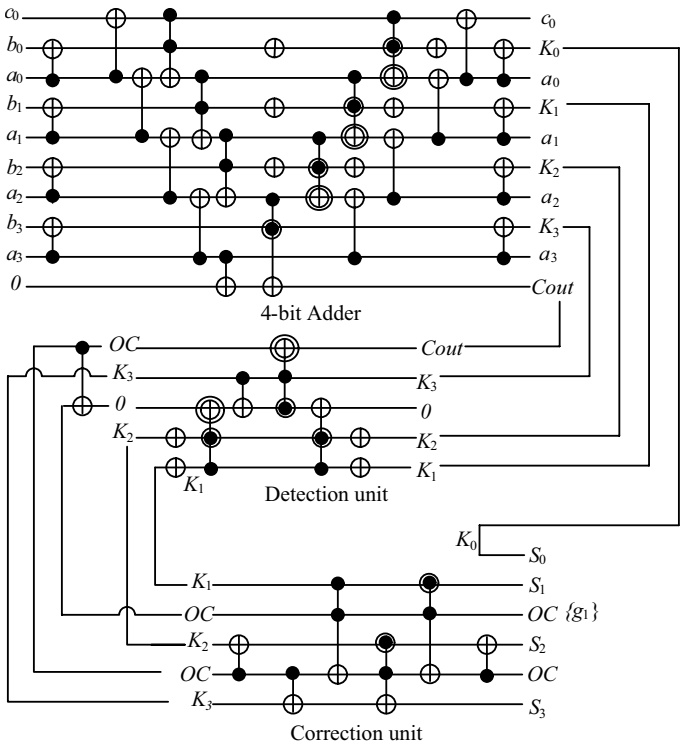


Figure 4.15. Proposed design of 1 digit reversible BCD adder with input carry (RBCD-1) based on approach 1

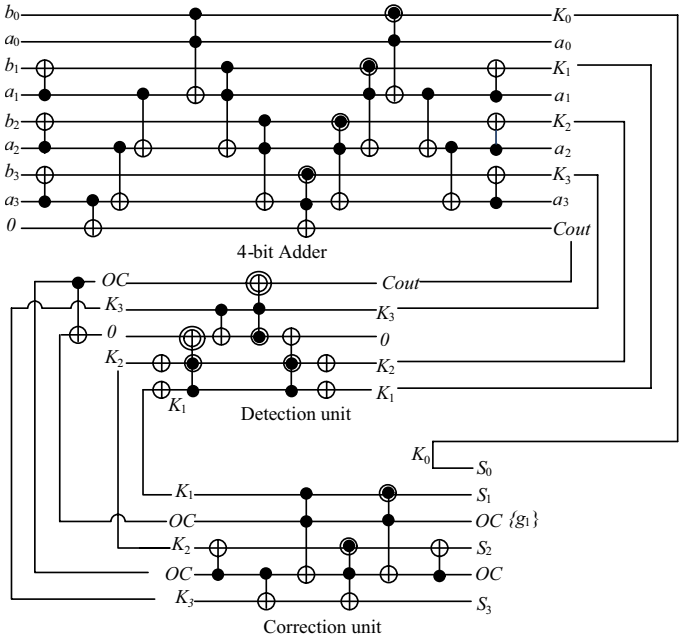
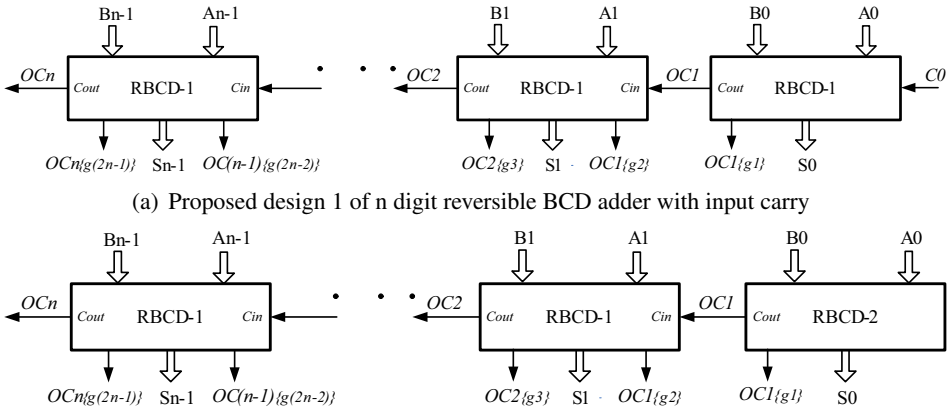


Figure 4.16. Proposed design of 1 digit reversible BCD adder with no input carry (RBCD-2) based on approach 1



(a) Proposed design 1 of n digit reversible BCD adder with input carry

(b) Proposed design 2 of n digit reversible BCD adder with no input carry

Figure 4.17. Proposed designs of n digit reversible BCD adder based on approach 1

BCD adder with input carry (c_0) is shown in Fig.4.19 and will be used with name RBCD-3 in this work. The design has 1 ancilla input bit and zero garbage outputs. The quantum cost of the proposed reversible BCD adder with input carry is 70 while the delay is 57 Δ. *Once we have designed the 1 digit reversible BCD adder (RBCD-3), the n digit reversible BCD adder with input carry can be designed by cascading of the 1 digit reversible BCD adder in the ripple carry fashion as illustrated in Fig.4.21(a).* Thus, the design 3 of the n digit reversible BCD adder with input carry has n ancilla inputs bits, $n-1$ garbage outputs, quantum cost of $70n$ and delay of $57n\Delta$. As shown in Fig.4.21(a) the design 3 of the n digit reversible BCD adder with input carry has $n-1$ garbage outputs as the first 1 digit reversible BCD adder will have no garbage output while the remaining $n-1$ 1 digit reversible BCD adders each will have 1 garbage output. This is because as the output carry of a BCD adder will work as the input carry to the next one and will be regenerated at the outputs. These regenerated input carries at the outputs will not be used further in the computation and hence will form garbage bits. Table 4.9 illustrates that the proposed design is better than the existing design in terms of number of ancilla input bits and garbage outputs while also being efficient in terms of the quantum cost and the delay.

4.7.3.2 Design 4 of Reversible BCD Adder With No Input Carry

We are also proposing another design of the n digit reversible BCD adder that has no input carry. We first design the 1 digit reversible BCD adder with no input carry which is shown in Fig.4.20, and will be used with name RBCD-4 further in this work. The design uses the 4 bit reversible ripple carry adder with no input carry illustrated in Fig.4.3 along with the design of the reversible binary to BCD converter illustrated in Fig.4.18. The proposed design of 1 digit reversible BCD adder has 1 ancilla input bit and has zero garbage outputs. The quantum cost of the design is 62 while the propagation delay is 54 Δ. Once we have designed the 1 digit reversible BCD adder with no input carry, the n digit reversible BCD adder with no input carry can be designed by utilizing the 1 digit reversible BCD adder with no input carry (RBCD-4) to add the least significant digit and then cascading $n-1$ 1 digit reversible BCD adder with input carry (RBCD-3) in the ripple carry fashion. The design of n digit reversible BCD adder with no input carry is illustrated in Fig.4.21(b). Thus,

the design 4 of n digit reversible BCD adder without input carry has n ancilla inputs bits, $n-1$ garbage outputs, quantum cost of $70n-8$ and delay of $57n-3\Delta$. Table 4.9 illustrates that the proposed design is better than the existing design in terms of number of ancilla input bits and the garbage outputs while also being efficient in terms of the quantum cost and the delay.

4.7.4 Comparison of n Digit Reversible BCD Adders

All the existing designs of the reversible BCD adders are with input carry c_0. Among our proposed design of n digit reversible BCD adder with input carry, the design 3 has the minimum number of ancilla inputs bits, garbage outputs, quantum cost and the delay. The results of all the existing works and our proposed work are summarized in Table 4.9. Among the existing works shown in Table 4.9 for the design of n digit reversible BCD adder, the design presented in [62] has the minimum number of garbage outputs, while the design 3 presented in [4] has the minimum number of ancilla inputs. Thus we have shown the comparison of our proposed work with the design presented in [62] and [4] in Tables 4.10, 4.11 and 4.12 in terms of number of ancilla inputs, number of garbage outputs, and the quantum cost, respectively for values of n ranging from n=8 digits to n=512 digits. It is to observed that delays of [62] and design 3 of [4] are not known, thus we are not able to compare our design with these designs in terms of delay. From Table 4.10, it can be observed that in terms of number of ancilla inputs, the proposed design 3 achieves the improvement ratios of 75% and 50% compared to the design presented in [62] and [4], respectively. The improvement ratios in terms of number of garbage outputs range from 78.12% to 75.04%, and 85.4% to 83.36% compared to the design presented in [62] and [4], respectively, the details are illustrated in Table 4.11. As illustrated in Table 4.12, the improvement ratios in terms of quantum cost are 58.57%, and range from 34.88% to 32.03% compared to the design presented in [62] and [4], respectively. Thus the proposed designs of reversible BCD adders are efficient in terms of number of ancilla inputs, garbage outputs, quantum cost and the delay compared to the existing designs in literature.

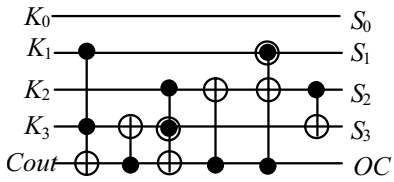


Figure 4.18. Design of binary to BCD converter [4]

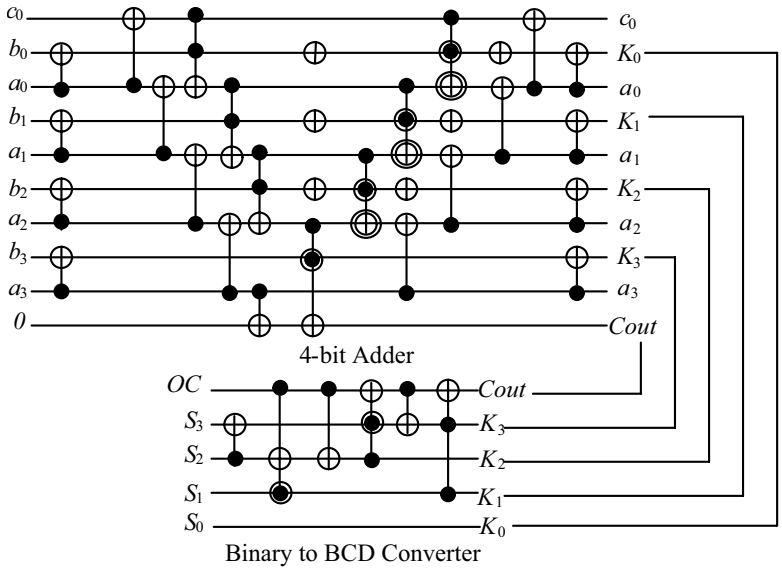


Figure 4.19. Proposed design of reversible BCD adder with input carry (RBCD-3) based on approach 2

4.8 Simulation and Verification

The proposed reversible ripple carry adder designs, reversible subtractors, reversible adder-subtractors, reversible detection unit, reversible correction units, reversible binary to BCD converter and the complete working designs of the reversible BCD adders are functionally verified through simulations. The simulation is performed by creating a library of reversible gates in Verilog Hardware Description Language and is used to code the proposed reversible designs. The Verilog library contains the Verilog codes of reversible gates such as the Fredkin gate, the Toffoli gate, the Peres gate, the TR gate, the Feynman gate etc. All the reversible designs of the adders and the subcompo-

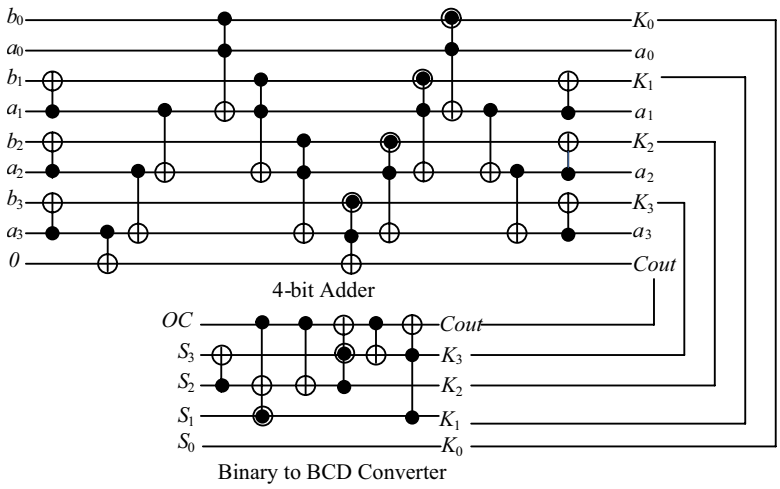
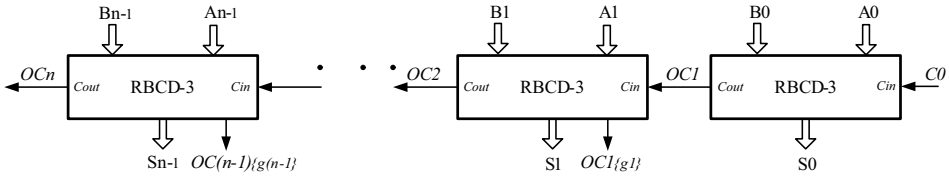


Figure 4.20. Proposed design of reversible BCD adder with no input carry (RBCD-4) based on approach 2

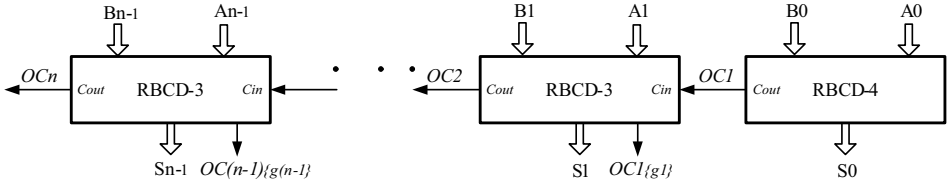
nents are coded in Verilog HDL by utilizing the reversible gates from the Verilog library of reversible gates. The test benches are created for every reversible circuits proposed in this work and for 4 bit, 8 bit and 1 digit reversible circuits exhaustive simulations are done to verify the correctness. The simulation flow used in this work is illustrated in Fig.4.22. The ModelSim and the SynaptiCAD simulators are used for the functional verification of the Verilog HDL codes. The waveforms are generated using the SynaptiCAD Verilog simulator.

4.9 Integration of Proposed Design Methodologies as a Synthesis Framework

The different design approaches of the reversible adder, subtractor, adder-subtractor and BCD adder illustrates that a set of design methodologies corresponding to the various arithmetic and logic units can be developed optimizing the different metrics. In this work, we propose to integrate them as well as the various design methodologies for arithmetic and logic units as a software tool suite. The synthesis portion consists of the algorithms that embed the proposed design methodologies and do not follow the traditional approach due to the variability in the various design methodologies covering a wide range of arithmetic and logic circuits. The proposed synthesis of reversible arithmetic circuits will not be unified and based on a specific synthesis technique, rather will be a collection of



(a) Proposed design 3 of n digit reversible BCD adder with input carry



(b) Proposed design 4 of n digit reversible BCD adder with no input carry (RBCD-4)

Figure 4.21. Proposed designs of n digit reversible BCD adder based on approach 2

algorithms based on the developed design methodologies which is somewhat similar to the approach discussed in [109]. The proposed synthesis framework is discussed below which will evolve and adapt in future based on the needs of the researchers working in the area of reversible arithmetic circuits. The proposed synthesis framework will also consist of design methodologies of reversible arithmetic units proposed by other researchers working in the area of reversible arithmetic circuits.

The framework illustrated in Figure 4.23 will consist of design methodologies for (1) reversible adder circuits; (2) reversible multiplier circuits; (3) reversible modular and exponentiation circuits; (4) reversible barrel shifter and comparator circuits; (5) reversible floating point units; (6) other important reversible arithmetic circuits. *The framework currently supports the synthesis of reversible adders, reversible subtractors, reversible adder-subtractors and reversible BCD adders based on the methodologies proposed in this work and also the synthesis of reversible adder circuits proposed in [56–58].* The proposed framework consists of three main components: synthesis engine, code generation and the simulation engine.

4.9.1 Synthesis Engine

The synthesis engine has the following steps: (1) The designer specifies the design requirement such as the parameters to optimize (number of ancilla inputs, garbage outputs, quantum cost, delay,

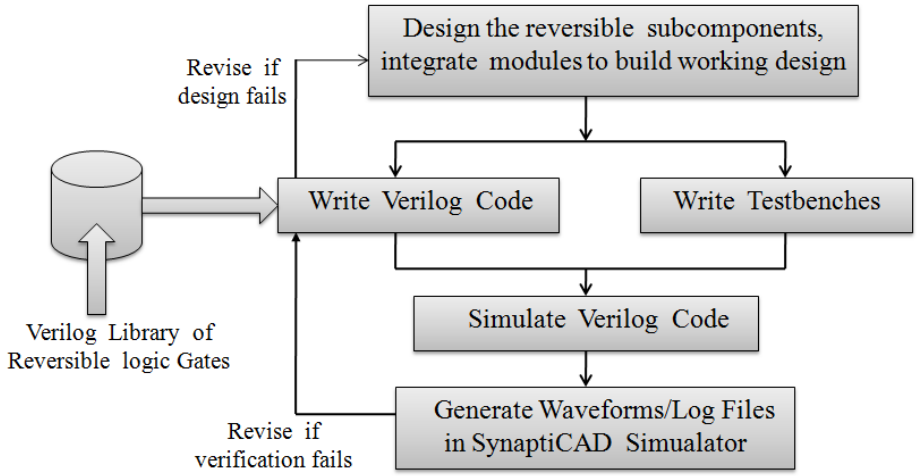


Figure 4.22. Simulation flow of reversible circuits using Verilog HDL

etc) along with the design scheme. For example, the designer can specify the requirement as 64 bit adder based on carry look-ahead scheme; (2) Based on user specification, with a top level python script, the system searches a look-up table to calculate the cost of the design methodology for the particular scheme of the reversible circuit in terms of the number of ancilla inputs, garbage outputs, quantum cost and delay. The look up table has the generalized cost functions for all the designs and design parameters. If the designer requirement is met, the framework proceeds to the next step, or else another scheme or architecture can be explored. For example, if the carry look-ahead scheme does not meet the requirements the user can explore carry skip or another scheme; (3) We have created a library of all the reversible gates coded in Verilog HDL such as the Fredkin gate, the Toffoli gate, the Peres gate, the TR gate, the Feynman gate etc. If a new reversible gate is proposed in the literature, it can be easily added to the library. In the integrated framework, a built-in library of the Verilog codes of the various design methodologies of different reversible arithmetic circuits is created from the Verilog library of reversible gates. The library currently supports the design of reversible binary and BCD adders, reversible subtractors and reversible adder-subtractor. The library would be enhanced in future with Verilog codes of design methodologies of various reversible arithmetic units proposed by the authors as well as the other researchers. Thus, in this

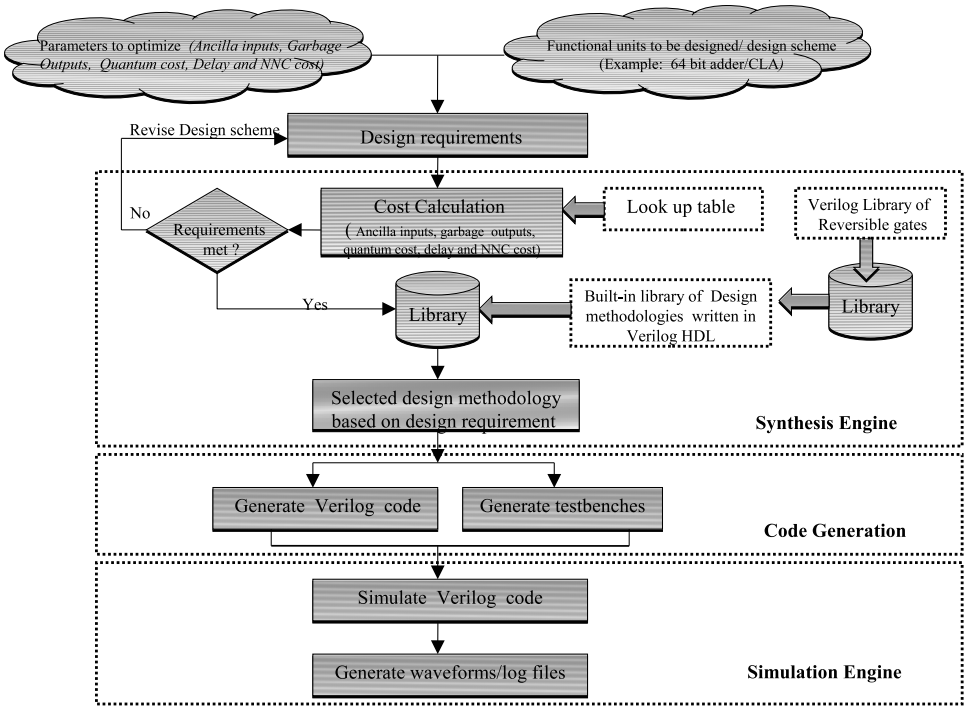


Figure 4.23. Proposed synthesis framework for design of reversible arithmetic and logic circuits

step, depending on the design requirement and the scheme, the tool selects the Verilog code of the design from the built-in library.

4.9.2 Code Generation

In this step, the Verilog code of the desired design is generated. The test benches needed to verify the functional correctness of the design are also generated.

4.9.3 Simulation Engine

In this step, the functional verification of our Verilog HDL codes are done using standard HDL simulators such as ModelSim and SynaptiCAD simulators. The waveforms will be generated along with the log files which will have the various costs of the design in terms of number of ancilla inputs, number of garbage outputs, quantum cost and delay.

4.10 Conclusions

In this work, we have presented efficient designs of reversible ripple carry binary adders, subtractors, adder-subtractors and BCD adders primarily optimizing the parameters of number of ancilla input bits and the garbage outputs. The optimization of the quantum cost and the delay are also considered. The reversible designs of subcomponents used in the BCD adder design such as detection unit, correction unit and the binary to BCD converter are also illustrated. The proposed reversible binary and BCD arithmetic circuits are shown to be better than the existing designs in terms of the number of ancilla inputs bits and the garbage outputs while maintaining the lower quantum cost and the delay. We conclude that the use of the specific reversible gates for a particular combinational function can be very much beneficial in minimizing the number of ancilla input bits, garbage outputs, quantum cost and the delay. All the proposed reversible designs are functionally verified at the logical level by using the Verilog hardware description language and the HDL simulators. The proposed reversible binary adder can also be useful towards the design of reversible floating point multiplier [110].

CHAPTER 5

REVERSIBLE SEQUENTIAL CIRCUITS

In this work, we introduce novel designs of reversible sequential circuits which minimize the quantum cost, delay and the number of garbage outputs, and are more efficient compared to the existing designs [111, 112]. The sequential circuits considered in this work are reversible designs of latches, such as D latch, T latch, JK latch, SR latch and their corresponding master-slave flip-flops. We also introduce negative enable reversible D latch to be used in master-slave flip-flops. Because of the negative enable D latch, we don't require the inversion of the clock for use in the slave latch. Further, this reduces the quantum cost of the master-slave flip-flops. We also introduce a novel strategy of using the Fredkin gate at the outputs of a reversible latch to make it an asynchronous set/reset latch. This strategy is used to design a Fredkin gate based asynchronous set/reset D latch and the master-slave D flip-flop. A reversible design of the universal shift register is also presented. The motivation to design the reversible universal shift register is its wide applications in computing systems, and serves as an example of a complex reversible sequential circuit. We have also illustrated the simulation flow based on Verilog HDL that is used to simulate the reversible sequential circuits.

5.1 Design Methodology

The output equations of the reversible gates are used as the templates for mapping the characteristic equation of the latch into an equivalent reversible design. For example, in the characteristic equation of a latch, suppose we have an expression as A·B$\oplus$C, it can be easily matched with the template of the output equation of the Peres gate, and hence Peres gate can be used to synthesis this

expression. The proposed design methodology is illustrated below with the design of the reversible JK latch as an example circuit.

Step 1: Make templates of the output equations of the basic reversible gates as follows:

- Peres gate: PG(A,B,C)= A·B⊕C
- Fredkin gate: F(A,B,C)= A'B+AC or F(A,B,C)= A'C+AB
- Toffoli gate(4 inputs, 4 outputs): TG4(A,B,C,D)=A·B·C⊕D
- Feynman gate: FG(A,B)=A⊕B
- NOT gate: NOT(A)=A'

Step 2: Derive the characteristic equation of the latch.

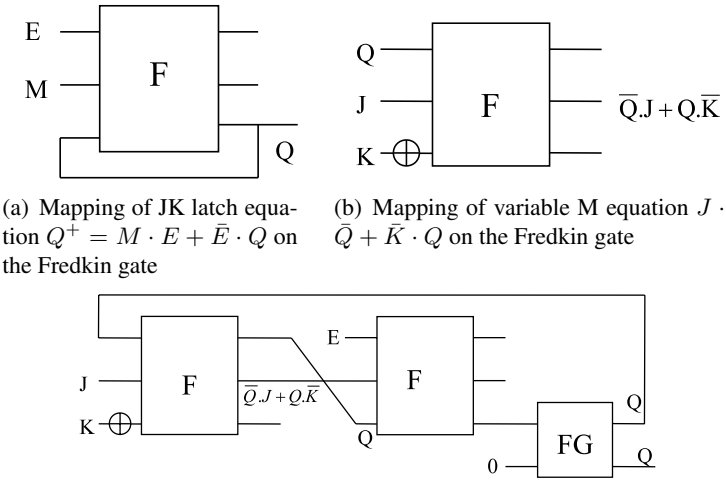
For example, the characteristic equation of the JK latch can be derived as $Q^+ = (J \cdot \bar{Q} + \bar{K} \cdot Q) \cdot E + \bar{E} \cdot Q$ where J and K are the inputs to JK latch, Q is the previous output, E is the enable or clock signal and Q^+ is the current output. Please note that in further discussions E represents the enable or clock signal.

Step 3: Derive the minimum number of garbage outputs needed to convert the characteristic equation as a reversible function.

For example, as evident from the characteristic equation of the JK latch for the eight inputs combinations (E=0,J=0,K=0,Q=0),(E=0,J=0,K=1,Q=0),
(E=0,J=1,K=0,Q=0), (E=0,J=1,K=1,Q=0), (E=1,J=0,K=0,Q=0),
(E=1,J=0,K=1,Q=0), (E=1,J=0,K=1,Q=1) and (E=1,J=1,K=1,Q=1), the output Q^+ is 0. Thus it will require at least 3 garbage outputs to have eight distinct output combinations when Q^+ is 0.

Step 4: Rewrite the characteristic equation by replacing the functions in the parenthesis by variables to have the modified characteristic equation with less variables.

For example, the JK latch characteristic equation can be rewritten as $Q^+ = M \cdot E + \bar{E} \cdot Q$ where M is the new variable substituted for $J \cdot \bar{Q} + \bar{K} \cdot Q$.



(a) Mapping of JK latch equation $Q^+ = M \cdot E + \bar{E} \cdot Q$ on the Fredkin gate

(b) Mapping of variable M equation $J \cdot \bar{Q} + \bar{K} \cdot Q$ on the Fredkin gate

(c) Reversible design of JK latch with minimal garbage outputs

Figure 5.1. An example of the design methodology

Step 5: From the templates in Step 1, find the template which exactly maps the equation in Step 4 with minimum quantum cost.

For example, the equation $Q^+ = M \cdot E + \bar{E} \cdot Q$ maps on the Fredkin gate with minimum quantum cost of 5 where E is equivalent of input A to the Fredkin gate. This step is illustrated in Fig. 5.1(a)

Step 6: Find the template matching for the functions represented by variables in Step 4.

For example, the variable M substituted for $J \cdot \bar{Q} + \bar{K} \cdot Q$ can be mapped on the Fredkin gate with the Fredkin gate inputs as A=Q, B=J and C=K' hence has the quantum cost of 6 (quantum cost of 1 Fredkin gate + quantum cost of 1 NOT gate). This step is illustrated in Fig. 5.1(b)

Step 7: Avoid the fanout in the derived reversible circuit by properly using the Feynman gates. Further maintain the lower bound in terms of garbage outputs by carefully utilizing the outputs.

For example, by avoiding the fanout using Feynman gates and carefully reutilizing the unused outputs, the derived design of the reversible JK latch is shown in Fig. 5.1.

5.2 Design of Reversible Latches

In this section, we present novel designs of reversible latches that are optimized in terms of quantum cost, delay and the number of garbage outputs.

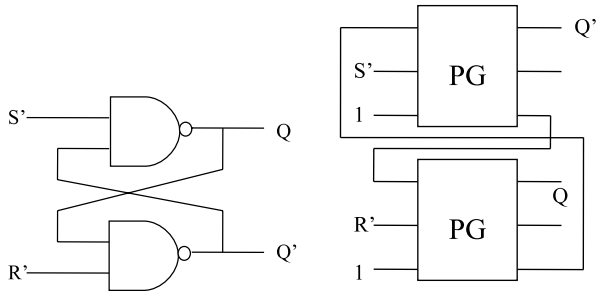
5.2.1 The SR Latch

The SR latch can be designed using two cross-coupled NOR gates or two cross-couple NAND gates. The S input sets the latch to 1 while the R input resets the latch to 0. As an example the NAND gates based design is shown in Fig. 5.2(a). In the existing literature, the cross coupled NOR gate strategy was used to design the Fredkin gate based SR latch and the cross-coupled NAND gate strategy was used for designing the Toffoli gate based SR latch [66, 70]. The Toffoli gate and the Fredkin gate based SR latches require two Toffoli gates and two Fredkin gates, respectively, and hence will have a quantum cost of 10 and delay of 10 Δ. In [69], the Modified Toffoli gate (MTG) based SR latch which requires two MTG gates was proposed. These reversible SR latch designs do not have enable signal (clock) hence are not gated in nature. We can decrease the quantum cost and the delay of the cross-coupled reversible SR latch by using the two Peres gate working as cross-coupled NAND gates as in Fig. 5.2(b). Since the Peres gate has a quantum cost of 4 and the delay of 4 Δ, thus, the proposed design will have a propagation delay of 8 Δ and the quantum cost of 8. To verify the functionality of the design, we coded the Peres gate based cross-coupled reversible SR latch in Verilog HDL using the simulation flow discussed in Section 10. Through simulations we observe that the cross-coupled reversible SR latch does not satisfy the behavior of the SR latch in all possible input cases. This behavior of the cross-coupled reversible SR latch that it oscillates in some input combinations is also pointed by Rice in [68]. Thus, we conclude that the cross-coupled method is not the proper design strategy for designing the reversible SR latch.

In order to design a reversible SR latch that can work for all possible input combinations with minimum number of garbage outputs, we studied the characteristic equation of the SR latch which is given as $Q^+ = S + \bar{R} \cdot Q$. From the characteristic equation, we observe that for five input combinations (S=0,R=0,Q=1),(S=1,R=0,Q=0),(S=1,R=0,Q=1),(S=1,R=1,Q=0) and (S=1,R=1,Q=1) we have Q^+=1, thus it requires at least 3 garbage outputs to have the reversible implementation of the

SR latch. Further, the SR latch enters in the unstable condition when S=1 and R=1. In order to design a reversible SR latch that can work for all input combinations and is optimized in terms of garbage outputs, we modify the output Q for the selective input combinations as follows: When (S=1, R=1,Q=0) the output Q^+ is assigned the value of 0; and when(S=1, R=1,Q=1) the output Q^+ is assigned the value of 1. Thus, when S=1 and R=1, we have Q^+=Q. After the modifications in the output for these two selective input combinations, we observe that now for only four input combinations (S=0,R=0,Q=1),(S=1,R=0,Q=0),(S=1,R=0,Q=1),and (S=1,R=1,Q=1) we have Q^+=1. Thus it will require only two garbage outputs to realize the reversible SR latch. Thus, the garbage outputs in the reversible SR latch will be one less than the number of garbage outputs in the conventional irreversible SR latch. Further, this removes the unstable condition in the SR latch when we have S=1 and R=1. The modified truth table of the SR latch is shown in Table 5.1. From Table 5.1, a new characteristic equation as $Q^+ = (S \oplus Q) \cdot (S \oplus R) \oplus Q$ is derived for the reversible SR latch. This equation can be easily matched with the template of the Peres gate considering A=$S \oplus Q$, B=$S \oplus R$ and C=Q where A,B,C are the inputs of the Peres gate using the design strategy mentioned in Section 5. The functions $S \oplus Q$ and $S \oplus R$ match with the template of the Feynman gate and hence Feynman gates can be used to generate these functions. The proposed design of the reversible SR latch is shown in Fig.5.3(a). The new reversible SR latch has the quantum cost of 7, delay of 7 Δ and has the bare minimum of 2 garbage outputs. The proposed design achieves an improvement of 30% both in terms of the quantum cost as well as the delay, and 50% improvement in terms of garbage outputs compared to the design presented in [70]. The results are summarized in Table 5.2

In order to design the gated reversible SR latch, we derive a new characteristic equation of the gated reversible SR latch as $Q^+ = (E \cdot (S \oplus Q) \cdot (S \oplus R)) \oplus Q$. From the characteristic equation, it can be easily seen that it requires three garbage outputs for the reversible realization of the gated reversible SR latch. Further, using the design strategy considering A= E, B=$(S \oplus Q)$, C=$S \oplus R$ and D=Q where A,B,C,D are the inputs of the 4 input Toffoli gate, the characteristic equation can be easily matched with the template of the Toffoli gate(TG4). The variables B= $(S \oplus Q)$ and C= $S \oplus R$ needed above match with the template of the Feynman gate. The gated reversible design of SR latch is shown in Fig. 5.3(b) and has the quantum cost of 16 (the quantum cost of 4 inputs Toffoli gate



(a) Conventional cross-coupled SR latch design

(b) Peres gate based SR latch design without enable

Figure 5.2. Peres gate based SR latch

Table 5.1. Modified truth table of the SR latch

S	R	Q		Q^+
0	0	0		0
0	0	1		1
0	1	0		0
0	1	1		0
1	0	0		1
1	0	1		1
1	1	0		0
1	1	1		1

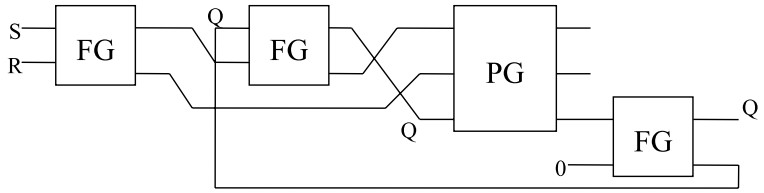
is 13 [44]), and has the delay of 16 Δ. The design has 3 garbage outputs which is minimal. The comparison with the existing work is summarized in Table 5.3

5.2.2 The D Latch

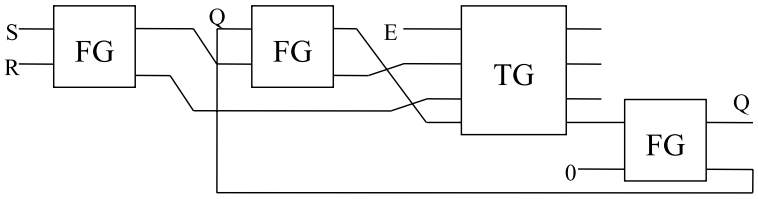
The characteristic equation of the D latch can be written as $Q^+ = D \cdot E + \bar{E} \cdot Q$. When the enable signal(clock) is 1, the value of the input D is reflected at the output that is Q^+=D.

Table 5.2. A comparison of reversible SR latches

	Quantum Cost	Delay	Garbage Outputs
Fredkin gate based [70]	10	10	2
Toffoli Gate based [70]	10	10	2
Proposed Design	7	7	2
Improvement in % w.r.t [70]	30	30	-



(a) New design of reversible SR latch



(b) New design of gated reversible SR latch

Figure 5.3. Reversible SR latch based on modified truth table

Table 5.3. A comparison of gated reversible SR latches

	Quantum Cost	Delay	Garbage Outputs
[67]	34	29	6
Proposed Design	16	16	3
Improvement in % w.r.t [67]	52.9	44.8	50

While, when E=0 the latch maintains its previous state, that is Q^+=Q. The characteristic equation of the D latch can be mapped to the Fredkin gate (F) as it matches the template of the Fredkin gate. Figure 5.4(a) shows the realization of the reversible D latch using the Fredkin gate and the Feynman gate (fanout is not allowed in reversible logic and the role of Feynman gate is to avoid the fanout) [69, 74, 113]. The quantum cost of the design presented in [69, 74, 113] is 6 and is realized with two garbage outputs. We observed that it cannot be further optimized in terms of quantum cost and the garbage outputs. This can be understood as follows: From the characteristic equation of the D latch $Q^+ = D \cdot E + \bar{E} \cdot Q$, we can see that for the four inputs combinations (E=0,D=0,Q=0),(E=0,D=1,Q=0), (E=1,D=0,Q=0), and (E=1,D=0,Q=1), the output Q^+ is 0. The addition of one garbage output can resolve only two output positions since one bit can produce only two distinct output combinations. Since $2^2 = 4 > 3$, thus it will require at least 2 garbage outputs to have four distinct output combinations when Q^+ is 0. *We are also showing the propagation delay of the Fredkin gate based D latch*. Since in this design we have 1 Fredkin gate and 1 Feynman gate in

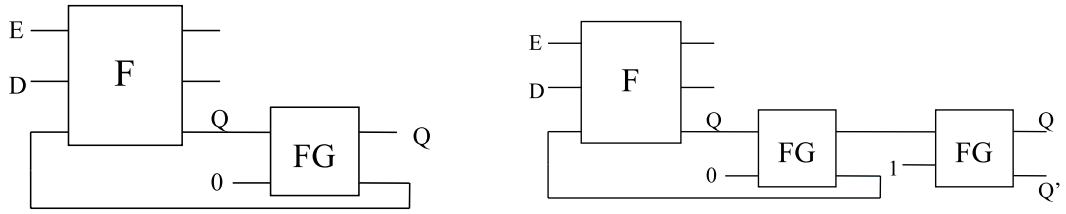
Table 5.4. A comparison of reversible D latches

	Quantum Cost	Delay	Garbage Outputs
[67]	47	25	6
[69]	10	10	2
Proposed Design	7	7	2
Improvement in % w.r.t [67]	85	72	67
Improvement in % w.r.t [69]	30	30	-

series, its propagation delay is 6 Δ which is the summation of 5 Δ propagation delay of the Fredkin gate and 1 Δ propagation delay of the Feynman gate.

5.2.2.1 The D Latch With Outputs Q and Q'

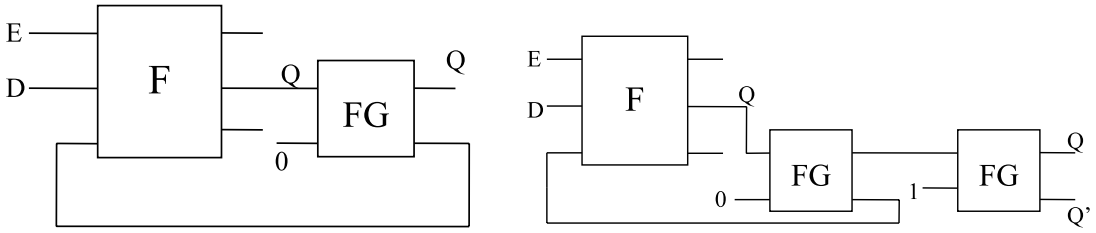
The design shown in Fig. 5.4(a) does not produce the complement output Q' which is required often in sequential circuits [70]. Among the existing designs, only the design presented in [67, 69] have both the outputs Q and its complement Q'. The design in [67] requires 4 New gates and 3 Feynman gates, and thus has the quantum cost of 47. The propagation delay of the D latch design in [67] is 25 Δ from input D to output Q', and has 6 garbage outputs. In [69] two Fredkin gates are used to design the D latch having the outputs Q and Q'. Since each Fredkin gate has the quantum cost of 5, thus the design presented in [69] has the quantum cost of 10 and needs two garbage outputs. Further, the design in [69] has two Fredkin gates connected in series, thus its propagation delay is 10 Δ. In this work, we propose a novel design of the D latch that has both the outputs Q and Q', and is designed with the 1 Fredkin gate and the 2 Feynman gates as shown in Fig 5.4(b). In the design, the Feynman gate is used to generate the complement of the output Q as illustrated in Fig.2.2(d). The proposed design has the quantum cost of 7 (quantum cost of the 1 Fredkin gate+ quantum cost of the 2 Feynman gates) and has bare minimum of two garbage outputs. The propagation delay of this design is 7 Δ. A comparison of the proposed design with the existing designs is shown in Table 5.4 which shows that the proposed design achieves improvement ratios of 85%, 72% and 67 % in terms of quantum cost, delay and the garbage outputs compared to the design presented in [67]. The improvement ratios compared to [69] are 30% both in terms of quantum cost as well as delay, while maintaining the minimum 2 garbage outputs.



(a) Fredkin gate based D latch

(b) Reversible D latch with outputs Q and Q'

Figure 5.4. Designs of reversible D latch



(a) Fredkin gate based negative enable reversible D latch with only output Q

(b) Fredkin gate based negative enable reversible D latch with outputs Q and Q'

Figure 5.5. Fredkin gate based negative enable reversible D latch

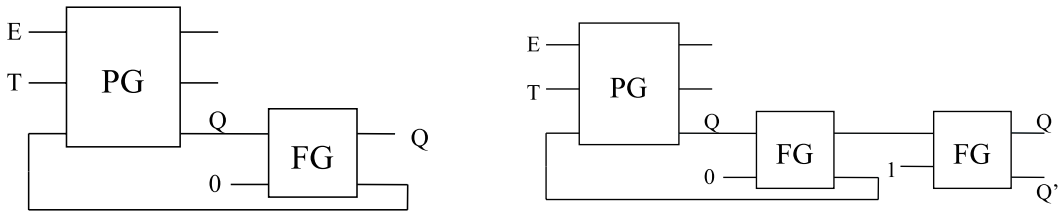
5.2.2.2 The Negative Enable Reversible D Latch

In this work, we introduce a design of the reversible D latch that will pass the input D to the output Q when E=0; otherwise maintains the same state. The characteristic equation of such a negative enable D latch can be written as $Q^+ = D \cdot \bar{E} + E \cdot Q$. This characteristic equation of the negative enable reversible D latch matches with the template of the Fredkin gate. Thus, it can be mapped on the 2nd output of the Fredkin gate as shown in Fig.5.5(a). The Feynman gate used in the design plays the role of avoiding the fanout of more than one. In this work, the negative enable D latch is designed with a special purpose of utilizing it in master-slave flip-flops. This is because as it will help to design master-slave flip-flops in which no clock inversion is required. The details of which are discussed in Section 7. The design shown in Fig.5.5(a) does not have the output Q' which can be generated as shown in Fig.5.5(b).

5.2.3 The T Latch

The characteristic equation of the T latch can be written as $Q^+ = (T{\cdot}Q){\cdot}E{+}\bar{E}{\cdot}Q$. But the same result can also be obtained from $Q^+ = (T \cdot E) \oplus Q$. The T(toggle) latch is a complementing latch which complements its value when T=1, that is when T=1 and E=1 we have Q^+ =Q'. When T=0, the T latch maintains its state and we have no change in the output. The T latch characteristic equation can be directly mapped to the Peres gate and the fanout at output Q can be avoided by cascading the Feynman gate (we can see that $T \cdot E \oplus Q$ matches with the template of the Peres gate). The proposed design is shown in Fig 5.6(a). The design has the quantum cost of 5 (quantum cost of 1 Peres gate+ 1 Feynman gate), delay of 5 Δ and requires 2 garbage outputs. The proposed reversible T latch design has the minimum garbage outputs as from the characteristic equation of the T latch $Q^+ = (T{\cdot}E){\oplus}Q$, we can see that for the four inputs combinations (E=0,T=0,Q=0),(E=0,T=1,Q=0), (E=1,D=0,Q=0), and (E=1,T=1,Q=1), the output Q^+ is 0. Thus it will require at least 2 garbage outputs to have four distinct output combinations when Q^+ is 0.

The existing design in literature [74] has the quantum cost of 6, delay of 6 Δ and also requires 2 garbage outputs. Thus, the proposed design is better than the design presented in [74], and achieves the improvement ratios of 17 % both in terms of quantum cost and delay as shown in Table 5.5. But these designs do not produce the complementary output Q'. In the existing literature the reversible T latch design having both Q and the complementary output Q' is presented in [67, 69]. In this work, we are proposing a novel design of the T latch based on the Peres and the Feynman gates that has both the outputs Q and Q'. The proposed design is shown in Fig 5.6(b) and has the quantum cost of 6, delay of 6 Δ and produces 2 garbage outputs. The design uses the Feynman gate to generate the complement of the output Q as illustrated in Fig.2.2(d). The proposed reversible T latch with outputs Q and Q' achieves an significant improvement compared to the designed presented in [67] in terms of quantum cost, delay and the garbage outputs, respectively. The comparison is summarized in Table 5.6.



(a) Peres gate based T latch (b) Reversible T latch with outputs Q and Q'

Figure 5.6. Designs of reversible T latch

Table 5.5. A comparison of reversible T latches having only output Q

	Quantum Cost	Delay	Garbage Outputs
[74]	6	6	2
Proposed Design	5	5	2
Improvement in %	17	17	-

Table 5.6. A comparison of reversible T latches having outputs Q and Q'

	Quantum Cost	Delay	Garbage Outputs
[67]	46	35	12
[69]	10	10	2
Proposed Design	6	6	2
Improvement in % w.r.t [67]	87	83	83
Improvement in % w.r.t [69]	40	40	-

Table 5.7. A comparison of reversible JK latches having only output Q

	Quantum Cost	Delay	Garbage Outputs
[74]	32	32	3
Proposed Design	12	12	3
Improvement in %	62.5	62.5	-

5.2.4 The JK Latch

The design of the reversible JK latch is already discussed in Section 5 and is shown in Fig.5.1. The JK latch have the capability to set the output, reset the output or complement the output depending on the value of J and K. When E(clock) is 1, the J input can set the output to 1 (when J=1), the reset input can reset the output to 0 (when K=1), and when both J and K are set to 1 the output Q is complemented. The proposed reversible JK latch design has the quantum cost of 12, delay of 12 Δ and produces 3 garbage outputs. The existing design in literature in [74] is designed with 2 4x4 Toffoli gates, 1 3x3 Toffoli gate and 1 Feynman gate. The quantum cost of 4x4 Toffoli gate is 13 [114]. Since quantum cost is also the measure of logical depth, the delay of 4x4 Toffoli gate is 13 Δ. Thus the JK latch proposed in [74] has the quantum cost of 32, delay of 32 Δ and needs 3 garbage outputs. Thus the design of JK Latch proposed in this work achieves an improvement of 62.5 % both in terms of quantum cost as well as delay, while maintaining the 3 garbage outputs. The result is summarized in Table 5.7.

The above discussed designs of JK latch do not produce the complemented output Q'. Thus we are illustrating another design of the JK latch in Fig. 5.7 which produces both the output Q and its complement Q'. The design has the quantum cost of 13, delay of 13 Δ and produces 3 garbage outputs. In this design also, the Feynman gate is used to generate the complement of the output Q. The proposed design achieves a significant improvement compared to the existing design in literature [67] and [69]. The comparison is summarized in Table 5.8.

5.3 Design of The Reversible Master-Slave Flip-Flops

The reversible master-slave flip-flops were first presented in [67] in which the authors had used the strategy of using one latch as a master and the other latch as a slave to design the reversible

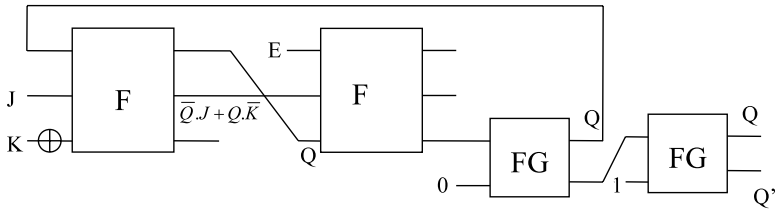


Figure 5.7. Reversible JK latch with outputs Q and Q'

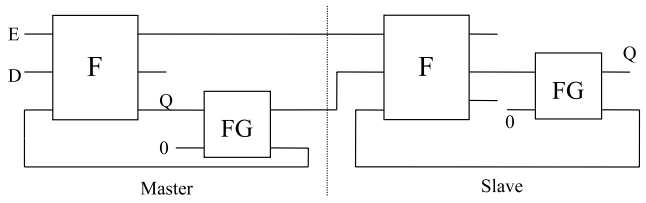
Table 5.8. A comparison of reversible JK latches having outputs Q and Q'

	Quantum Cost	Delay	Garbage Outputs
[67]	46	35	12
[69]	16	16	3
Proposed Design	13	13	3
Improvement in % w.r.t [67]	72	63	75
Improvement in % w.r.t [69]	19	19	-

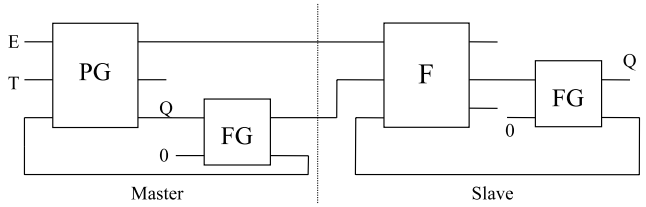
flip-flops. The same strategy was followed in [68, 69, 74]. The proposed work is also based on the same strategy and the goal is to optimize the quantum cost and the delay of the flip-flops along with the garbage outputs. All the existing reversible master-slave flip-flop designs require the clock to be inverted for use in the slave latch. *The proposed master-slave flip-flops designs have a special characteristic that they don't require the clock to be inverted for use in the slave latch. This is because as they use the negative enable D latch as the slave latch thus no clock inversion is required.*

Figure 5.8(a) shows the design of the master-slave D flip-flop in which we have used positive enable Fredkin gate based D latch shown in Fig. 5.4(a) as the master latch, while the slave latch is designed from the negative enable Fredkin gate based D latch shown earlier in Fig. 5.5(a). Figures 5.8(b), 5.8(c) and 5.8(d) show the designs of master-slave T flip-flop, JK flip-flop and SR flip-flop, respectively. *It is to be noted that in the proposed master-slave flip-flop designs, the master is designed using the positive enable corresponding latch, while the slave is designed using the negative enable Fredkin gate based D latch. For example, in the master-slave JK flip-flop, the master is designed using the positive enable JK latch, while the slave is designed with the negative enable D latch. The use of negative enable D latch makes sure that we do not have to invert the clock which saves the NOT gate generally used in the designs for inversion of clock. This decreases*

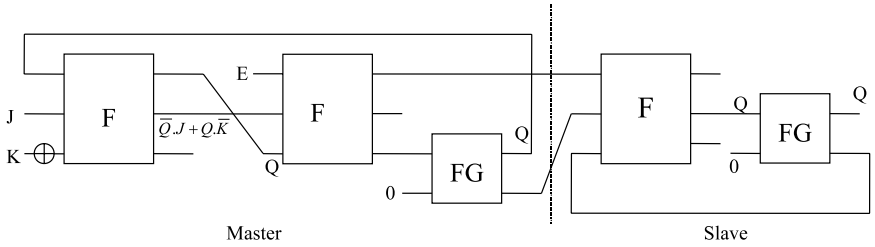
the quantum cost and the propagation delay of the design. Now comparing with the existing designs in literature, [67] has only shown the design of master-slave JK flip-flop. In [68], [74] and [69], the designs of master-slave D flip-flop, JK flip-flop and T flip-flop are shown. Thus, we can conclude that *the proposed work is also the first work in literature to show the design of master-slave SR flip-flop and has the quantum cost of 22, delay of 22 Δ and has 4 garbage outputs*. A comparison of the proposed designs of master-slave flip-flops with the existing designs in literature is shown in Tables 5.9, 5.10 and 5.11. From Table 5.9, it can be concluded that the proposed master-slave D flip-flop achieves the improvement ratios of 74%, 65.7% and 75% in terms of quantum cost, delay and the garbage outputs, compared to the design presented in [68]. The improvement compared to the design presented in [74] is 7.6% both in terms of quantum cost as well as delay, while maintaining the same number of garbage outputs. The improvement compared to the design presented in [69] is 7.6% both in terms of quantum cost as well as delay, and 25% in terms of number of garbage outputs. Regarding the design of the reversible master-slave T flip-flop, it can be concluded from Table 5.10 that the proposed design achieves the improvement ratios of 77.5% , 75% and 78.5% in terms of quantum cost, delay and the garbage outputs compared to the design presented in [68]. The improvement compared to the design presented in [74] is 15% both in terms of the quantum cost as well as the delay, while maintaining the same number of garbage outputs. The improvements compared to the design presented in [69] is 35% both in terms of quantum cost as well as delay, and 25% in terms of number of garbage outputs. The proposed reversible master-slave JK flip-flop also achieves the improvements of 72%, 59% and 71.4 % in terms of quantum cost, delay and the number of garbage outputs, respectively, compared to the design presented in [68]. The improvement compared to the design presented in [74] is 55% both in terms of quantum cost and delay while maintaining the minimum number of garbage outputs. Further, the proposed reversible master-slave JK flip-flop design achieves the improvements of 21.7%, 18% and 25% in terms of quantum cost, delay and the number of garbage outputs compared to the design presented in [69].



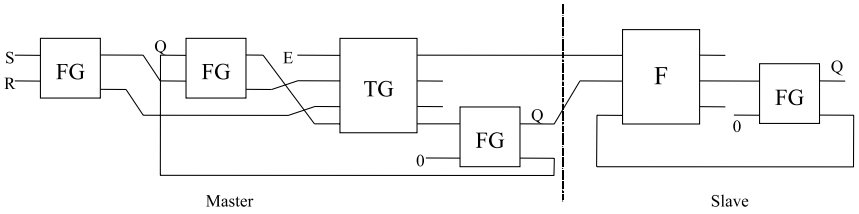
(a) Reversible master-slave D flip-flop



(b) Reversible master-slave T flip-flop



(c) Reversible master-slave JK flip-flop



(d) Reversible master-slave SR flip-flop

Figure 5.8. Design of reversible master-slave flip-flops

Table 5.9. A comparison of reversible master-slave D flip-flops

	Quantum Cost	Delay	Garbage Outputs
[68]	47	35	12
[74]	13	13	3
[69]	13	13	4
Proposed Design	12	12	3
Improvement in % w.r.t [68]	74	65.7	75
Improvement in % w.r.t [74]	7.6	7.6	-
Improvement in % w.r.t [69]	7.6	7.6	25

Table 5.10. A comparison of reversible master-slave T flip-flops

	Quantum Cost	Delay	Garbage Outputs
[68]	65	44	14
[74]	83	13	3
[69]	17	17	4
Proposed Design	11	11	3
Improvement in % w.r.t [68]	77.5	75	78.5
Improvement in % w.r.t [74]	15	15	-
Improvement in % w.r.t [69]	35	35	25

Table 5.11. A comparison of reversible master-slave JK flip-flops

	Quantum Cost	Delay	Garbage Outputs
[68]	64	44	14
[74]	39	39	4
[69]	23	22	5
Proposed Design	18	18	4
Improvement in % w.r.t [68]	72	59	71.4
Improvement in % w.r.t [74]	55	55	-
Improvement in % w.r.t [69]	21.7	18	25

5.4 Design of The Reversible Latch and The Master-Slave Flip-Flop With Asynchronous Set and Reset Capability

In this work, we propose a novel design of the reversible D latch that can be asynchronously set and reset according to the control inputs. Before discussing the design of the asynchronously set/reset D latch, the properties of the Fredkin gate that will be helpful in understanding the design are discussed below.

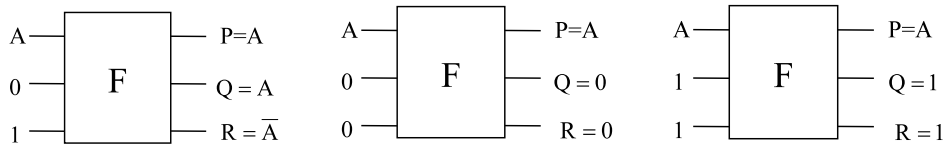
- As shown in Fig.5.9(a), the Fredkin gate can be used to avoid the fanout of a signal by assigning that signal to its input A. The other two inputs B and C of the Fredkin gate are assigned the values as B=0 and C=1. This will result in copying of the input A of the Fredkin gate at the outputs P and Q, thus avoiding the fanout problem.

- As shown in Fig. 5.9(b), by assigning the value 0 to the inputs B and C of the Fredkin gate we can reset the outputs Q and R to 0. This is a very useful property to asynchronously reset the Q and R outputs of the Fredkin gate.

- As shown in Fig. 5.9(c), by assigning the value 1 to the inputs B and C of the Fredkin gate we can set the outputs Q and R to 1. This is a very useful property to asynchronously set the outputs Q and R of the Fredkin gate.

The design of the asynchronously set/reset D latch is shown in Figure 5.10(a). The design has 2 Fredkin gates and 1 Feynman gate. We can observe that the first Fredkin gate maps the D latch characteristic equation, while the second Fredkin gate help in asynchronous set/reset of the output Q. The fanout is avoided by use of the Feynman gate. The design has two control inputs C1 and C2. When C1=0 and C2=1, the design works in normal mode implementing the D latch characteristic equation. When C1=0 and C2=0, the second Fredkin gates will reset the output Q to 0. When C1=1 and C2=1, the design will be set to Q=1. Thus, the control inputs help the design to work in various modes. The design has the quantum cost of 11, delay of 11 Δ and has 4 garbage outputs.

The reversible design of the master-slave D flip-flop with asynchronous set/reset is shown in Fig. 5.10(b). The design contains positive enable D latch as the master latch and negative enable

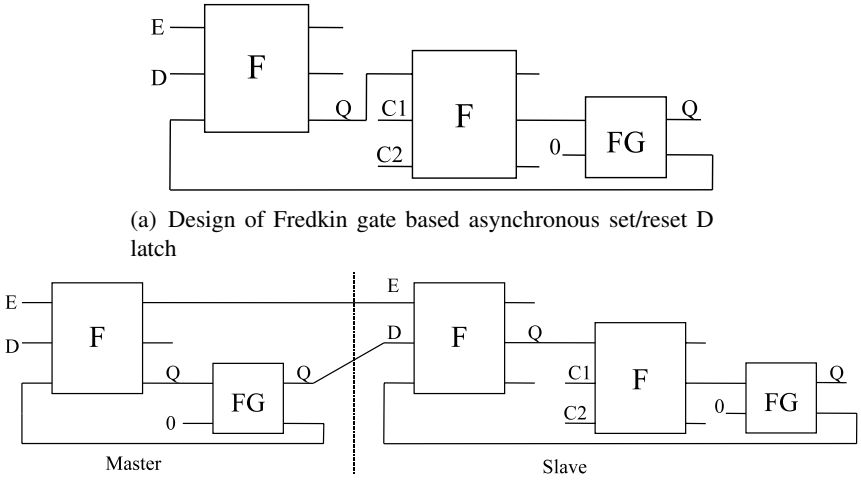


(a) Application of the Fredkin gate to avoid the fanout

(b) Asynchronous reset of the Q and R outputs of the Fredkin gate

(c) Asynchronous set of the Q and R outputs of the Fredkin gate

Figure 5.9. Working of the Fredkin gate in various modes



(a) Design of Fredkin gate based asynchronous set/reset D latch

(b) Reversible asynchronous set/reset master-slave D flip-flop

Figure 5.10. Design of asynchronous set/reset reversible sequential circuits

asynchronous set/reset D latch as the slave latch. The design has the quantum cost of 17, delay of 17 Δ and has 5 garbage outputs. *The designs of the reversible T latch, the reversible JK latch, the reversible SR latch can also be conditioned similarly as asynchronous set/reset designs. For example, asynchronous set/reset design of T latch can be designed by replacing the Feynman gate in Fig.5.6(a) with 1 Fredkin gate and 1 Feynman gate. Fredkin gate will have control signals C1 and C2 while the Feynman gate will avoid the fanout.*

5.5 Design of Reversible Universal Shift Register

In this section, we present a reversible universal shift register to demonstrate the application of the proposed asynchronous set/reset D flip-flop in designing complex reversible sequential circuits. Universal shift register is a register that has both shifts and parallel load capabilities [115].

The reversible universal shift register needs to have the following functionalities and components:

- Reversible D flip-flops with capability of asynchronously reset.
- Clock input for synchronization of operations
- Shift-right control signal and shift-left control signal to enable shift right operation and shift left operation, respectively.
- A parallel-load control to enable a parallel transfer through the n input lines. Parallel output lines to capture the data.
- A control unit designed from reversible 4:1 multiplexers that enable various operations such as shift right or shift left, and is also helpful in leaving the register values unchanged if required. The 4:1 multiplexers are controlled by two select signals S1 and S0. When S1S0=00, input 0 is selected; S1S0=01 enables input 1 to be selected, and so on for the other inputs.

Figure 5.12 shows the design of a 4 bit reversible universal shift register as an example circuit. The shift register consists of 4 reversible D flip-flops with asynchronous reset capability, and four reversible 4:1 multiplexers (R-4x1 MUX) that work as a control unit. The design of a 4 bit reversible multiplexer is shown in Fig. 5.11. It is designed from 3 Fredkin gates and has 5 garbage outputs. The quantum cost of the design is 15. In the design shown in Fig.5.11, the first three 4:1 MUXes have 3 garbage outputs each. This is because as the garbage outputs having the S0 and S1 signals can be used as inputs to the next 4:1 MUX. But the 4th 4:1 MUX have 5 garbage outputs since the garbage outputs having the S0 and S1 signals cannot be used further in computation. Thus in the design of the 4 bit reversible universal shift register 4 4:1 MUXes will have 3x3+5=14 garbage outputs, and their quantum cost is 4x15=60.

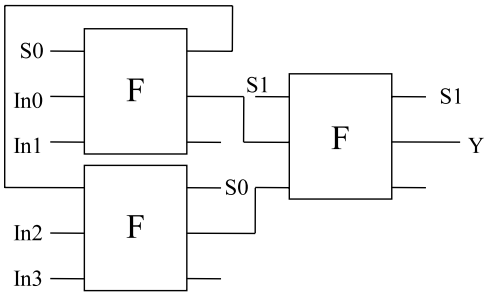


Figure 5.11. Design of a reversible 4:1 MUX

Another component in the design of a reversible universal shift register is reversible D flip-flops with asynchronous reset capability, the design of which is shown in Fig. 5.10(b). We can see from Fig.5.12 that we need C1 and C2 signals to be passed as inputs to the next D flip-flops. In each D flip-flop, the copies of the C1 and C2 signals can be generated by using the 2 Feynman gates, one for C1 and one for C2. We don't need Feynman gates at the 4th D flip-flop as C1 and C2 are not required further in computation. Thus the quantum cost of 4 D flip flops is $3 \times$ ($2 \times$ quantum cost of Feynman gate + quantum cost of D flip flop) + quantum cost of D flip-flop = $3 \times (2 \times 1 + 36) + 36 = 150$. We can see further from Fig. 5.12 that we need 10 Feynman gates in the design to avoid the fanout. For example, the Feynman gates are needed at the outputs A0, A1 and so on. Thus, the total quantum cost of the 4 bit universal shift register will be 10+60+150=220. The working of the reversible universal shift register can be described as follows:

When S1S0=00, the outputs of each reversible D flip-flops are fed back as inputs to them. Thus during the next edge of the clock, the same value is maintained at the outputs of the reversible flip flops and no change occurs in their states. When S1S0=01, this enables input 1 of the multiplexers to be passed as input to the reversible D flip-flops. As can be seen from the Fig. 5.12, S1S0=01 enables the shift-right operation with the serial input getting transferred to reversible flip-flop A3. When S1S0=10, the 2nd inputs of the multiplexers are enabled resulting in a shift-left operation with the serial input getting transferred to the reversible flip-flop A0. When we have S1S0=11, the 3rd inputs of the multiplexers enabled resulting in transfer of information from parallel input lines to the reversible D flip flops, simultaneously, in the next clock edge.

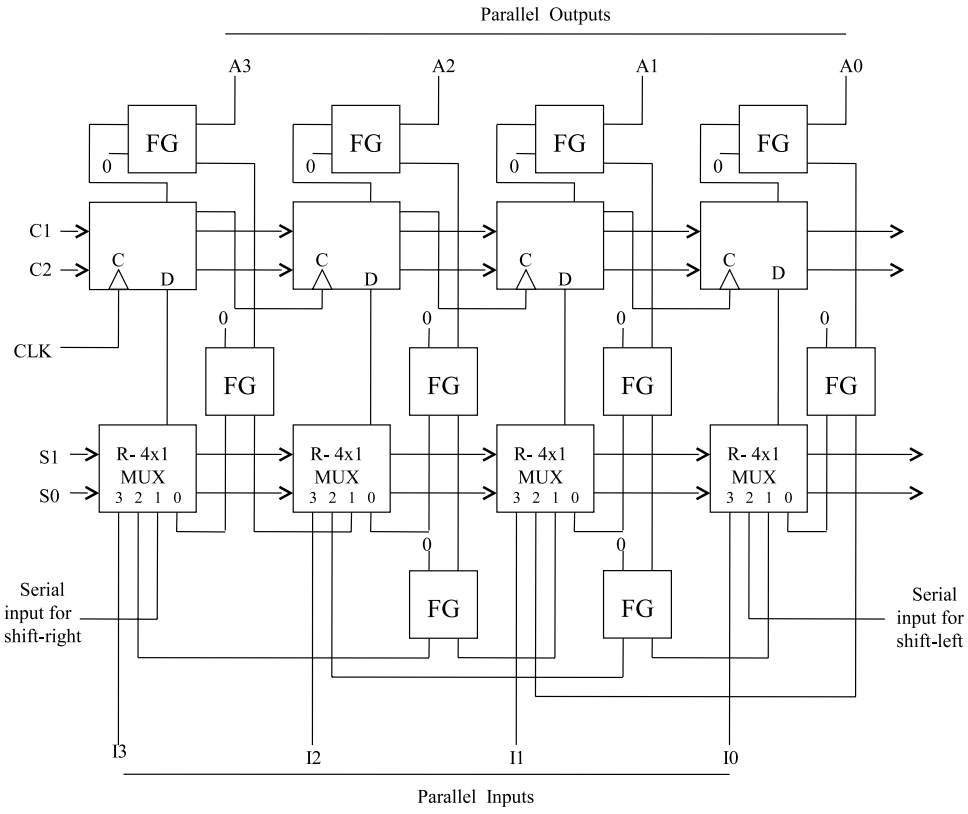


Figure 5.12. Design of a reversible universal shift register

5.6 Simulation and Verification

The proposed designs were functionally verified through simulations coding their designs in the Verilog Hardware Description Language. In order to achieve this, we built a library of reversible gates in Verilog HDL and use it to code the proposed designs of sequential circuits. The library contains the Verilog codes of reversible gates such as the Fredkin gate, the Toffoli gate, the Peres gate, etc. We have created test benches for every reversible sequential circuits proposed in this work to verify their correctness. The functional verification of our Verilog HDL codes is done using the ModelSim and SynaptiCAD simulators. The waveforms shown in this work are generated using the SynaptiCAD Verilog simulator. The simulation flow used in this work is shown in Fig.5.13.

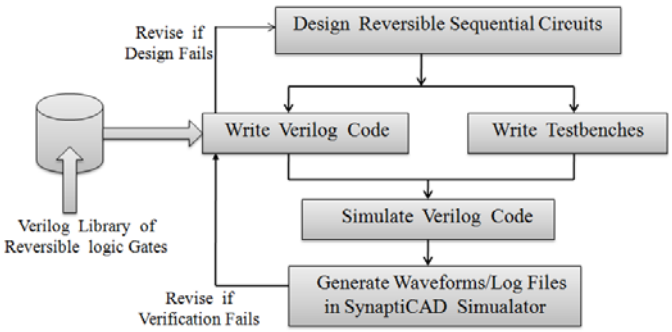


Figure 5.13. Verilog HDL based simulation flow for reversible sequential circuits

5.6.1 Simulation of The Proposed Reversible Latches

Figure 5.14(a) shows the simulation of the D latch. In our Verilog simulations, we have added an artificial signal CLR equivalent of clear signal to initialize the output Q to the value 0. This will help the simulation to run correctly since the value of the output Q won't be unknown while initializing the simulation. Arrows A and B in the Fig. 5.14(a) show that when CLK=1, the value of D becomes the output Q, that is we have Q^+ = D. When CLK=0, the latch maintains its previous state, that is we have Q^+ = Q. Figure 5.14(b) shows the simulation of the D latch with outputs Q and its complement Q'(QB) and can be explained similar to Fig.5.14(b). *While simulating the latches we have functionally verified both types of latches having output Q as well as the latches having output Q and Q'. But, since the simulation results of the latches having the output Q is similar to the simulation results of the latches having outputs Q and Q', we will only show the simulation results of the latches with output Q.*

Figure 5.15(a) shows the simulation of the T latch. Arrow A, B and D in the figure show that when CLK=1 and T=1, the output Q toggles and we will have next state Q^+ = Q' (complement of the previous state). Arrow C represents the case when CLK=1 and T=0, the output will maintain its previous state and we will have Q^+ = Q. We have also functionally verified the working of the T latch having outputs Q and Q'. Figure 5.15(b) shows the simulation of the JK latch. Arrow A shows that when CLK=1, J=0 and K=1 we will have the output Q=0. Arrow B shows that when CLK=1, J=1 and K=1 the output toggles and we will have next state Q^+ = Q'(complement of the previous

state). Arrow C shows that when CLK=1, J=1 and K=0 we will have the output Q=1. It can be observed from Fig. 5.15(b) that when CLK=0 the JK flip flop maintains the previous state that is Q^+ = Q. This verifies the functionality of the reversible JK latch. Figure 5.17 shows the simulation of the SR latch without enable signal (no clock). Arrow A shows that when S=0 and R=1 we will have the output Q=0, arrow B shows that when S=1 and R=0 we will have the output Q=1. Arrow C and D show that when S=1 and R=1, the output Q maintains its previous state that is we will have Q^+ = Q. Figure 5.16(b) shows the simulation of the gated SR latch. Its functional verification is same as the SR latch when the value of CLK is 1, otherwise when the value of the CLK is 0 it maintains its previous state.

5.6.2 Simulation of The Proposed Reversible Master-Slave Flip-Flops

Figure 5.17(a) shows the simulation of the proposed reversible master-slave negative-edge triggered D flip-flop. Arrows A and C show that when CLK=1, the master latch latches the value D=1 which is reflected by the slave latch as the output Q=1 when CLK=0. Arrow B shows that when CLK=1, the master latches the value D=0 which is reflected by the slave latch as the output Q=0 when CLK=0. This verifies the working of the master-slave D flip-flop. Figures 5.17(b),5.17(c) and 5.17(d) show the simulation of the master-slave T flip-flop, JK flip-flop and SR-flip flop, respectively. The functional correctness of these designs can easily be observed from the waveforms.

5.6.3 Simulation of The Proposed Reversible Asynchronous Set/Reset D Latch and Master-Slave Flip-Flop

Figure 5.18(a) shows the simulation of the proposed reversible asynchronous set/reset D Latch. In Fig. 5.18(a) arrows A, B, D and E show that when control signals C1=0 and C2=1, the latch works in normal mode and whenever we have CLK=1 the value of the D is reflected at the output. Arrow C shows that when C1=1 and C2=1, we have the output Q asynchronously set to 1 irrespective of the value of the clock signal (For example as shown in arrow C when C1=1 and C2=1, the output Q becomes 1 even when the CLK has the value as 0). Arrow E shows that when C1=0 and C2=0, the output Q asynchronously resets to 0 irrespective of the value of the clock signal.

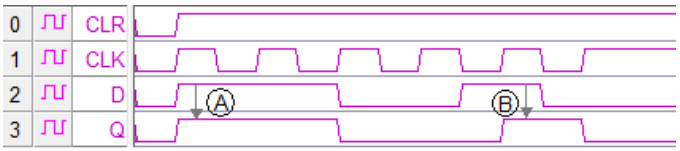
Figure 5.18(a) shows the simulation of the proposed reversible asynchronous set/reset D flip-flop in which master is designed from positive enable D latch while the slave is designed from asynchronous set/reset D latch as shown in Fig. 5.10(b). In Fig.5.18 arrows C and E show that the flip-flop can be asynchronously set/reset depending on the value of control signals C1 and C2. Arrows A, B and D represent the case when the design works as a negative-edge triggered master-slave D flip flop as controls are C1=0 and C2=1.

5.6.4 Simulation of The Proposed Reversible Shift Register

Figure 5.19 shows the simulation of the proposed 4-bit reversible universal shift register. Its working in shift-right and shift-left modes is shown in Fig.5.19(a). Arrow A shows that when S1=1 and S0=0, the designs works in left shift mode and the value of SISL (Serial Input for Shift-Left) is transferred to output A0. The value at output A0 is left shifted by one place in every clock cycle. Arrow B shows that when S1=0 and S0=1, the design works in right shift mode and the value of the SISR (Serial Input for Shift-Right) is transferred to output A3. The value at output A3 is right-shifted by one place in every clock cycle. Figure 5.19(b) shows the parallel and the no change mode. In Fig. 5.19(b) arrows A, B, C, D show the parallel load mode when S1=1 and S0=0 in which the value at inputs I0,I1,I2 and I2 are transferred to outputs A0, A1,A2 and A3, respectively. Arrow E shows that when S1=0 and S0=0 there will be no change and the shift register will maintain its previous state. Thus the waveforms matches with the functionality of the universal shift register.

5.7 Conclusions

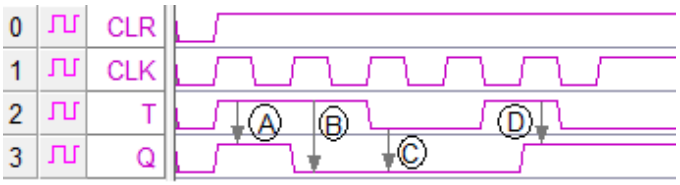
In this work we have presented novel designs of reversible latches and flip- flops which are being optimized in terms of quantum cost, delay and the garbage outputs. The present work differs from the existing approaches in literature which have optimized the reversible sequential circuit designs in terms of number of reversible gates and garbage outputs. We have also discussed the new designs of reversible D latch and D flip-flop with asynchronous set/reset capability. The reversible design of universal shift register is also presented as an example of designing complex reversible sequential circuits. The reversible designs are functionally verified at the logical level by using



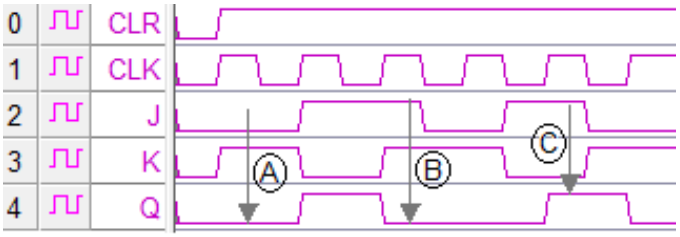
(a) Simulation of D latch with output Q

(b) Simulation of D latch with outputs Q and Q'

Figure 5.14. Simulation of D latch

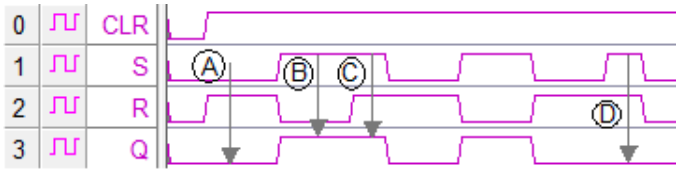


(a) Simulation of T latch

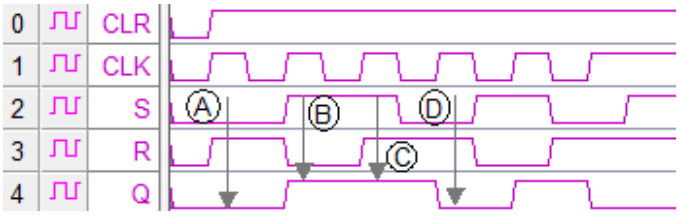


(b) Simulation of JK latch

Figure 5.15. Simulation of T and JK latch



(a) Simulation of SR latch without enable

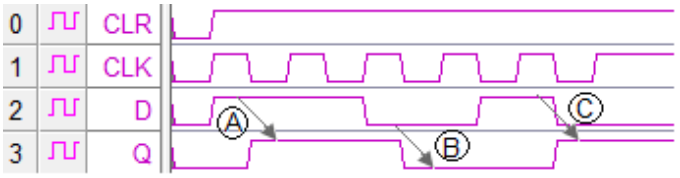


(b) Simulation of gated SR latch

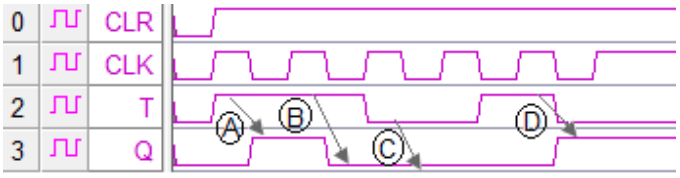
Figure 5.16. Simulation of SR latch

Verilog hardware description language. The simulation flow used for verifying is also presented. The proposed reversible sequential designs are shown better than the existing designs in literature in terms of quantum cost, delay and the garbage outputs.

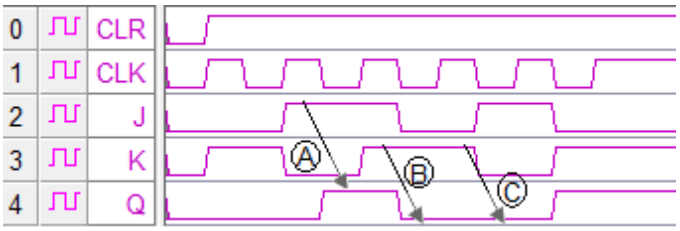
One of the limitations of our work is that it does not calculate the exact minimal quantum gate realization (lower bound) of the reversible sequential circuits. The works in [50,51] are fundamental contributions in the direction of deriving exact minimal elementary quantum gate realization of reversible combinational circuits. Their main goal is to optimize the number of quantum gates, however, they refer to work in [44] for garbage minimization and do not address the issue of delay. Further, although these works do not address sequential circuits, it may be possible to modify and adopt the proposed methods to design reversible sequential circuits. The next future step in the design of reversible sequential circuits is to investigate synthesis of reversible sequential circuits with possibly minimizing or (with minimal) all three metrics of quantum gates, delay and garbage outputs. The proposed designs will form the basis of efficient reversible sequential circuits to be used in nano-computing regime.



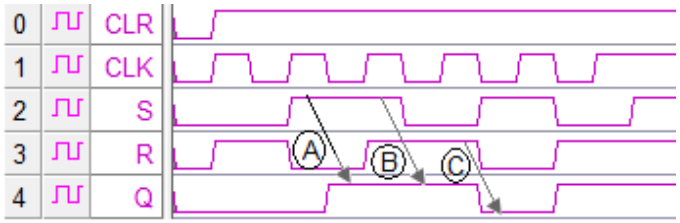
(a) Simulation of master-slave D flip-flop



(b) Simulation of master-slave T flip-flop

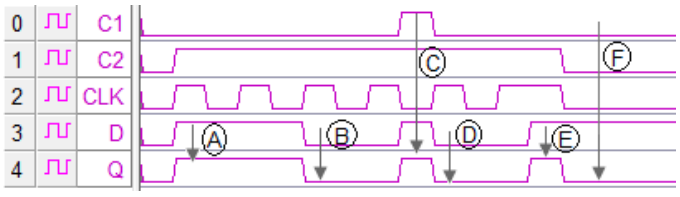


(c) Simulation of master-slave JK flip-flop

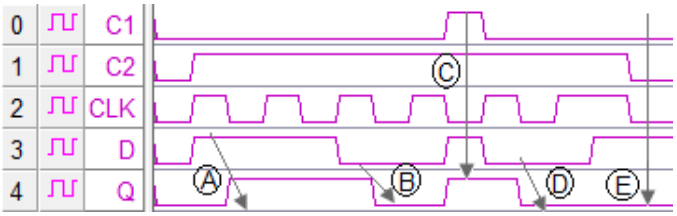


(d) Simulation of master-slave SR flip-flop

Figure 5.17. Simulation of master-slave flip-flops

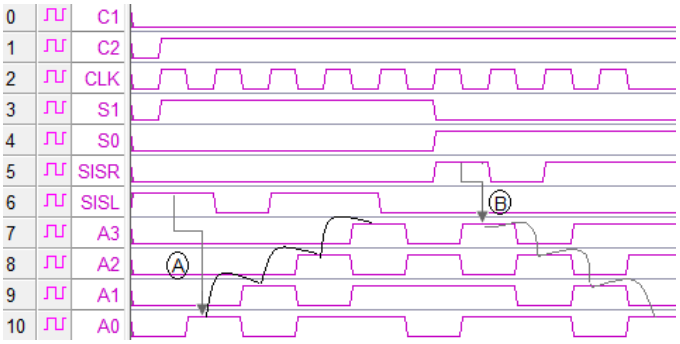


(a) Simulation of asynchronous set/reset D latch

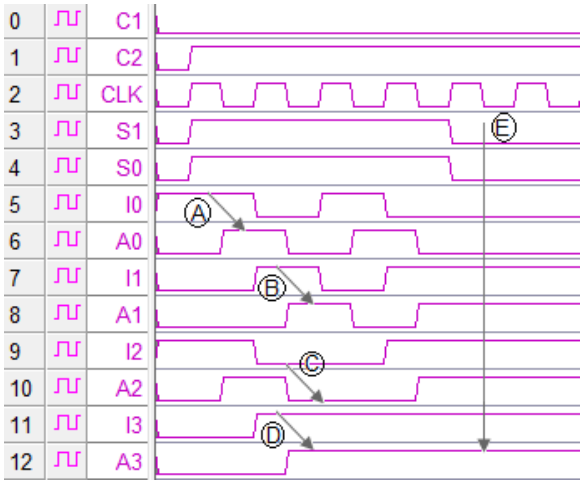


(b) Simulation of asynchronous set/reset D flip-flop

Figure 5.18. Simulation of asynchronous set/reset D Latch and flip-flop



(a) Simulation of shift register in left/right mode



(b) Simulation of shift register in parallel load mode

Figure 5.19. Simulation of 4 bit universal shift register

CHAPTER 6

CONCURRENTLY TESTABLE REVERSIBLE CIRCUITS

As discussed in chapter 2, Quantum Dot Cellular Automata (QCA) is one of the emerging nanotechnologies that exhibit a small feature size, high clock frequency and ultra low power consumption. Due to the significant error rates in nano-scale manufacturing, there is a critical need to maintain extremely low device error rates in nanotechnologies including the QCA. In the manufacturing of QCA, permanent defects can occur in the synthesis and deposition phases. However, defects are more likely to take place during the deposition phase. QCA devices are also prone to transient faults caused by thermodynamic effects, radiation and other effects, as the energy difference between the ground and the excited state is small. To the best of our knowledge, the concurrent testing of faults in QCA and QCA-based sequential circuits has not been addressed in the literature. Concurrent error detection (CED) is defined as the property of circuits in which faults/errors can be detected at run time while the circuit is performing the normal operations. In this work, we propose a class of novel designs for the implementation of concurrently testable circuits for molecular QCA based on a special type of reversible logic called conservative reversible logic [116–118]. In conservative reversible logic, in addition to one- to-one mapping, there would be an equal number of 1s in the outputs as there would be on the inputs. Further, this work also presents novel conservative logic gates for QCA computing called the 'MV-cqca'(Majority voter conservative QCA gate). The conservative logic gates are used to design concurrently testable combinational and sequential circuits. Further, the logic block and the routing fabric (both are programmable) are the two key components of an FPGA. Thus, we have shown the reversible and Fredkin gate based concurrently testable designs of the configurable logic block (CLB) and the routing switch of a molecular QCA-based FPGA. Analysis of power dissipation in the proposed FPGA is also shown [119]. The design

and verification of the QCA layouts were performed using the QCADesigner and HDLQ tools. We also proposed a inverse and compare scheme for concurrent detection of multiple faults in reversible logic circuits [120].

6.1 Conservative Reversible Fredkin Gate

A conservative logic gate is a multiple-output logic element in which number of ls in the inputs is equal to the number of 1s in the outputs.Thus, the conservative logic gate can be reversible if the one-to-one mapping is maintained between the inputs and the outputs. If one-to-one mapping is not preserved the conservative logic gate will not be reversible in nature. Fredkin gate is the most popular reversible and conservative logic gate. Fredkin gate can be described as mapping (A, B, C) to (P=A, Q=A'B+AC, R=AB+A'C), where A, B, C are inputs and P, Q, R are outputs, respectively, and is shown in Fig. 6.1. Table 6.1 shows the truth table of Fredkin Gate and it can be seen that Fredkin gate produces the same number of 1s in the outputs as on the inputs, in addition to the one-to-one mapping feature of reversibility. Moreover, it is parity preserving: its input parity is equal to the output parity. The proposed QCA layout of the Fredkin gate is shown in Figure 6.3. The basics of the QCA computing such as the basic QCA devices, information flow is already discussed in the Chapter 2.

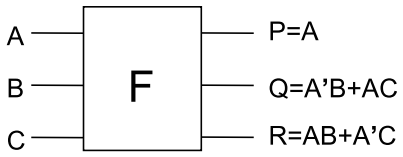


Figure 6.1. Fredkin gate

6.2 Fault Modeling of QCA Implementation of Fredkin Gate

In the proposed work, the QCA layout of the Fredkin gate is converted into the corresponding hardware description language notations using the HDLQ Verilog library [121]. The HDLQ design tool consists of a Verilog HDL library of QCA devices, i.e., MV, INV, fan-out, Crosswire, L-shape wire with fault injection capability. The HDLQ model of the QCA layout of the Fredkin gate is

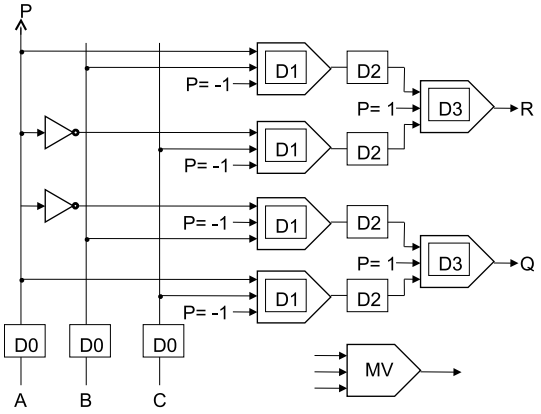


Figure 6.2. QCA design of Fredkin gate using the four-phase clocking scheme, in which the clocking zone is shown by the number next to D (D0 means clock 0 zone, D1 means clock 1 zone and so on)

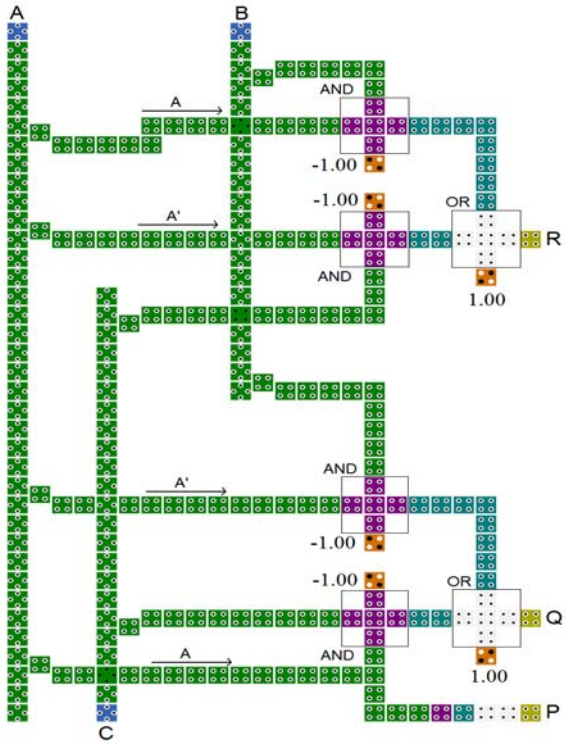


Figure 6.3. QCA layout of Fredkin gate

Table 6.1. Truth table for Fredkin gate

A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	1	0
0	1	1		0	1	1
1	0	0		1	0	0
1	0	1		1	1	0
1	1	0		1	0	1
1	1	1		1	1	1

shown in Fig.6.4. In Fig.6.4, FO represents the fanout QCA device, LS represents the L-shape wire, INV represents the QCA inverter, CW represents the crosswire, MJ represents the majority voter. Thus it can be seen that modeled QCA layout of the Fredkin gate has 4 FOs, 2 INVs, 5 CWs, 9 LSs and 6 MJs. The HDLQ modeled design of the Fredkin gate is simulated for the presence of all possible single missing/additional cell defect in MJs (majority voters), INVs (Inverters), FOs (fanouts), Crosswires (CWs) and L-shape wires (LSs). The design is simulated using the Verilog HDL simulator in the presence of faults to determine the corresponding outputs.

We conducted exhaustive testing of the HDLQ model of the Fredkin gate with 8 input patterns in the presence of all possible single missing/additional cell defect. Testing of the Fredkin gate generated 28 unique fault patterns at the output, as shown in Tables 6.2, 6.3 and 6.4. Due to limitation of page width, the fault pattern table is divided into Tables 6.2, 6.3 and 6.4, where Table 6.2 illustrates the 10 fault patterns, Table 6.3 illustrates the next 10 fault patterns, and Table 6.4 represents the last 8 fault patterns. In the fault patterns study shown in the Tables, ai is the 3 bit pattern with an equivalent decimal value of i. For example, a0 represents 000 (decimal 0) and a7 represents 111 (decimal 7). The exhaustive fault pattern tables show that Fredkin gate can be very much effective in the concurrent testing (run time detection of fault while the circuit is performing normal operation) of single missing/additional cell defect in QCA computing as any single missing/additional cell defect can be easily detected by counting the number of 1s at the inputs and the outputs. For example, as can be seen in fault pattern one in Table 6.2 for the input vector a2 (A=0,B=1,C=0) we have output as a3 (P=0,Q=1,R=1), for input vector a6 (A=1,B=1,C=0) we have output as a4 (P=1,Q=0,R=0), for

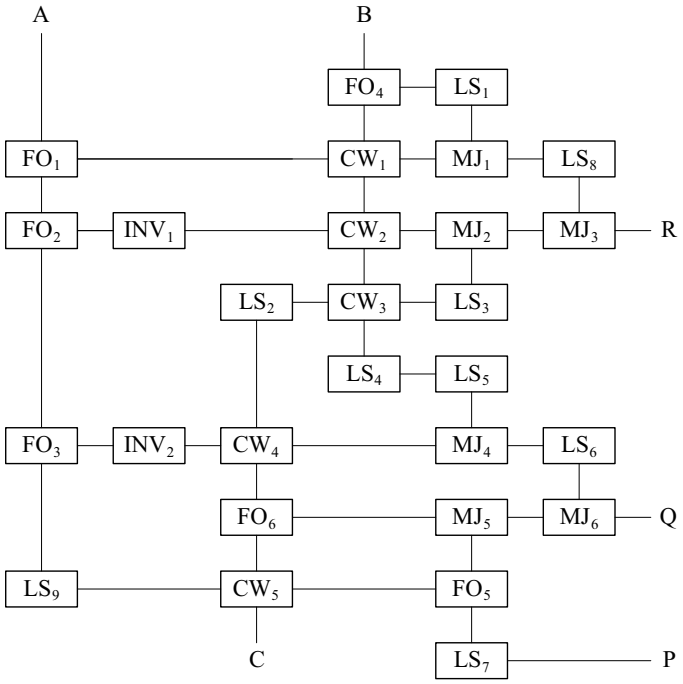


Figure 6.4. Modeling of QCA layout of Fredkin gate, where FO represents the fanout QCA device, LS represents the L-shape wire, INV represents the QCA inverter, CW represents the crosswire and MJ represents the majority voter

input vector a7 (A=1,B=1,C=1) we have output as a6 (P=1,Q=1,R=0). Thus we can see that when there is a single missing/additional cell defect, the number of 1s in its output set will differ from the number of 1s in its input set, or the output set is correct. Thus the fault can be easily detected by counting the number of 1s at the inputs and counting the number of 1s at the outputs, and comparing them. By property of conserving logic, when there is no fault the number of 1s at the inputs is same as the number of 1s at the outputs. Since Fredkin gate preserves the number of 1s at the outputs to same as the number of 1s at the inputs, it can also detect the transient faults that result in difference in number of 1s at the inputs and the outputs. Hence, Fredkin gate can concurrently detect permanent, as well as, transient faults in molecular QCA.

6.3 Concurrently Testable Latches for Molecular QCA

In this section, we present the design of concurrently testable latches based on concurrently testable Fredkin gate.

Table 6.2. Fault patterns in Fredkin gate (1-10)

Input Vector	Fault Free	Fault Patterns									
		1	2	3	4	5	6	7	8	9	10
a0	a0	a0	a0	a1	a1	a0	a0	a1	a2	a1	a1
a1	a1	a1	a1	a1	a1	a0	a1	a0	a1	a1	a1
a2	a2	a3	a2	a2	a3	a2	a0	a3	a2	a3	a3
a3	a3	a3	a3	a3	a3	a2	a1	a2	a1	a3	a3
a4	a4	a4	a5	a5	a5	a4	a4	a4	a4	a5	a4
a5	a6	a6	a7	a7	a7	a7	a6	a6	a6	a6	a6
a6	a5	a4	a4	a5	a4	a5	a5	a5	a5	a5	a5
a7	a7	a6	a6	a7	a6	a7	a7	a7	a7	a7	a7

Table 6.3. Fault patterns in Fredkin gate (11-20)

Input Vector	Fault Free	Fault Patterns									
		11	12	13	14	15	16	17	18	19	20
a0	a0	a4	a2	a0	a2	a0	a0	a1	a0	a2	a4
a1	a1	a7	a3	a1	a3	a1	a3	a0	a3	a3	a5
a2	a2	a6	a0	a0	a2	a2	a2	a3	a2	a0	a6
a3	a3	a7	a1	a1	a3	a3	a3	a2	a3	a1	a7
a4	a4	a0	a4	a4	a4	a5	a4	a6	a4	a6	a0
a5	a6	a0	a6	a6	a6	a7	a4	a4	a6	a6	a2
a6	a5	a1	a5	a7	a5	a5	a5	a7	a5	a7	a1
a7	a7	a1	a7	a7	a7	a7	a5	a5	a7	a7	a3

6.3.1 D Latch

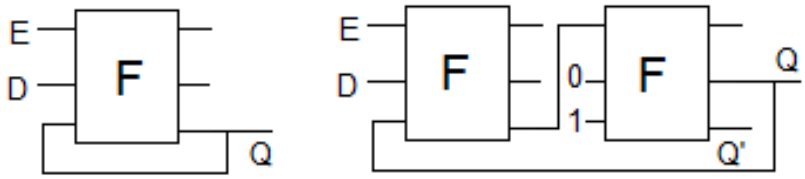
The characteristic equation of the D latch can be written as $Q^+ = D \cdot E + \bar{E} \cdot Q$. The equation can be mapped onto the Fredkin gate (F). Figure 6.5(a) shows the design of the concurrently testable D latch using Fredkin gate. It is to be noted that fan-out is not allowed in reversible logic, but allowed in molecular QCA. Therefore, the design shown in Fig. 6.5(a) is valid for molecular QCA. However, it does not produce the Q' (the complement of Q), which may be required in a number of places while designing the sequential circuits. Thus, we are also showing another design of D latch in Fig.6.5(b), a design that also produces the complement output.

6.3.2 T Latch

The characteristic equation of the T latch can be written as $Q^+ = (T \cdot Q) \cdot E + \bar{E} \cdot Q$. However, the same result can also be obtained from $Q^+ = (T \cdot E) \oplus Q$. Figure 6.3.2 shows the proposed

Table 6.4. Fault patterns in Fredkin gate (21-28)

Input Vector	Fault Free	Fault Patterns							
		21	22	23	24	25	26	27	28
a0	a0	a4	a0	a0	a0	a0	a0	a0	a2
a1	a1	a1	a1	a1	a1	a0	a1	a1	a3
a2	a2	a6	a2	a2	a2	a2	a2	a2	a2
a3	a3	a3	a3	a1	a2	a2	a3	a3	a3
a4	a4	a4	a6	a4	a4	a4	a4	a6	a4
a5	a6	a0	a4	a6	a7	a6	a6	a6	a6
a6	a5	a5	a7	a7	a5	a5	a7	a7	a5
a7	a7	a1	a5	a7	a7	a7	a7	a7	a7



(a) Fredkin D latch (b) Fredkin D latch with complement output

Figure 6.5. Concurrently testable D latch design using Fredkin gates

design of concurrently testable T latch in which the 1st Fredkin gate produces $(T \cdot E)$. The 2nd and 3rd Fredkin gate generates $(T \cdot E) \oplus Q$ (2nd and 3rd Fredkin gate combinedly generates the XOR function).

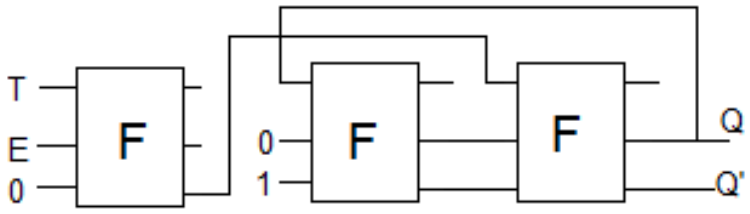


Figure 6.6. Concurrently testable T latch

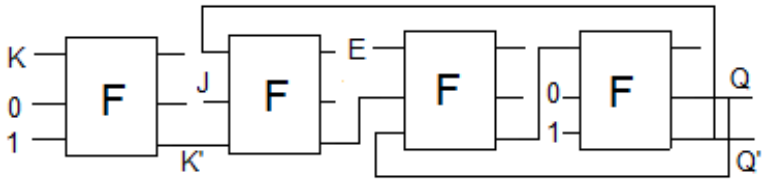


Figure 6.7. Concurrently testable JK latch

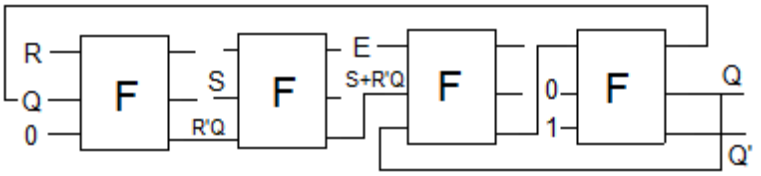


Figure 6.8. Concurrently testable SR latch design

6.3.3 JK Latch

The characteristic equation of the JK latch can be written as $Q^+ = (J \cdot \bar{Q} + \bar{K} \cdot Q) \cdot E + \bar{E} \cdot Q$. After computing the equation $(J \cdot \bar{Q} + \bar{K} \cdot Q)$, it can be mapped on the D Latch to design the JK Latch. Figure 6.3.3 shows the proposed design of concurrently testable JK latch. The 1st Fredkin produces K', which is passed to the 2nd Fredkin to generate $(J \cdot \bar{Q} + \bar{K} \cdot Q)$. The output $(J \cdot \bar{Q} + \bar{K} \cdot Q)$ produced by the 2nd Fredkin gate is passed to the 3rd and 4th Fredkin gate working as a D latch.

6.3.4 SR Latch

The characteristic equation of the SR latch can be written as $Q^+ = (S + \bar{R} \cdot Q) \cdot E + \bar{E} \cdot Q$. After computing the equation $(S + \bar{R} \cdot Q)$, it can be mapped on the D Latch to design the SR Latch. Figure 6.3.4 shows the proposed design of the concurrently testable SR latch. The 1st Fredkin gate produces $\bar{R} \cdot Q$ which is passed to 2nd Fredkin gate to produce $(S + \bar{R} \cdot Q)$. The $(S + \bar{R} \cdot Q)$ is passed to 3rd and 4th Fredkin gate working as a D latch to finally generate the SR latch.

6.4 Simulations for Functional Verification

All the designs were verified using QCADesigner ver. 2.0.3 [90]. In the bistable approximation, we used the following parameters: cell size=18 nm, number of samples=182800, convergence tolerance=0.001000, radius of effect=41 nm, relative permittivity= 12.9, clock high=9.8e-22, clock low=3.8e-23, clock amplitude factor=2.000, layer separation=11.5000nm, maximum iterations per sample=1000. In our QCA layouts, we set out to create workable designs with compact layouts. Each Fredkin gate in the layouts (in the critical path) will delay the output by one cycle since Fredkin gate is designed from 4 clocking zones (please refer Fig. 6.2). Figure 9 shows the simulation results of the Fredkin gate QCA layout shown in Fig. 6.3, verified using the QCADesigner tool and it can be seen that output is produced after a delay of 1 clock cycle, as the input are applied to the Fredkin gate at clock zone 0 while the output are available at clock zone 4 (the simulation waveforms is same as the truth table of Fredkin gate which verifies the correctness of the design). An important objective in our designs was to ensure that the designs are practical and usable and hence through the QCA designer simulation, it was verified that the signals arrive properly without degradation. For example, in order to work correctly, all signals should arrive simultaneously at the QCA majority gate [122]. We also observed there is a limit on the maximum number of QCA cells that can be connected to the same clock zone because the signal deteriorates beyond the limit.

6.4.1 QCA Layout and Simulation of Proposed Latches

Figures 6.4.1 and 6.4.1 show the QCA layout and simulation of the D latch with output Q but it does not produce the complement output (QBAR). In Fig.6.4.1, the arrow-A and arrow-C show that when D=0 and E=1, we will have output as Q=0 (the input D=0 will be reflected in the output), and in the next cycle since E=0 Q will maintain its value of 0. Arrow-B shows that when E=1 and D=1, Q will become 1 (the value of D=1 will be reflected in the output), and in the next cycle since E=0 Q will maintain its value 1. All the output will be delayed by one clock cycle as Fredkin gate has the output delayed by 1 clock cycle.

Figures 6.4.1 and 6.4.1 show the QCA layout and simulation of the D latch with normal output Q, as well as the complement output QBAR. In Fig. 6.4.1, O1 and O2 represent the intermediate

output. We have used the intermediate output so that the readers can better understand the simulation results and the work can be reproduced by others (the actual output is named Q and QBAR). In Fig.6.4.1, we will get the correct output after the delay of two cycles after passing the input as we have two Fredkin gates cascaded in series. Table 6.5 summaries the working of D latch having the output Q as well as complement output QBAR (summarization of Fig. 6.4.1). The tip of the arrows A,C and E in Fig.6.4.1 represents that the value at the input D is reflected at the output Q after two clock cycle delay. Arrows B and D represents that when E=0, the output Q is same as the old value of Q (reflected two cycles after passing of inputs). This verifies the working of the D latch.

Figures 6.4.1 and 6.4.1 show the QCA layout and simulation results of JK latch, respectively. We are providing the detailed analysis of JK latch QCA layout as it demonstrates special characteristic of QCA sequential circuits and the efforts required to design functionally correct sequential circuit in QCA. In the QCA layout of JK latch, YKB represents the complement of input K generated after a delay of 1 clock cycle. Now the next Fredkin gate has three inputs one of them is YKB and others two are QBAR and J. Since YKB is produced after delay of one clock cycle thus in order to have all the input arrive simultaneously to the Fredkin gate to produce the correct result, we have delayed the input QBAR and J to the Fredkin gate by one clock cycle as shown in Fig. 6.4.1. This produces YJKQ output having the value as JQ'+K'Q after a delay of 2 clock cycles. This is passed to the 3rd Fredkin gate having other two input as E and Q. Since YJKQ is available after delay of 2 clock cycle, we have inserted two clock cycles delay to the input E and Q to make all the signals arrive simultaneously to the Fredkin gate.In the proposed JK latch we will get the correct output after delay of 4 clock cycles as 4 Fredkin gates forms the critical path to the output Q. Table 6.6 summarizes the working of the JK latch as shown in Fig.6.4.1. Arrow A shows that when J=1 and K=0 output Q becomes 1, Arrow B shows that when J=0 and K=1 we will get the output Q=0 and Arrow C shows that when J=1 and K=1 the output toggles of its current value.

We simulated and verified the T Latch and SR latch designs in QCADesigner tool. In the T latch, the output Q will reflect the input values after a delay of two clock cycles as the critical path to Q has two Fredkin gates. Similarly in the SR latch, it takes 4 clock cycles to produce the correct output due to 4 Fredkin gates in the critical path. Table 5 shows the number of Fredkin gates in the

critical path for various latches (Each Fredkin gate in the critical path has a delay of 1 clock cycles, which means that having 4 Fredkin gates in the critical path will delay the output by 4 clock cycles). Thus, the number of clock cycles after which output Q reflects the values of inputs in various latches is shown in Table 6.7. As discussed earlier, the QCA design of the Fredkin gate requires 6 majority voters with 4 clocking zones. The Fredkin gate based D latch with enable signal having only the output Q requires 1 Fredkin gate for its design as shown in Fig. 6.5(a) and Fig.6.4.1, thus needing 6 majority voters and 4 clocking zones. From Fig.6.5 and Fig. 6.4.1, it can be seen that the D latch with enable signal having both Q and Qbar outputs requires 2 Fredkin gates for its design, thus requiring 12 majority voters and 8 clock zones. While the non-testable D latch design with enable signals converted from gate level schematic in [123] to its corresponding QCA design will require 4 majority voters and 4 clock zones. The Fredkin gates based JK latch with enable signal shown in Fig.7 and Fig.14, has 4 Fredkin gates in its design, thus requiring a total of 24 majority voters and 8 clocking zones, while the non-testable design as converted from gate level schematic in [123] to its equivalent QCA design will require 6 majority voters and 4 clocking zones. Thus, the advantages associated with proposed Fredkin gate based latches regarding concurrent testing comes with some area overhead.

Table 6.5. Verification of D latch

Arrow	Input	Output (after two clock cycles)
A	E=1 D=0 Q=1	Q=0
B	E=0 D=1 Q=1	Q=1
C	E=1 D=1 Q=0	Q=1
D	E=0 D=0 Q=1	Q=1
E	E=1 D=0 Q=1	Q=0

Table 6.6. Verification of JK latch

Arrow	Input	Output (after four clock cycles delay)
A	E=1 J=1 K=0 Q=1	Q=1
B	E=1 J=0 K=1 Q=1	Q=0
C	E=1 J=1 K=1 Q=1	Q=0

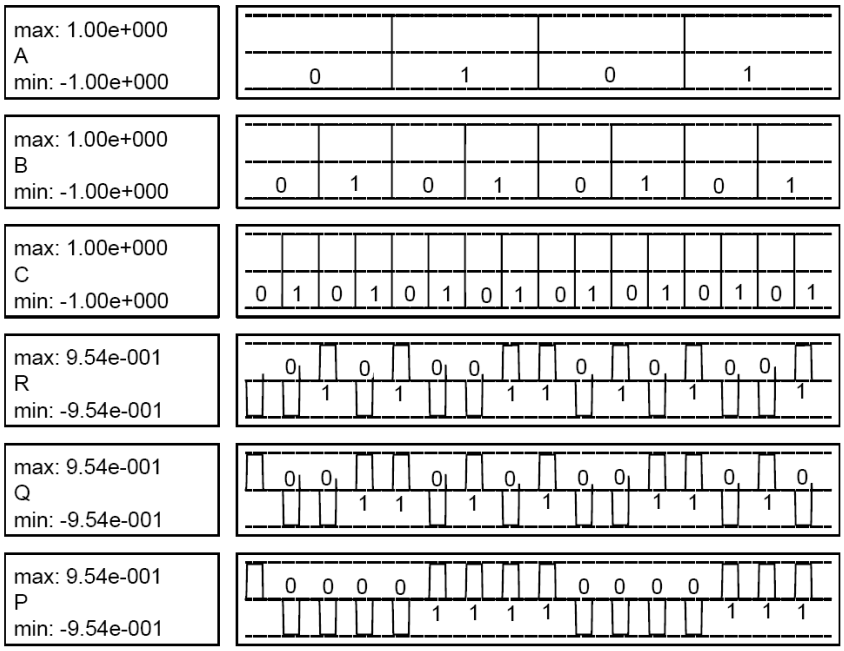


Figure 6.9. Simulation of Fredkin gate

6.5 Concurrently Testable Configurable Logic Block (CLB) Design

The logic block and the routing fabric (both are programmable) are the two key components of an FPGA. The logical functions are implemented using logic blocks (configurable logic blocks) whereas the interconnections are made through the routing fabric. The configurable logic block (CLB) internally consists of basic logic element which in turn is designed from lookup table; memory element and a D flip flop.

6.5.1 Fredkin Gate Based Lookup Table (LUT)

An n input lookup table can realize any n input Boolean function and is designed by a m:1 multiplexer (where $m = 2^n$) and m one bit storage cells. It is a well-known fact that m:1 multiplexer can be implemented with m-1 2:1 multiplexers. Fredkin gate has two of its outputs as 2:1 multiplexers, thus multiplexer based approach of designing the LUT is very much suited for designing concurrently testable LUT for molecular QCA. An example of 3 inputs LUT using Fredkin gate as

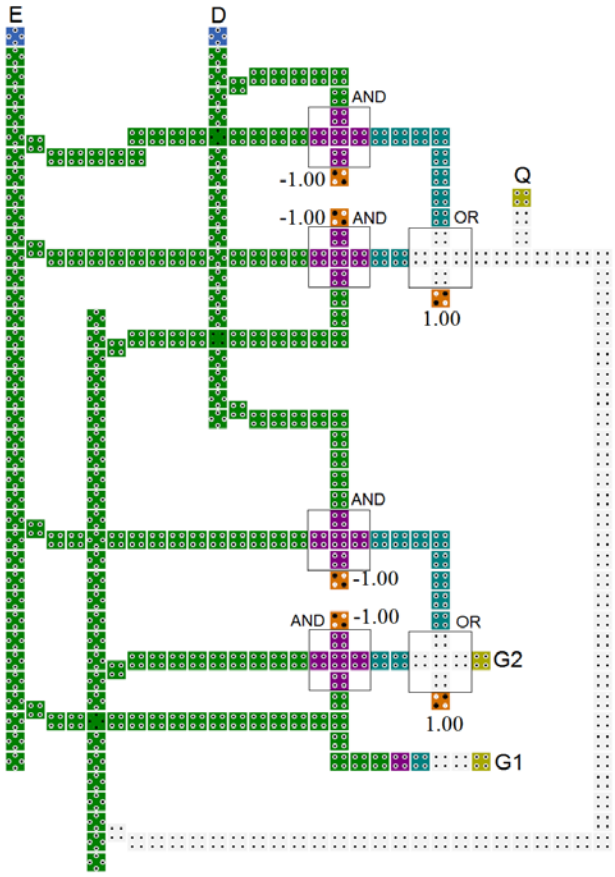


Figure 6.10. QCA layout of D latch with output Q

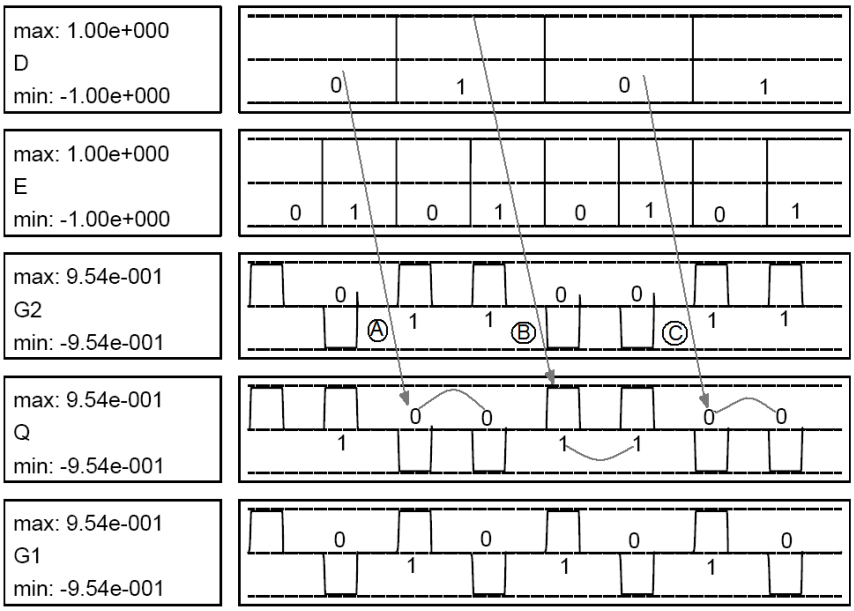


Figure 6.11. Simulation of D latch with output Q

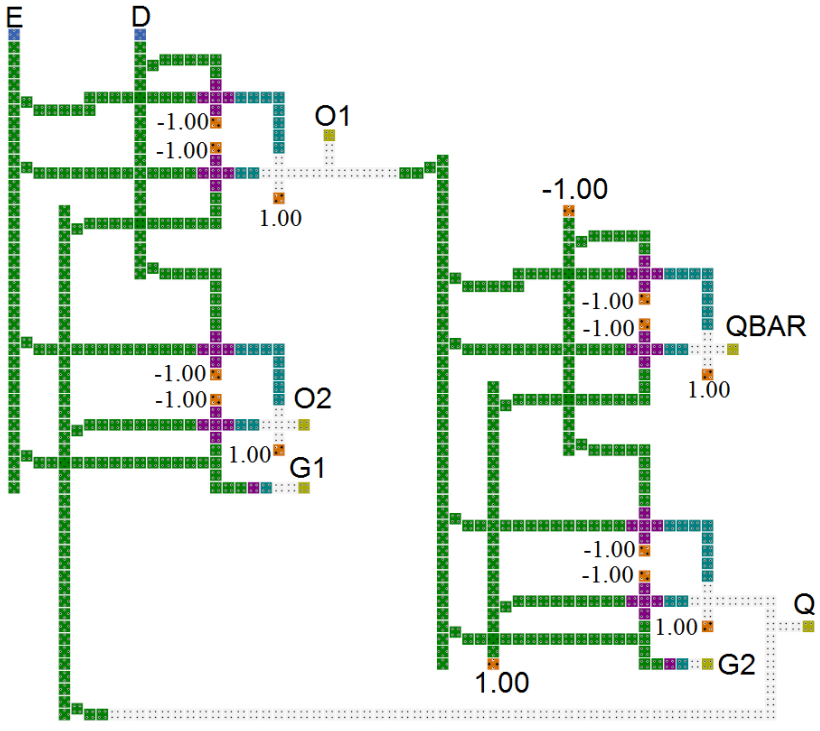


Figure 6.12. QCA layout of D latch with output Q as well as complement output QBAR

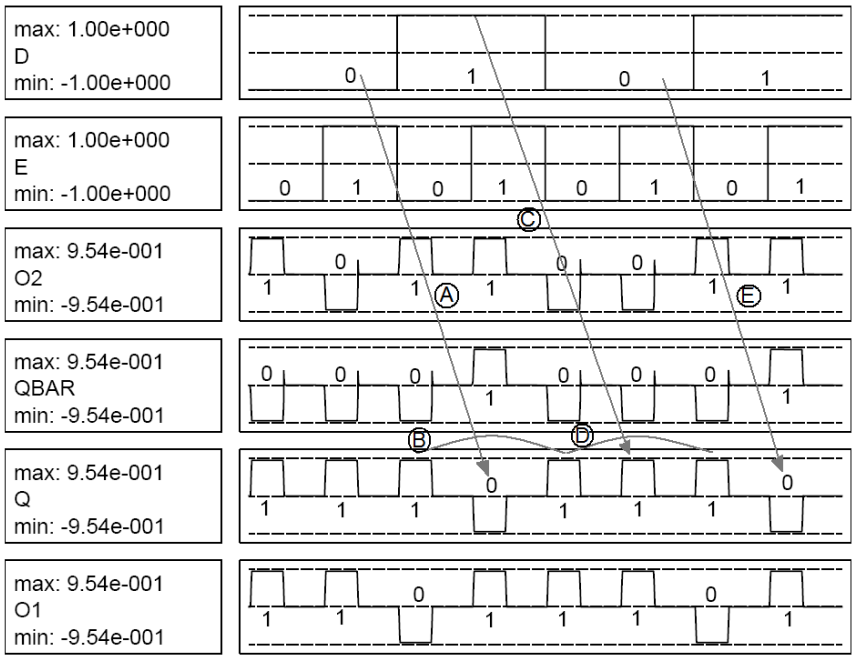


Figure 6.13. Simulation of D latch with output Q and complement output QBAR

a 2:1 multiplexer is shown in Fig. 6.16. It requires 7 reversible Fredkin gates and 8 one bit memory cells to design 3 inputs look up table. The design of a memory cell will be discussed in the next sub-section.

6.5.2 Concurrently Testable One Bit Memory Cell and D Flip Flop

We have used D Latch as a memory cell in the QCA-based FPGA as Fredkin gate can be easily modelled as a D-Latch. D flip-flop is another integral component of the basic logic element of concurrently testable QCA-based FPGA. Figure 6.17 shows the design of conservative reversible master-slave flip-flop in which the testable D latch is cascaded in master-slave fashion for its design. Master-slave D flip-flop is designed with 3 Fredkin gates as fanout is allowed in QCA design. While designing the QCA layout of sequential circuits we have make sure that the signals arrive at the same time to the Fredkin gates input. This is done by inserting the delays.

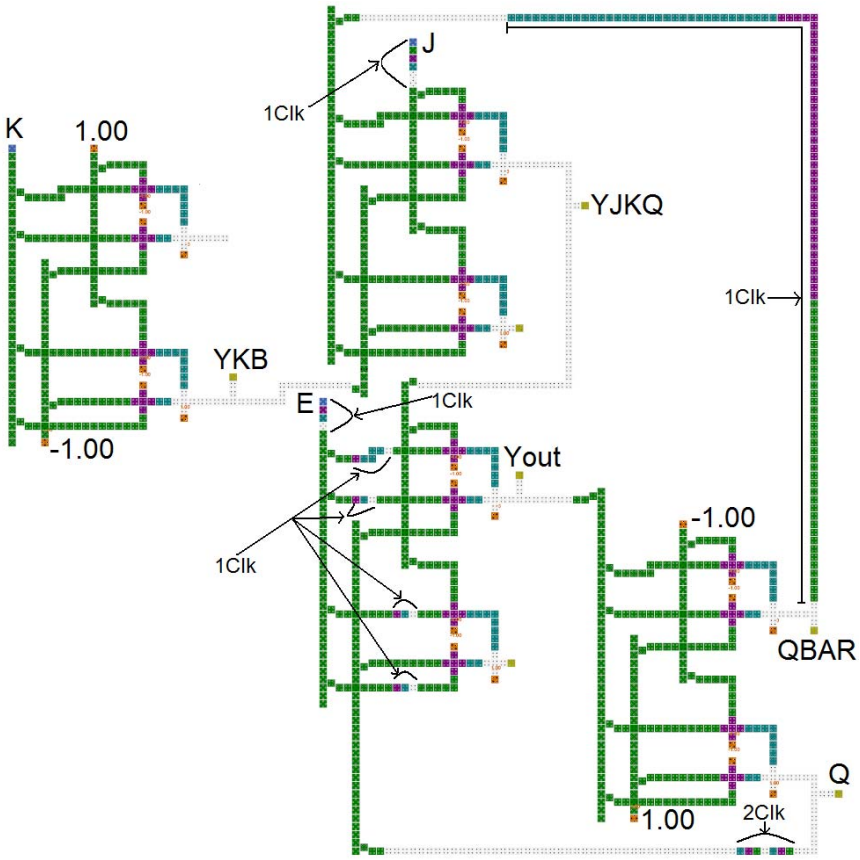


Figure 6.14. QCA layout of JK latch

6.5.3 Concurrently Testable FPGA Basic Logic Element (BLE)

One BLE consists of a LUT (lookup table), a D-FF and a 2:1 MUX. The Fredkin gate based 3 inputs LUT, Fredkin gate based D flip flop and a Fredkin gate working as a 2:1 MUX are combined to design the basic logic element of concurrently testable QCA-based FPGA. The design is shown in Fig. 6.18.

6.5.4 Concurrently Testable Configurable Logic Block (CLB)

The configurable logic block (CLB) of the proposed FPGA is designed by clustering the basic logic elements. In existing literature, the cluster based CLB architecture is well implemented in conventional technologies like CMOS [124]. We are also using the cluster-based approach of

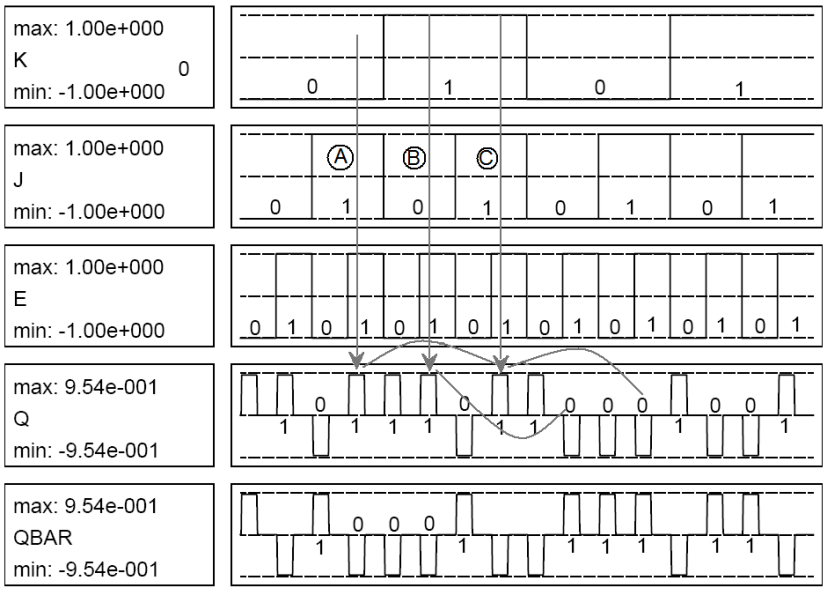


Figure 6.15. Simulation of JK latch

Table 6.7. Summary of verification of latches

	Fredkin Gates in Critical Path	Clock Cycles after which inputs reflects at outputs
D Latch with Output Q	1	1
D Latch without Fan-out	2	2
T Latch	2	2
JK Latch	4	4
SR Latch	4	4

designing the CLB from basic logic elements (BLEs). An illustration of the concurrently testable CLB design for molecular QCA is shown in Fig. 6.19. In Fig.6.19, one CLB includes 3 BLEs (basic logic elements) and 5 inputs and 3 outputs. In Fig. 6.19, FLUT represents 3 inputs lookup table based on Fredkin gate; F-DFF is Fredkin gate based D flip flop; F2:1 and F8:1 represents 2:1 and 8:1 multiplexers, respectively, designed from Fredkin gate; FAND is AND gate designed from Fredkin gate; FNOT is the inverter designed from Fredkin gate and Mcell is the memory cell designed from Fredkin gate.

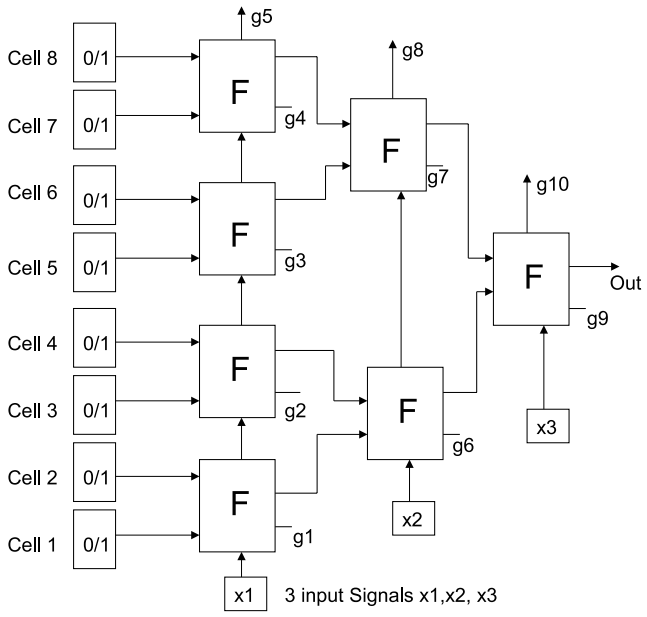


Figure 6.16. 3 inputs concurrently testable lookup table designed using 7 Fredkin gates (F) and 8 one bit reversible memory cells

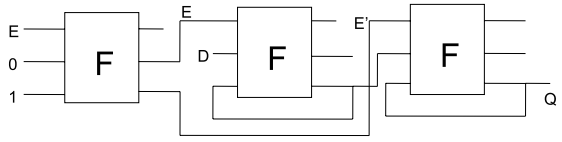


Figure 6.17. Fredkin D flip-flop

6.5.5 Concurrently Testable Routing Switch for QCA-Based FPGA

The FPGA routing switch consists of a n:1 multiplexer, a buffer and SRAM configuration cells, as shown in Fig. 6.20(a). The inputs to the multiplexer comes from the other routing conductors in the network or from the logic block outputs. The output of the buffer can be connected to a routing conductor or to a logic block input. In the routing switch, the programmability to select the input signal is realized through the SRAM configuration cells. As discussed above n:1 multiplexer can be implemented with n-1 2:1 multiplexer thus requiring n-1 Fredkin gates. The required SRAM cells can be configured as 1-bit memory cells designed as D latches from Fredkin gate. The buffer is required in conventional FPGAs to restore the signals as the switches are implemneted with pass transistor logic. We don't require any buffer in QCA implementation. A demonstration of the

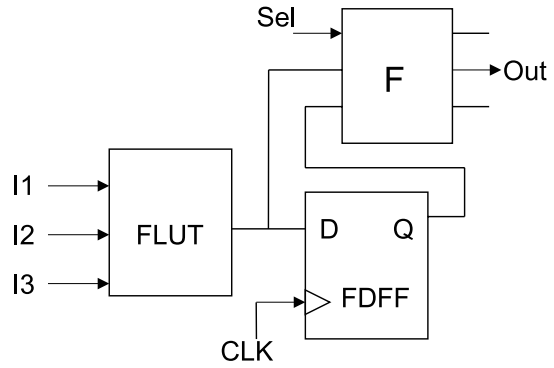


Figure 6.18. Proposed design of LUT based logic element for QCA-based FPGA

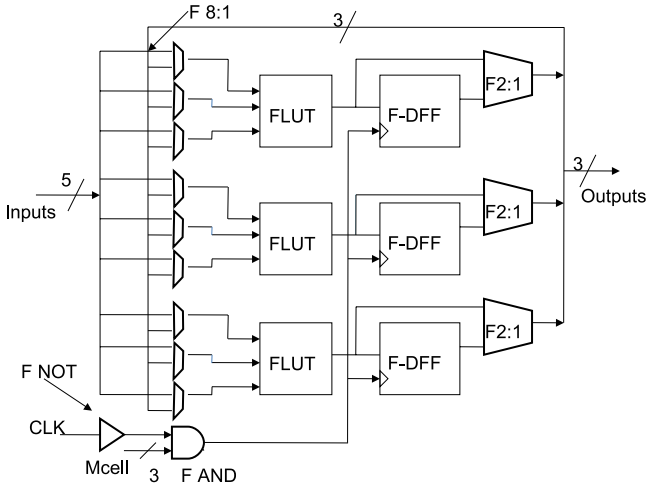
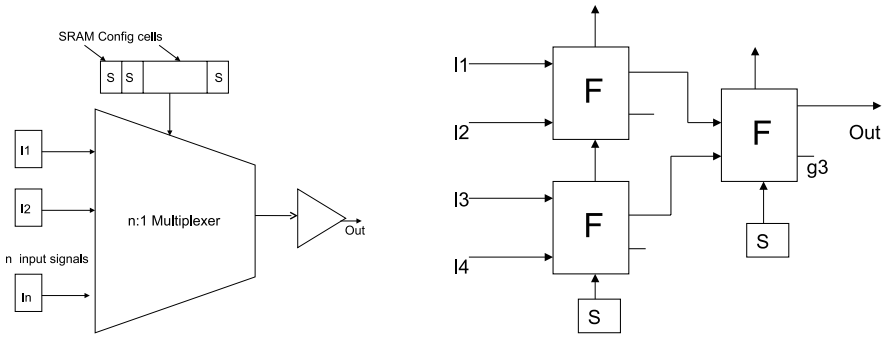


Figure 6.19. Proposed concurrently testable CLB design using clustering approach

proposed approach of designing concurrently testable routing switch is shown in Fig.6.20(b) for 4 inputs.

6.5.6 Power Dissipation in the Proposed QCA-Based FPGA

An important objective in our work is to approximate the power dissipation in the proposed QCA-based FPGA as there is no QCA tool available for power analysis. Recently in [125], an upper bound is provided for the power dissipation in QCA circuits. Over different inputs combinations, the maximum power dissipation in QCA majority gate is approximately 71.99 meV. We synthesized MCNC benchmark functions in the proposed FPGA having 4 inputs LUT(4-LUT) based BLEs using



(a) Conventional routing switch (b) Proposed routing switch

Figure 6.20. Design of routing switch

CAD flow mentioned in [124]. This gives us the number of BLEs required by each benchmark function. Each 4 inputs BLE will require 35 Fredkin gates (31 Fredkin gate for 4-LUT, 1 Fredkin gate as a 2:1 Mux and 3 Fredkin gates for D flip flop). Each Fredkin gate requires 6 majority QCA gates, thus 4 inputs BLE will have maximum power dissipation of 35 x 71.99 x 6 meV=15.117 eV. We computed the power dissipation of various synthesized MCNC benchmarks by multiplying the number of BLEs required by each with the power dissipation in each BLE. It is observed that the maximum power dissipation is for clma benchmark function having value of 125.742 x 10^3 eV. Hence, the power dissipation in proposed QCA FPGA is in eV that is negligible compared to conventional FPGA.

6.6 Proposed Majority Voter Conservative QCA Gate (MV-cqca)

The Fredkin gate is a conservative gate which has been considered widely in the literature as suitable for implementation of molecular QCA circuits based on majority voters. In this research, our objective has been to reduce the computational complexity of building majority voter based molecular QCA circuits. Thus, we develop an alternative conservative logic gate that can be used to build QCA circuits, which are more efficient than the Fredkin gates as well as incur much lesser overheads. The proposed conservative logic gate is to be used along with the Fredkin gate when the designer would be interested in sacrificing the reversibility and save the number of QCA cell, while

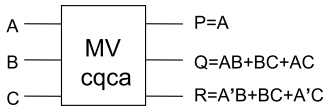


Figure 6.21. Proposed MV-cqca gate

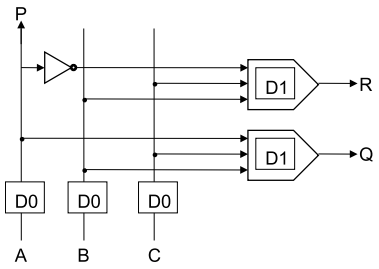


Figure 6.22. QCA design of MV-cqca gate

keep the test strategy based on conservative logic. We propose a new 3-input 3-output conservative logic gate called *majority voter conservative QCA gate, (MV-cqca)*, for use in the design of majority voting based QCA circuits. The gate has two of its outputs working as majority voters; the third output being a hardwired connection to one of the inputs. The mapping of inputs to outputs for the MV-cqca are: P= A; Q=AB+BC+AC [MV(A,B,C)] ; R=A'B+A'C+BC [MV(A',B,C)], where A, B, C are the inputs and P, Q, R are the outputs, respectively. Figure 6.21 shows the block diagram of the MV-cqca gate and Table 6.8 shows its truth table with the same number of 1s in the inputs as well as in the outputs. Figure 6.22 shows the QCA implementation of the MV-cqca gate with one level MV logic and two MVs.

Table 6.8. Truth table of MV-cqca gate

A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	0	1
0	1	1		0	1	1
1	0	0		1	0	0
1	0	1		1	1	0
1	1	0		1	1	0
1	1	1		1	1	1

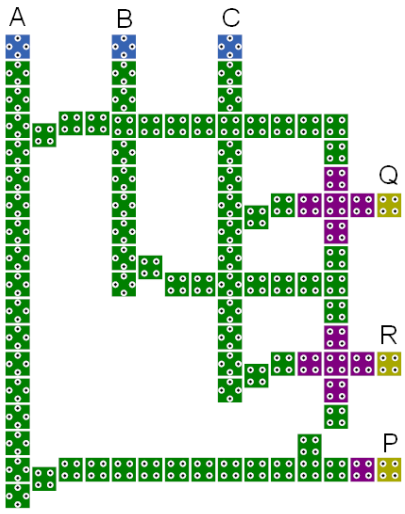


Figure 6.23. QCA layout of MV-cqca gate

The QCA layout of the MV-cqca gate is shown in Figure 6.3 and 6.23. Using the HDLQ model, we conducted exhaustive testing of the Fredkin and the MV-cqca gates with 8 input patterns in the presence of all possible single missing/additional cell defects. Testing of the MV-cqca generated 8 unique fault patterns, as shown in Table 6.9. Table 6.9 shows that by when there is no fault the number of 1s in the inputs is the same as the number of 1s in the outputs. Thus, counting the number of 1s in the inputs and the outputs can help to easily detect single missing/additional cell defect in the QCA layout of the MV-cqca gate.

Table 6.9. Fault patterns in MV-cqca gate

Input Vector	Fault Free	Fault Patterns							
		1	2	3	4	5	6	7	8
a0	a0	a0	a0	a0	a0	a0	a2	a0	a0
a1	a1	a3	a0	a3	a1	a1	a1	a0	a0
a2	a1	a3	a0	a1	a3	a3	a3	a1	a1
a3	a3	a3	a3	a1	a1	a3	a3	a3	a2
a4	a4	a4	a4	a6	a6	a4	a4	a4	a5
a5	a6	a4	a7	a6	a4	a4	a4	a6	a6
a6	a6	a4	a7	a4	a6	a6	a6	a7	a7
a7	a7	a7	a7	a7	a7	a7	a5	a7	a7

6.6.1 Comparison of Fredkin and MV-cqca Gates

Our simulation study using QCADesigner ver. 2.0.3 [90] is based on the bistable approximation method. The following parameters are used: cell size=18 nm, number of samples=182800, convergence tolerance=0.001000, radius of effect=41 nm, relative permittivity= 12.9, clock high=9.8e-22, clock low=3.8e-23, clock amplitude factor=2.000, layer separation=11.5000 nm, maximum iterations per sample=1000. Table 6.10 compares the Fredkin gate and the MV-cqca gate. In Table 6.10, since the number of clocking zones required to design the MV-cqca gate is smaller, MV-cqca is faster compared to the Fredkin gate. The total number of QCA cell required in the MV-cqca gate is only 38.2% of cells required by the Fredkin gate and the area occupied by the MV-cqca gate is only 29% of the area occupied by the Fredkin gate (Fredkin gate requires 246 QCA cell with the area of 0.37 um^2 while MV-cqca requires only 94 QCA cell with the area of 0.11 um^2). Thus, the proposed MV-cqca gate is superior to the Fredkin gate in several ways.

Table 6.10. A comparison of Fredkin and MV-cqca gates

	Fredkin	MV-cqca
Majority Voters	4	2
Clk Zones	4	2
Total Cells	246	94
Area	0.4812um x 0.76984um =0.37 um^2	0.298 um x0.38454um = 0.11 um^2

6.6.2 Comparison of Fredkin and MV-cqca Gates Based on Benchmark Functions

In order to compare the Fredkin gate and the proposed MV-cqca gate for logic synthesis, we have implemented 13 standard three variable Boolean combinational functions proposed in [126] for QCA. These 13 functions cover all the 256 Boolean functions for three variables. Table 6.11 compares the Fredkin and MV-cqca gates by synthesizing these 13 standard functions. Implementing the standard functions using the Fredkin gate requires a total of 246 MVs and 136 clock zones. For the proposed MV-cqca gate, only 86 MVs and 62 clock zones are required. Hence, implementing with MV-cqca gate achieves a reduction of 65% and 54.4% in terms of MVs and clock zones, respectively. This shows that the MV-cqca gate performs better than the Fredkin gate in terms of speed and area. We also synthesize MCNC and reversible logic benchmark functions [127] using

Table 6.11. Synthesis comparison on thirteen standard functions

No.	Function	Fredkin Implementation			MV-cqca Implementation		
		Fredkin	MV	Clk Zs	MV-cqca	MV	Clk Zs
1	F=ABC	2	12	8	2	4	4
2	F=AB	1	6	4	1	2	2
3	F=ABC+AB'C'	3	18	12	3	6	4
4	F=ABC+A'B'C'	4	24	12	6	12	8
5	F=AB+BC	2	12	8	2	4	4
6	F=AB+A'B'C	5	30	16	5	10	8
7	F=ABC+A'BC'+AB'C'	6	36	16	6	12	6
8	F=A	1	6	4	1	2	2
9	F=AB+BC+AC	5	30	16	1	2	2
10	F=AB+B'C	1	6	4	3	6	4
11	F=AB+BC+A'B'C'	6	36	16	6	12	8
12	F=AB+A'B'	2	12	8	4	8	6
13	F=ABC+A'B'C+AB'C'+A'BC'	3	18	12	3	6	4
Total		41	246	136	43	86	62

the Fredkin and MV-cqca gates for a broader analysis as shown in Table 6.12. We want to highlight here that the majority logic-based synthesis method will be directly applicable to the proposed MV-cqca gate. An example can be seen for the rd32 benchmark function, which is a full adder having the equations :

Sum= A XOR B XOR Cin

=MV(Cin', MV(A,B',Cin), MV(A',B,Cin))

Cout=AB+BC+CA =MV(A,B,Cin)

Thus, it requires only 3 MV-cqca gates to implement the rd32 benchmark function. Table 6.12 shows that implementing the 8 MCNC and reversible logic benchmark functions using the Fredkin gate requires 702 MVs and 180 clock zones, compared to 266 MVs and 88 clock zones when the MV-cqca gate is used. Therefore, implementing the MCNC and reversible logic benchmarks with the MV-cqca gate achieves a reduction of 62.1% and 51.1% in terms of the number of MVs and clock zones, respectively. Since the designs based on MV-cqca gates require fewer clocking zones compared to Fredkin gate-based designs, they will be faster in speed.

Table 6.12. Synthesis comparison on benchmark functions

No.	Benchmark	Fredkin Implementation			MV-cqca Implementation		
		Fredkin	MVs	Clk Zs	MV-cqca	MVs	Clk Zs
1	2-4 dec	6	36	8	6	12	4
2	2of5	21	126	28	23	46	14
3	6sym	30	180	36	33	66	18
4	rd32	4	24	16	3	6	4
5	con1	22	132	20	26	52	12
6	majority	13	78	28	13	26	14
7	b1	13	78	20	17	34	10
8	xor5	8	48	24	12	24	12
Total		117	702	180	133	266	88

6.7 Concurrent Multiple Error Detection Methodology For Emerging Nanocircuits

In the existing works on reversible logic, the parity mismatch between the inputs and the outputs is used for concurrent error detection. Hence, the existing works are limited for single bit error at the outputs of the reversible logic circuits.

6.7.1 Proposed Inverse and Compare Scheme

For each reversible gate R that maps each input vector X to a unique output vector Y (producing R(X)=Y), there also exist a inverse reversible gate R' which maps each input vector Y to a unique output vector X (producing R'(Y)=X). Thus, the cascading of a reversible gate with its inverse will regenerate the inputs. Figure 6.24 shows an example of n inputs reversible gate cascaded with its inverse leading to regeneration of the inputs. The cascading of a reversible logic gate with its inverse will minimize the garbage outputs. This is because regeneration of the inputs results in 'garbageless' reversible circuits as inputs are not considered as garbage signals [34]. We observed that we can use this property for concurrent detection of faults in reversible logic circuits that results in multi-bit errors at the outputs. It is to be noted that fan-out is not allowed in reversible logic. Thus, the fan-out is avoided by using the reversible Feynman gate (FG). Feynman gate is a 2 inputs 2 outputs reversible gate having inputs to outputs mapping as P=A and Q= $A \oplus B$ where P and Q are the outputs, and A and B are the inputs, respectively. Feynman gate is shown in Fig.6.25(a). Thus in

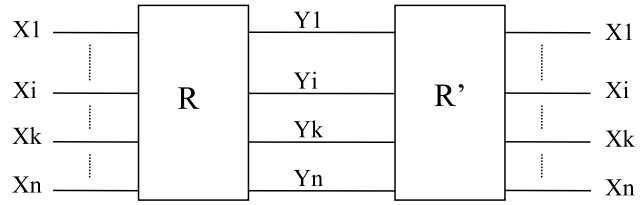
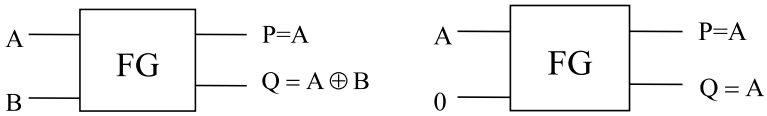


Figure 6.24. Cascading of a reversible gate R with its inverse R'

Feynman gate setting the input B to 0 as shown in Fig. 6.25(b) will copy the input A to both the outputs P and Q thus avoids the fan-out problem.



(a) Feynman gate (b) Feynman gate for copying

Figure 6.25. Feynman gate and its use for avoiding the fan-out

Figure 6.26 shows the proposed concurrent error detection methodology. In Fig. 6.26, the garbage outputs are directly passed to the inverse gate R' to regenerate the inputs. The primary output is represented by Yk where $1 \leq k \leq n$. Hence, it is passed through the Feynman gate (FG) to have the copies of the output Yk, so that one copy can be used as the primary output and the other copy can be passed to the inverse gate (R') to regenerate the input Xk. Thus, in case of faults in either R or its inverse R' or in both of them, there will be a mismatch between the regenerated inputs compared to the original inputs. The regenerated inputs can be compared with the original inputs using a comparator to detect the faults as shown in Fig.3 (the comparator is assumed fault free as generally considered in redundancy based error detection schemes [128]). To simplify the discussion, we are assuming that FG gates used for avoiding the fan-out will be fault tolerant in nature. Further, fault can be easily detected in Feynman gate because when input B is set to 0 in Feynman gate, input A and outputs P and Q should have same value. Any mismatch in the values of input A and the outputs P and Q when input B=0, will result in fault detection in the Feynman gate. Almost all nanotechnologies except quantum computing that have applications of reversible

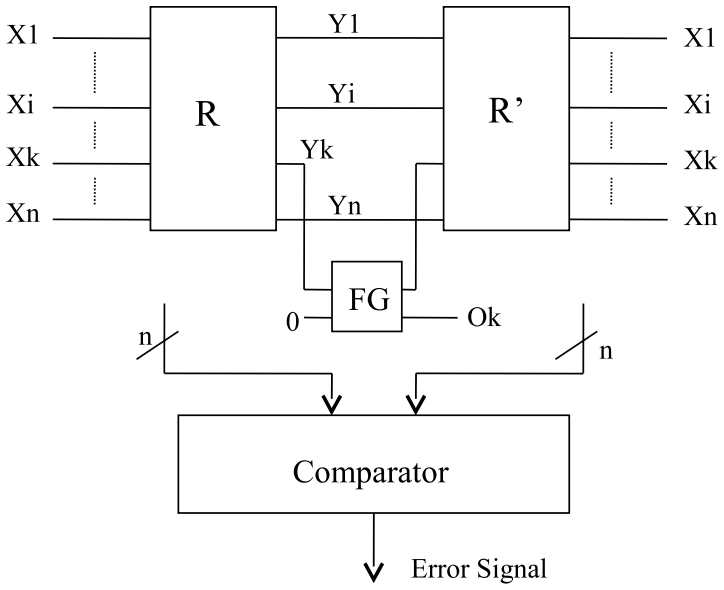


Figure 6.26. Proposed scheme for concurrent error detection for multi-bit errors at the outputs (fan-out is avoided by using Feynman gate (FG)).

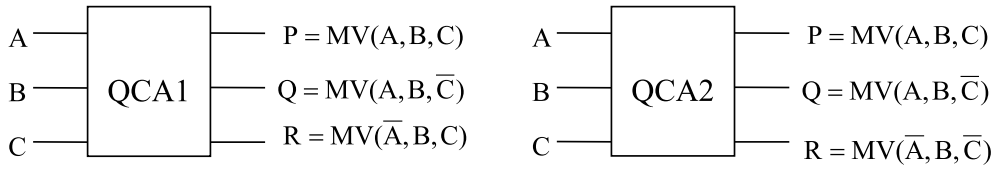
logic allows the fan-out, so Feynman gates will not be required in those nanotechnologies to copy the useful outputs. Thus the proposed scheme provides the 'garbageless' reversible logic circuits with the primary advantage of concurrent error detection.

6.7.2 Comparison of The Proposed Scheme of Concurrent Error Detection With Duplication Based Approach

The duplicate and the compare scheme that is widely used in literature for concurrent error detection has the limitation that the it won't work for the cases in which both the monitored as well as the duplicated circuit have the identical errors [128]. In this case errors can go unnoticed. The inverse and compare scheme will be beneficial in this case as it can also detect the errors when the monitored as well as the inverse circuit have the identical errors. This is because in the inverse and compare scheme we regenerate the inputs so errors in either monitored or inverse circuit or in both of them cannot go unnoticed. Further, since the proposed scheme results in garbageless reversible circuits it is especially suitable for reversible computing as the primary goal in reversible logic design and synthesis is to minimize the number of garbage outputs.

6.7.3 Application to Emerging Nanotechnologies

To demonstrate the application of the proposed approach of concurrent error detection in emerging nanotechnologies, we choose quantum dot cellular automata (QCA) nanotechnology as an example since reversible logic has potential applications in QCA computing. QCA computing is based on majority voting, thus recently two new 3x3 (3 inputs: 3 outputs) reversible gates QCA1 and QCA2 suitable for majority based QCA computing are proposed [6]. The reversible QCA1 gate can be described as mapping (A, B, C) to (P=MV(A,B,C), Q=MV(A,B,C'), R=MV(A',B,C), where A, B, C are inputs and P, Q, R are outputs, respectively. The reversible QCA2 gate can be described as mapping (A, B, C) to (P=MV(A,B,C), Q=MV(A,B,C'), R=MV(A',B,C'), where A, B, C are inputs and P, Q, R are outputs, respectively. Figure 6.27 shows the QCA1 and QCA 2 gates. Since QCA1 and QCA2 are most useful for QCA computing once we have the inverse of QCA1 and QCA2 gates, the proposed method of concurrent error detection can be applied to QCA computing. In this work, we have called the inverse of QCA1 gate as IQCA1, while the inverse of QCA2 gate is called IQCA2.



(a) QCA1 gate (b) QCA2 gate

Figure 6.27. QCA1 and QCA 2 reversible gates

In order to derive the inverse of QCA1 gate (IQCA1) we used the truth table of QCA1 gate. The truth table of the QCA1 gate is shown in Table 6.13. From Table IV, we derive the truth table of the IQCA1 gate as shown in Table 6.14. From Table V, IQCA1 can be described as mapping (P, Q, R) to (A=MV(P,Q,R'), B=MV(P,Q,R), C=MV(P,Q',R), where P, Q, R are inputs and A, B, C are outputs, respectively. Figure 6.28(a) illustrates the IQCA1 gate. Similarly for deriving the inverse of QCA2 gate (IQCA2), we used the truth table of QCA2 gate and derived the IQCA2 logic equations as

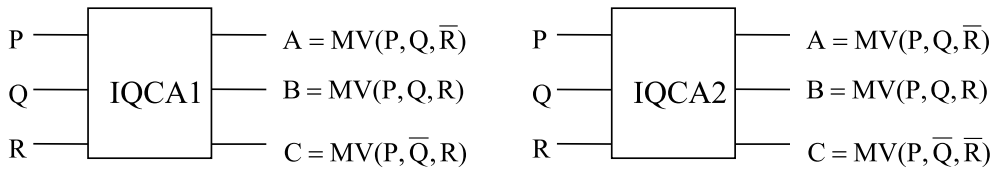
mapping (P, Q, R) to (A=MV(P,Q,R'), B=MV(P,Q,R), C=MV(P,Q',R'), where P, Q, R are input and A, B, C are output, respectively. Figure 6.28(b) shows the IQCA2 reversible gate.

Table 6.13. Truth table of QCA1

A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	1	1
0	1	1		1	0	1
1	0	0		0	1	0
1	0	1		1	0	0
1	1	0		1	1	0
1	1	1		1	1	1

Table 6.14. Truth table of IQCA1

P	Q	R		A	B	C
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		1	0	0
0	1	1		0	1	0
1	0	0		1	0	1
1	0	1		0	1	1
1	1	0		1	1	0
1	1	1		1	1	1



(a) IQCA1 gate (b) IQCA2 gate

Figure 6.28. IQCA1 and IQCA 2 reversible gates

Once we have the inverse of QCA1 and QCA2 gates, they can be cascaded with them, respectively, to regenerate the inputs for concurrent error detection of multi-bit error at the outputs in QCA computing. In Fig. 6.29, an example of the proposed approach is shown for QCA computing by combining QCA1 and IQCA1 together. In QCA computing, fan-out is allowed; hence in Fig.6.29

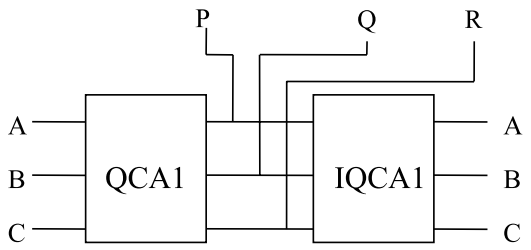


Figure 6.29. Cascading of QCA1 and IQCA1 to regenerate the inputs

we don't require the Feynman gates in the design to avoid the fan-out problem. Thus, implementing a reversible gate in QCA and cascading with its inverse will result in concurrent detection of faults in QCA circuits. The proposed methodology is independent of the reversible gate. Thus any reversible gate along with its inverse implemented in QCA technology can be used for concurrent detection of multiple faults in QCA circuits. We want to emphasize a very special characteristic of the proposed approach of concurrent testing based on reversible logic for QCA computing. By cascading a reversible logic gate with its inverse will result in dissipation less QCA circuit, as all inputs are regenerated that results in no information loss [8].

6.8 Conclusions

We propose the use of conservative reversible logic based on Fredkin gate to design concurrently testable circuits for QCA computing. The proposed concurrent testing methodology is based on conservative property of Fredkin gate and is beneficial for both permanent and transient faults which results in mismatch between the number of 1s in the inputs and the outputs. The concurrently testable designs for the latches (D Latch, T Latch, JK Latch and SR Latch), as well as complex combinational and sequential circuits such as FPGA building blocks, their QCA layouts and the simulation details are presented. Further, we have demonstrated a new methodology of concurrent error detection in reversible logic circuits. The proposed strategy is based on the inverse property of reversible logic that helps in the regeneration of the inputs. This results in detection of multi-bit errors at the outputs by comparing the original inputs with the regenerated inputs. The inverse and the compare scheme will be able to detect all types of faults in reversible logic circuits. The proposed

methodology of concurrent error detection based on property of reversible logic is generic in nature, and will be applicable to any emerging nanotechnology, such as QCA, nano-CMOS designs, which may be susceptible to single or multiple transient and permanent faults. An application of the proposed approach for concurrent error detection in emerging technologies is illustrated for QCA nanotechnology

CHAPTER 7

TWO VECTORS TESTABLE REVERSIBLE SEQUENTIAL CIRCUITS

In this chapter, we propose the design of two vectors testable sequential circuits based on conservative logic gates [129]. The proposed sequential circuits based on conservative logic gates outperforms the sequential circuits implemented in classical gates in the area of testing. Any sequential circuit based on conservative logic gates can be tested for classical unidirectional stuck-at faults using only two test vectors. The two test vectors are all 1s, and all 0s. The designs of two vector testable latches, master-slave flip-flops, double edge triggered flip-flops, asynchronous set/reset D latch and D flip-flop are presented. Our work is significant because we are providing the design for reversible sequential circuits completely testable for any stuck-at fault by only two test vectors, thereby eliminating the need for any type of scan-path access to internal memory cells. The reversible design of the double edge triggered flip-flop is proposed for the first time in literature. We also showed the application of the proposed approach towards 100% fault coverage for single missing/additional cell defect in the QCA layout of the Fredkin gate. We are also presenting a new conservative logic gate called Multiplexer Conservative QCA gate (MX-cqca) that is not reversible in nature but has similar properties as the Fredkin gate of working as 2:1 multiplexer and surpasses the Fredkin gate in terms of complexity (the number of majority voter), speed and area.

7.1 Background

As discussed in Chapter 6, a conservative logic gate is a multiple-output logic element in which the number of ls at the inputs is equal to that of the corresponding outputs. According to [34, 83], a conservative logic circuit can be considered as a directed graph whose nodes are conservative logic gates, and the edges are wires of arbitrary lengths. The fanout at the output is not allowed

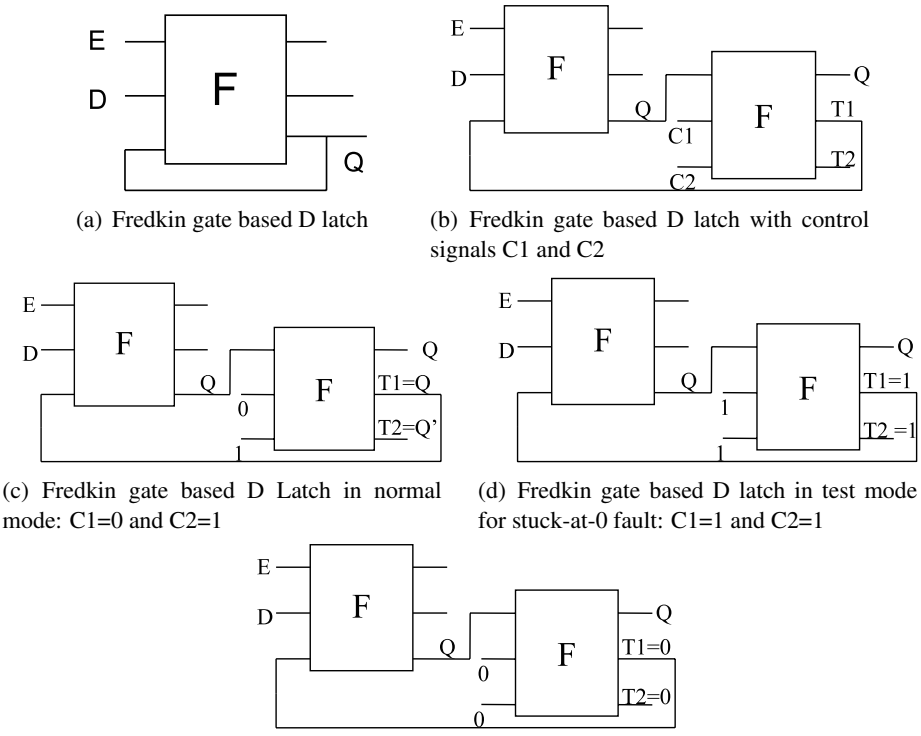
in conservative logic circuits. A conservative logic network can be reversible in nature if the one-to-one mapping is maintained between the inputs and the outputs, while it will be irreversible in nature if one-to-one mapping is not preserved. Researchers in [83–85] have proved that: (i) in the event of unidirectional stuck-at-faults in a conservative logic network, either the number of 1s in its output set will differ from the number of 1s in its input set, or the output set is correct; (ii) in a conservative logic network the two vector test set, all 1s and all 0s, provide 100% coverage for unidirectional stuck-at faults. Any stuck-at-1 fault in the conservative logic circuit can be detected by setting all inputs to 0s followed by subsequent checking of the outputs for the presence of any 1s. Any stuck-at-0 faults can be detected by setting all inputs to 1s followed by subsequent checking of outputs for the presence of any 0s. The comprehensive proofs can be referred in [83–85]. The proposed work is based on the conservative reversible Fredkin gate which is discussed in detail in Chapters 2 and 5.

7.2 Design of Testable Reversible Latches

The characteristic equation of the D latch can be written as $Q^+ = D \cdot E + \bar{E} \cdot Q$. In the proposed work E (Enable) refers to the clock and are used interchangeably in place of clock. When the enable signal (clock) is 1, the value of the input D is reflected at the output that is Q^+=D. While, when E=0 the latch maintains its previous state, that is Q^+=Q. The reversible Fredkin gate has two of its outputs working as 2:1 MUxes, thus the characteristic equation of the D latch can be mapped to the Fredkin gate (F). Figure 7.1(a) shows the realization of the reversible D latch using the Fredkin gate. But fan-out is not allowed in conservative reversible logic. Moreover, the design cannot be tested by two input vectors all 0s and all 1s because of feedback, as the output Q would latch 1 when the inputs are toggled from all 1s to all 0s and could be misinterpreted as stuck-at-1 fault.

In this work, we propose to cascade another Fredkin to output Q as shown in Fig. 7.1(b). The design has two control signals, C1 and C2. The design can work in two modes: (a) normal mode; (b) test mode. *Normal Mode* : The normal mode is shown in Fig.7.1(c) in which we will have C1C2=01 and we will have the design working as a D latch without any fanout problem. *Test Mode (Disrupt the Feedback)*: In test mode , when C1C2=00 as shown in Fig.7.1(d) it will make the design

testable with all 0s input vectors as output T1 will become 0 resulting in making it testable with all 0s input vectors. Thus any stuck-at-1 fault can be detected. When C1C2=11 as shown in Fig.7.1(e) the output T1 will become 1 and the design will become testable with all 1s input vectors for any stuck-at-0 fault. It can seen from above that C1 and C2 will disrupt the feedback in test mode, and in normal mode will take care of the fan-out. Thus our proposed design works as a reversible D latch and can be tested with only two test vectors, all 0s and all 1s, for any stuck-at fault by utilizing the inherent property of conservative reversible logic.



(a) Fredkin gate based D latch

(b) Fredkin gate based D latch with control signals C1 and C2

(c) Fredkin gate based D Latch in normal mode: C1=0 and C2=1

(d) Fredkin gate based D latch in test mode for stuck-at-0 fault: C1=1 and C2=1

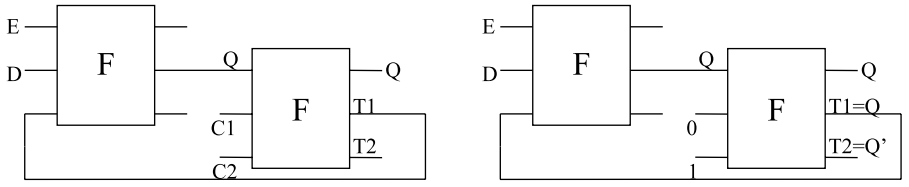
(e) Fredkin gate based D latch in test mode for stuck-at-1 fault: C1=0 and C2=0

Figure 7.1. Design of testable reversible D latch using conservative Fredkin gate

7.2.1 Design of Testable Negative Enable Reversible D Latch

A negative enable reversible D latch will pass the input D to the output Q when E=0; otherwise maintains the same state. The characteristic equation of the negative enable D latch is $Q^+ =$

$D \cdot \bar{E} + E \cdot Q$. This characteristic equation of the negative enable reversible D latch can be mapped on the 2nd output of the Fredkin gate as shown in Fig. 7.2(a). The second Fredkin gate in the design take cares of the fanout. The second Fredkin gate in the design also helps in making the design testable by two test vectors all 0s and all 1s by breaking the feedback based on control signals C1 and C2 as illustrated above for positive enable reversible D latch. The negative enable D latch is helpful in the design of testable reversible master-slave flip-flops. This is because as it can work as a slave latch in the testable reversible master-slave flip-flops in which no clock inversion is required. The details of which are discussed in the section describing reversible master-slave flip-flops.



(a) Fredkin gate based negative enable D latch with control signals C1 and C2

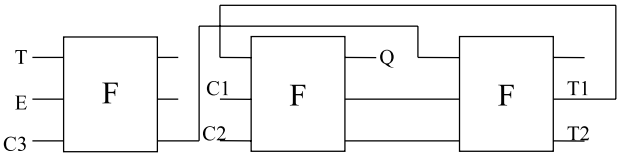
(b) Fredkin gate based negative enable D latch in normal mode

Figure 7.2. Design of testable negative enable D latch using conservative Fredkin gate

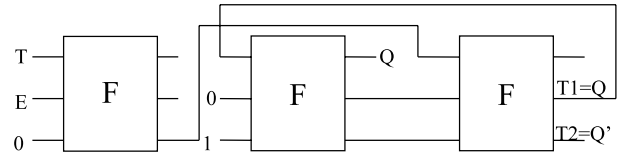
7.2.2 Design of Testable Reversible T Latch

The characteristic equation of the T latch can be written as $Q^+ = (T \cdot Q) \cdot E + \bar{E} \cdot Q$. But the same result can also be obtained from $Q^+ = (T \cdot E) \oplus Q$. The T(toggle) latch is a complementing latch which complements its value when T=1, that is when T=1 and E=1 we have Q^+ =Q'. When T=0, the T latch maintains its state and we have no change in the output. Figure 7.3(a) shows the proposed design of reversible testable T latch with C1,C2, and C3 as control signals. The C3 control signal helps to realize the reversible AND function as when C3=0 we can generate $T \cdot E$ at one of the outputs of the Fredkin gate as illustrated in Fig.7.3(b). The C1 and C2 are the main control signals that helps in breaking the feedback to make the design testable as well as enabling the normal mode of operation. In normal mode as illustrated in Fig. 7.3(b), the values of the control signals will be C1=0 and C2=1 thus helping in realizing the function $(T \cdot E) \oplus Q$. In test mode, when C1=0 and C2=0 as shown in Fig.7.3(d) it will break the feedback and helping in testing the design with all 0s

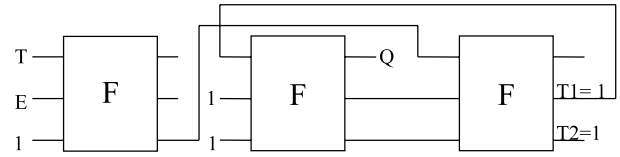
test vector for any stuck-at-1 fault, while when C1=1 and C2=1 as shown in Fig.7.3(c) it will break the feedback and helps in testing the design with all 1s test vector for any stuck-at-0 fault. The other type of reversible testable latches based on conservative reversible logic such as the JK latch and the SR latch can be designed similarly, thus are not discussed in this work.



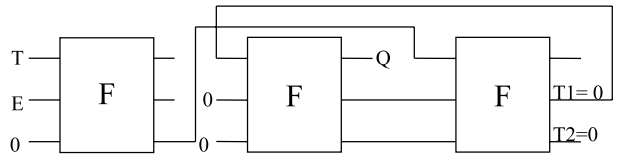
(a) Fredkin gate based T latch with control signals C1, C2 and C3, where C3 helps in realizing the AND function while C1 and C2 operates the test mode as well as the normal mode



(b) Fredkin gate based T latch in normal mode: C1=0 and C2=1



(c) Fredkin gate based T latch in test mode for detecting any stuck-at-0 fault: C1=1 and C2=1



(d) Fredkin gate based T latch in test mode for detecting any stuck-at-1 fault: C1=0 and C2=0

Figure 7.3. Design of testable reversible T latch using conservative Fredkin gate

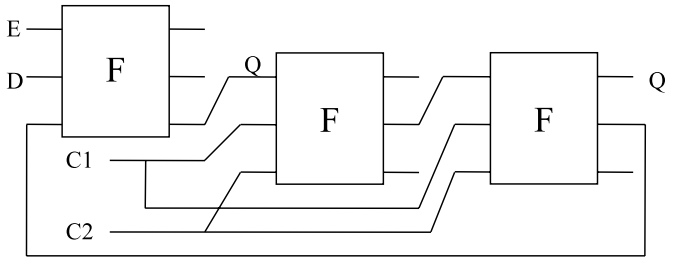
7.2.3 Design of Testable Asynchronous Set/Reset D Latch

The design of the asynchronously set/reset D latch is shown in Figure 5.10(a). The design has 3 Fredkin gates. We can observe that the first Fredkin gate maps the D latch characteristic equation, while the second and the third Fredkin gates take care of the fan-out and also help in asynchronous

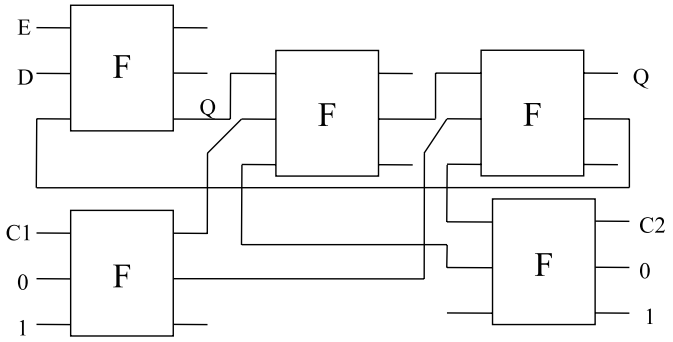
set/reset of the output Q. The design has two control inputs C1 and C2. When C1=0 and C2=1, the design works in normal mode implementing the D latch characteristic equation. When C1=0 and C2=0, the second and third Fredkin gates will reset the output Q to 0. When C1=1 and C2=1, the design will be set to Q=1. Thus, the control inputs help the design to work in various modes. But the design shown in Fig. 7.4(a) has fan-out of more than one in C1 and C2 inputs which is prohibited in reversible logic. Thus, a modified design of the D latch with asynchronous set/reset capability in which there is no fan-out is shown in Fig.7.4(b). There is a special characteristic of the reversible D latch design shown in Fig.7.4(b). The design shown in Fig. 7.4(b) has the control signals C1 and C2 which helps in disrupting the feedback. For example, the feedback is disrupted when C1C2=00; the output Q to be feedback resets to 0 which makes the reversible D latch testable with all 0s test vector for any stuck-at-1 fault. Similarly, when C1C2=11 the output Q sets to 1 and the design becomes testable with all 1s test vector for any stuck-at-0 fault. Thus, the proposed reversible D latch design with asynchronous set/reset also significantly reduces the testing cost. Thus if we design asynchronous set/reset only with Fredkin gates we can have the significant testing benefits.

7.3 Design of Testable Master-Slave Flip-Flops

In the existing literature, the master-slave strategy of using one latch as a master and the other latch as a slave is used to design the reversible flip-flops [68, 69, 74, 112]. In this work, we have proposed the design of testable flip-flops using the master-slave strategy that can be detected for any stuck-at faults using only two test vectors all 0s and all 1s. Figure 7.5(a) shows the design of the master-slave D flip-flop in which we have used positive enable Fredkin gate based testable D latch shown in Fig. 7.1(b) as the master latch, while the slave latch is designed from the negative enable Fredkin gate based testable D latch shown earlier in Fig. 7.2(a). The testable reversible D flip-flops has four control signals mC1,mC2,sC1 and sC2. mC1 and mC2 control the modes for the master latch, while sC1 and sC2 control the modes for the slave latch. In the normal mode, when the design is working as a master-slave flip-flop the values of the controls signals will be mC1=0 and mC2=1, sC1=0 and sC2=1 (as similar to values of the control signals C1 and C2 earlier described for the testable D latches).



(a) Design of testable Fredkin gate based asynchronous set/reset D latch with fanout



(b) Design of testable Fredkin gate based reversible asynchronous set/reset D latch without fanout

Figure 7.4. Design of testable reversible asynchronous set/reset D latch

In the test mode:

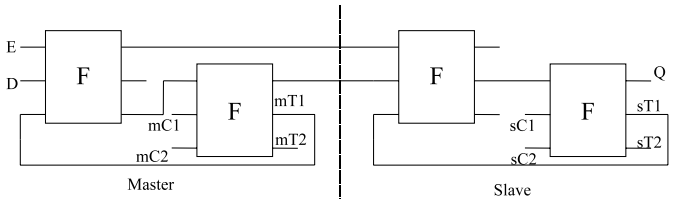
- to make the design testable with all 0s input vectors for any stuck-at-1 fault, the values of the controls signals will be mC1=0 and mC2=0, sC1=0 and sC2=0. This will make the outputs mT1 and sT1 as 0 that results in breaking the feedback and the design becomes testable with all 0s input vectors for any stuck-at-1 fault.

- to make the design testable with all 1s input vectors for any stuck-at-0 fault, the values of the control signals will be mC1=1 and mC2=1, sC1=1 and sC2=1. This will result in outputs mT1 and st1 to have the value of 1 breaking the feedback and resulting in the design testable with all 1s input vectors for any stuck-at-0 fault.

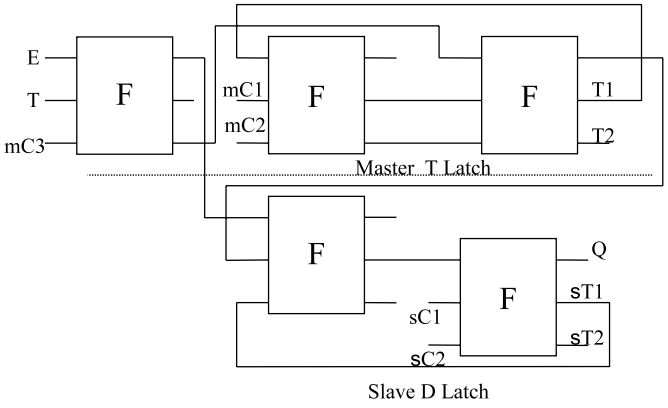
The other type of master-slave flip-flops such as the testable master-slave T flip-flop, testable master-slave JK flip-flop and testable master-slave SR flip-flop can be designed similarly in which master

is designed using the positive enable corresponding latch, while the slave is designed using the negative enable Fredkin gate based D latch. For example, as illustrated in Fig.7.5(b), in the design of master-slave T flip-flop the master is designed using the positive enable T latch, while the slave is designed with the negative enable D latch.

The reversible design of the master-slave D flip-flop with asynchronous set/reset is shown in Fig. 7.6. The design contains positive enable testable D latch shown in Fig.7.1(b) as the master latch and negative enable asynchronous set/reset D latch shown in Fig.7.4(b) as the slave latch.



(a) Fredkin gate based testable reversible master-slave D flip-flop



(b) Fredkin gate based testable reversible master-slave T flip-flop

Figure 7.5. Fredkin gate based testable reversible master-slave flip-flops

7.4 Design of Testable Reversible Double Edge Triggered (DET) Flip-Flops

The double edge triggered flip-flop is a computing circuit that sample and store the input data at both the edges, that is at both the rising and the falling edge of the clock.The master-slave strategy is the most popular way of designing the flip flop. In the proposed work E (Enable) refers to the clock and are used interchangeably in place of clock. In the negative edge triggered master-slave

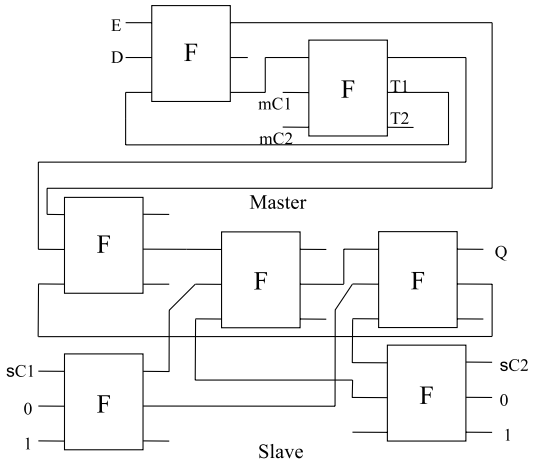


Figure 7.6. Fredkin gate based testable reversible asynchronous set/reset master-slave D flip-flop

flip-flop when E=1 (the clock is high), the master latch passes the input data while the slave latch maintains the previous state. When E=0 (the clock is low), the master latch is in the storage state while the slave latch passes the output of the master latch to its output. Thus, the flip-flop does not sample the data at both the clock levels and waits for the next rising edge of the clock to latch the data at the master latch.

In order to overcome the above problem, researchers have introduced the concept of double edge triggered (DET) flip-flops which sample the data at both the edges. Thus DET flip-flops can receive and sample two data values in a clock period thus frequency of the clock can be reduced to half of the master-slave flip flop while maintaining the same data rate. The half frequency operations make the DET flip flops very much beneficial for low power computing as frequency is proportional to power consumption in a circuit. The DET flip-flop is designed by connecting the two latches, viz., the positive enable and the negative enable in parallel rather than in series. The 2:1 MUX at the output transfer the output from one of these latches which is in the storage state (is holding its previous state). The conventional design of the DET flip-flop is illustrated in Fig.7.7(a) [130]. The equivalent testable reversible design of the DET flip flop is proposed in this work and is shown in Fig.7.7(b).

In the proposed design of testable reversible DET flip-flop, the positive enable testable reversible D latch and the negative enable testable reversible D latch are arranged in parallel. The Fredkin gates

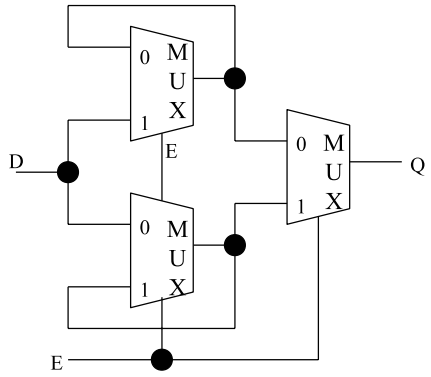
labeled as 1 and 2 forms the positive enable testable D latch while the Fredkin gates labeled as 3 and 4 forms the negative enable testable D latch. In reversible logic fanout is not allowed so the Fredkin gate labeled as 1 is used to copy the input signal D. The Fredkin gate labeled as 6 works as the 2:1 MUX and transfer the output from one of these testable latches (negative enable D latch or the positive enable D latch) that is in the storage state (is holding its previous state) to the output Q. In the proposed design of testable reversible DET flip-flop pC1 and pC2 are the controls signals of the testable positive enable D latch, while nC1 and nC2 are the control signals of the testable negative enable D latch. Depending on the values of the pC1, pC2, nC1 and nC2 the testable DET flip-flops works either in normal mode or in the testing mode.

- *Normal Mode*: The normal mode of the DET flip-flop is illustrated in Fig.7.8(a) in which the pC1=0, pC2=1, nC1=0 and nC2=1. The pC1=0, pC2=1 helps in copying the output of the positive enable D latch thus avoiding the fanout while the nC1=0 and nC2=1 helps in copying the output of the negative enable D latch thus avoiding the fanout.

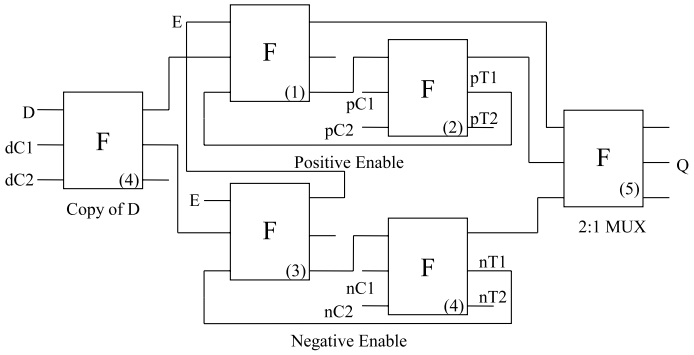
- *Test Mode*: There will be two test modes :

  - *All 1s Test Vector*: This mode is illustrated in Fig.7.8(c) in which control signals will have value as pC1=1, pC2=1, nC1=1 and nC2=1. The pC1=1 and pC2=1 help in breaking the feedback of the positive enable D latch, while the nC1=1 and nC2=1 help in breaking the feedback of the negative enable D latch. This makes the design testable by all 1s test vector for any stuck-at-0 fault.

  - *All 0s Test Vector*: This mode is illustrated in Fig.7.8(b) in which the control signals will have value as pC1=0, pC2=0, nC1=0 and nC2=0. The pC1=0 and pC2=0 help in breaking the feedback of the positive enable D latch while the nC1=0 and nC2=0 help in breaking the feedback of the negative enable D latch. This makes the design testable by all 0s test vector for any stuck-at-1 fault.



(a) Conventional DET flip-flop



(b) Fredkin gate based DET flip-flop

Figure 7.7. Fredkin gate based double edge triggered (DET) flip-flop

7.5 Application of Two Vectors Testing Approach to QCA Computing

QCA computing is one the promising technology to implement reversible logic gates. The QCA design of Fredkin gate is shown in Fig. 6.2 using the four-phase clocking scheme, in which the clocking zone is shown by the number next to D (D0 means clock 0 zone, D1 means clock 1 zone and so on). It can be seen that the Fredkin gate has two level majority voter (MV) implementation, and it requires 6 MVs and 4 clocking zones for implementation. *The number of clocking zones in a QCA circuit represents the delay of the circuit (delay between the inputs and the outputs). Higher the number of clocking zones, lower the operating speed of the circuit [6].*

The QCA layouts of the Fredkin gate is shown in Figure 6.3. In the proposed work, the QCA layout of the Fredkin gate is converted into the corresponding hardware description language no-

tations using the HDLQ Verilog library [121] as discussed in Chapter 6. The HDLQ model of the QCA layout of the Fredkin gate is shown in Fig.6.4. In the Fig.6.4, FO represents the fanout QCA device, LS represents the L-shape wire, INV represents the QCA inverter, CW represents the cross-wire, MJ represents the majority voter. Thus it can be seen that modeled QCA layout of the Fredkin gate has 4 FOs, 2 INVs, 5 CWs, 9 LSs and 6 MJs. The HDLQ modeled design of the Fredkin gate is simulated for the presence of all possible single missing/additional cell defect in MJs (majority voters), INVs (Inverters), FOs (fanouts), Crosswires(CWs) and L-shape wires (LSs). The design is simulated using the Verilog HDL simulator in the presence of faults to determine the corresponding outputs. As discussed in Chapter 6, testing of the Fredkin gate generated 28 unique fault patterns at the output, as shown in Tables 6.2, 6.3 and 6.4. From fault tables we can see that there are 10 fault patterns 5,6,13,15,18,23,24,25,26,27 that will produce the correct outputs for input vectors a0 (all 0s) and a7 (all 1s) even when there is a fault. Thus two test vectors a0 and a7 can only provide 64.28% fault coverage. Thus in order to give the test vectors a0 and a7 100% fault we identified the logic devices in the HDLQ model of the Fredkin gate which can be replaced by their fault-tolerant counterpart. This will give the 100% fault coverage for any single missing/additional cell defect to the two test vectors all 0s and all 1s. We observed that fanouts ($F0_2$ and $F0_3$), inverters (INV_1 and INV_2), crosswires (CW_4 and CW_2) and majority voters (MJ_1,MJ_3,MJ_4,MJ_5 and MJ_6) are devices in the QCA layout of the Fredkin gate that are making the design untestable by all 0s and all 1s test vectors. Thus, these devices can be replaced by their fault tolerant counterparts in the QCA layout of the Fredkin gate to have the equivalent design that gives 100% fault coverage to test vectors all 1s and all 0s. The HDLQ model of the QCA layout having 100% coverage for any single missing/additional cell defect to test vectors all 0s and all 1s is shown in Fig.7.9. In Fig.7.9, the shaded devices represent their fault tolerant counterparts. Thus, conservative logic based QCA circuits based on our proposed QCA layout of the Fredkin gate show in Fig.7.9 can be tested by all 0s and all 1s test vectors for presence single missing/additional cell defects.

7.6 Proposed Multiplexer Conservative QCA Gate (MX-cqca)

For many of the designs, the designer could potentially be interested in using the testing advantages of conservative logic but saving the number of QCA cells. Thus, in this work we propose a new conservative logic gate that is conservative in nature but is not reversible. The proposed conservative logic gate is called multiplexer conservative QCA gate(MX-cqca) and has 3 inputs and 3 outputs. Mx-cqca has one of its outputs working as a multiplexer that will help in mapping the sequential circuits based on it, while the other two outputs work as AND and OR gates, respectively. The mapping of the inputs to outputs of the MX-cqca is: P= AB; Q=AB'+BC; R=B+C, where A, B, C are the inputs and P, Q, R are the outputs, respectively. Figure 7.10 shows the block diagram of the MX-cqca gate. Table 7.1 shows the truth table of the MX-cqca gate. The table verifies the gate's conservative logic nature, i.e., that the numbers of 1s in the inputs is equal to the number of 1s in the outputs. Figures 7.11 and 7.12 shows the QCA design and layout of the proposed MX-cqca gate. From the QCA design, we can see that the proposed MX-cqca gate requires 4 clocking zones and 5 majority gates for its QCA implementation. Table 7.2 shows the comparison between the proposed MX-cqca gate and the Fredkin gate in terms of area and number of QCA cells. The table illustrates that MX-cqca is better than the existing Fredkin gate for implementing multiplexer-based designs (The MX-cqca gate requires 5 majority voters and 218 QCA cells with an area of 0.71 um^2. Thus, it has 1 less majority gate compared to the Fredkin gate, 11% less QCA cells and 5.4% less area.

We also modeled the QCA layout of the MX-cqca gate using the HDLQ Verilog libray for performing the fault testing. The HDLQ model of the QCA layout of the Fredkin gate is shown in Fig.7.13. Thus it can be seen that modeled QCA layout have 4 FOs, 1 INV, 5 CWs, 8 LSs and 5 MJs. We conducted exhaustive testing of the HDLQ model of the Mx-cqca gate with 8 input patterns in the presence of all possible single missing/additional cell defect. Testing of the Mx-cqca gate generated 24 unique fault patterns at the output, as shown in Tables 7.3,7.4 and 7.5. Due to limitation of page width the fault pattern table is divided into Tables 7.3, 7.4 and 7.5, where Table 7.3 illustrates the 8 fault patterns, Table 7.4 illustrates the next 10 fault patterns, and Table 7.5 represents the last 8 fault patterns.

From fault tables we can see that there are 9 fault patterns 3,7,13,17,19,20,22,23,24 that will produce the correct outputs for test vectors a0 and a7 (all 0s and all 1s) even when there is fault. Thus two test vectors a0 and a7 can only provide 62.5% fault coverage. Thus in order to give the test vectors 100% fault we identified the logic devices in the HDLQ model of the Mx-cqca gate which can be replaced by their fault-tolerant counterpart to give the 100% fault coverage to two test vectors all 0s and all 1s, for any single missing/additional cell defect. We observed that fanout ($F0_3$), inverter (INV_1), crosswire (CW_4) and majority voters (MJ_1,MJ_2,MJ_3,MJ_4 and MJ_5) are devices in the QCA layout of the Mx-cqca gate that are making the design untestable by all 0s and all 1s test vectors. Thus, these devices can be replaced by their fault tolerant counterparts to have the equivalent design that gives 100% fault coverage to test vectors, all 1s and all 0s, for any single missing/additional cell defect. The HDLQ model of the QCA layout having 100% coverage for single missing/additional cell defect by all 0s and all 1s test vectors is shown in Fig.7.14. The shaded devices in the Fig.7.14 represent their fault tolerant counterparts. Thus, conservative logic based QCA circuits based on the QCA layout of the Mx-cqca gate illustrated in Fig.7.14 can be tested by all 0s and all 1s test vectors for presence of single missing/additional cell defect.

Table 7.1. Truth table of MX-cqca gate

A	B	C		P	Q	R
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	0	1
0	1	1		0	1	1
1	0	0		0	1	0
1	0	1		0	1	1
1	1	0		1	0	1
1	1	1		1	1	1

Table 7.2. A comparison of Fredkin and MX-cqca gates

	Fredkin	MX-cqca
Majority Voters	6	5
Clk Zones	4	4
Total Cells	246	218
Area	0.4812um x 0.7698um = 0.37um^2	0.479um x 0.721um = 0.35 um^2

Table 7.3. Fault patterns in Mx-cqca gate (Part 1)

Input Vector	Fault Free	Fault Patterns							
		1	2	3	4	5	6	7	8
a0	a0	a0	a1	a0	a0	a0	a2	a0	a4
a1	a1	a1	a1	a1	a3	a1	a3	a1	a5
a2	a1	a1	a0	a1	a1	a5	a1	a5	a5
a3	a3	a3	a3	a3	a1	a7	a3	a7	a7
a4	a2	a6	a3	a0	a3	a2	a2	a2	a2
a5	a3	a7	a3	a1	a5	a3	a3	a3	a3
a6	a5	a1	a4	a7	a5	a1	a5	a5	a5
a7	a7	a3	a7	a7	a6	a3	a7	a7	a7

Table 7.4. Fault patterns in Mx-cqca gate (Part 2)

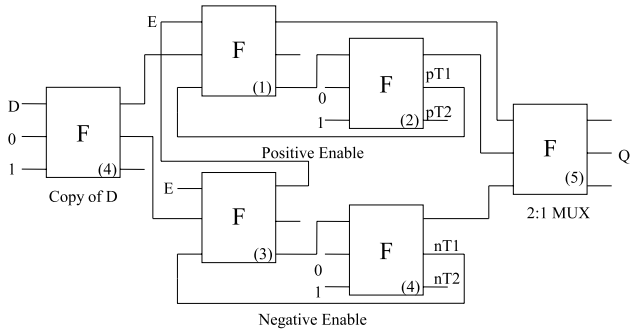
Input Vector	Fault Free	Fault Patterns							
		9	10	11	12	13	14	15	16
a0	a0	a1	a0	a0	a1	a0	a2	a1	a2
a1	a1	a0	a1	a1	a0	a1	a3	a1	a1
a2	a1	a3	a1	a3	a1	a1	a1	a1	a1
a3	a3	a1	a1	a1	a3	a3	a3	a3	a3
a4	a2	a3	a2	a2	a3	a2	a0	a2	a2
a5	a3	a2	a3	a3	a2	a1	a1	a3	a1
a6	a5	a7	a5	a7	a5	a7	a5	a4	a5
a7	a7	a3	a5	a5	a7	a7	a7	a7	a7

7.7 Conclusions

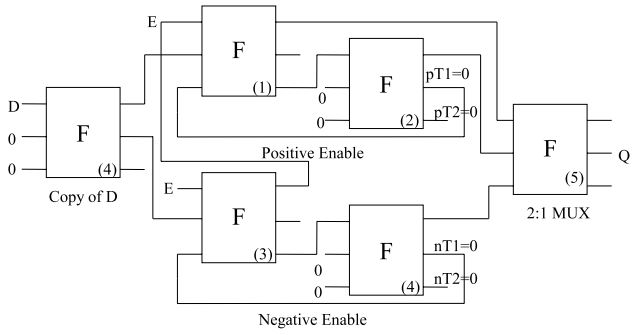
This work proposes testable reversible sequential circuits based on conservative logic. Conservative logic is testable for any unidirectional stuck-at faults using only two test vectors, all 0s and all 1s. The proposed conservative reversible sequential circuits have feedback that deters their testing by only two test vectors, thus a technique is demonstrated to disrupt the feedback in test mode. Experimental simulation on a single missing/additional cell defect has verified the application of the conservative logic towards fault testing in QCA computing. A new conservative gate that is not reversible is also proposed especially suiting QCA computing. A limitation of the proposed work is that it cannot detect multiple missing/additional cell defects. In conclusion, this work advances the state of the art of testing reversible sequential circuits based on stuck-at-fault model, as well as, reversible circuits implemented in QCA circuits having single missing/additional cell defect.

Table 7.5. Fault patterns in Mx-cqca gate (Part 3)

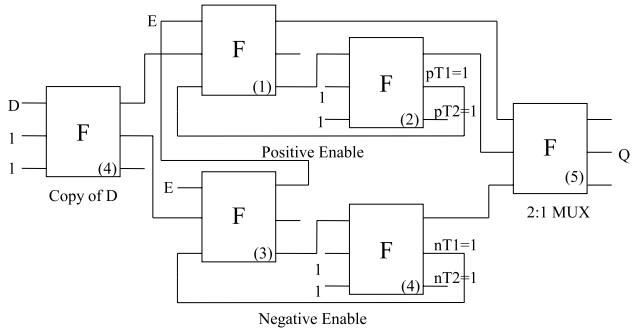
Input Vector	Fault Free	Fault Patterns							
		17	18	19	20	21	22	23	24
a0	a0	a0	a0	a0	a0	a2	a0	a0	a0
a1	a1	a0	a0	a1	a1	a3	a3	a1	a1
a2	a1	a1	a1	a1	a3	a3	a1	a1	a1
a3	a3	a3	a2	a3	a3	a3	a3	a3	a3
a4	a2	a2	a2	a2	a2	a0	a2	a0	a0
a5	a3	a2	a2	a3	a3	a1	a3	a1	a1
a6	a5	a5	a5	a7	a7	a7	a5	a5	a5
a7	a7	a7	a6	a7	a7	a7	a7	a7	a7



(a) Normal mode



(b) Test mode for stuck-at-1 fault



(c) Test mode for stuck-at-0 fault

Figure 7.8. Working of Fredkin gate based double edge triggered flip-flop

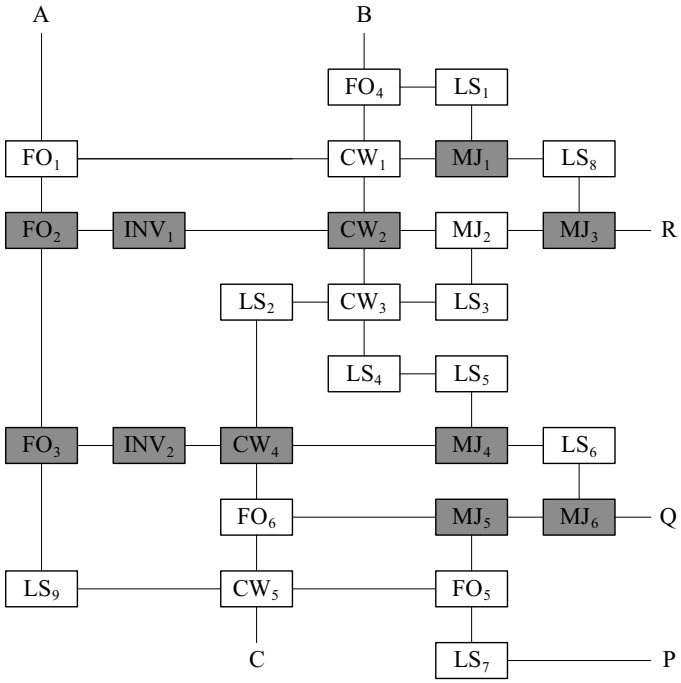


Figure 7.9. QCA layout of the Fredkin gate testable with only all 0s and all 1s test vectors for any single missing/additional cell defect (the shaded devices represent their fault tolerant counterpart)

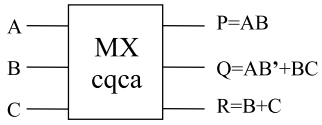


Figure 7.10. Proposed MX-cqca gate

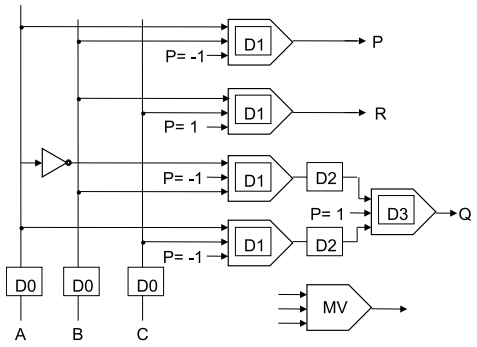


Figure 7.11. QCA design of MX-cqca gate

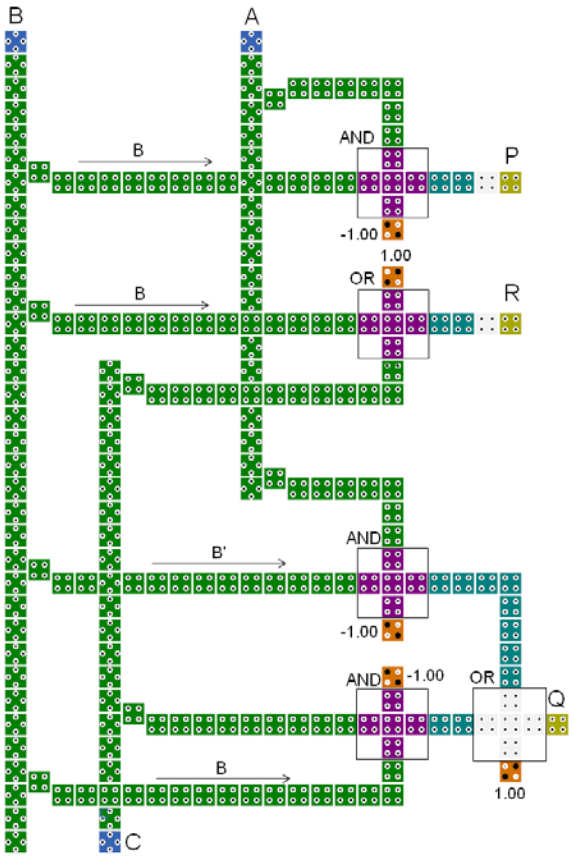


Figure 7.12. QCA layout of MX-qca gate

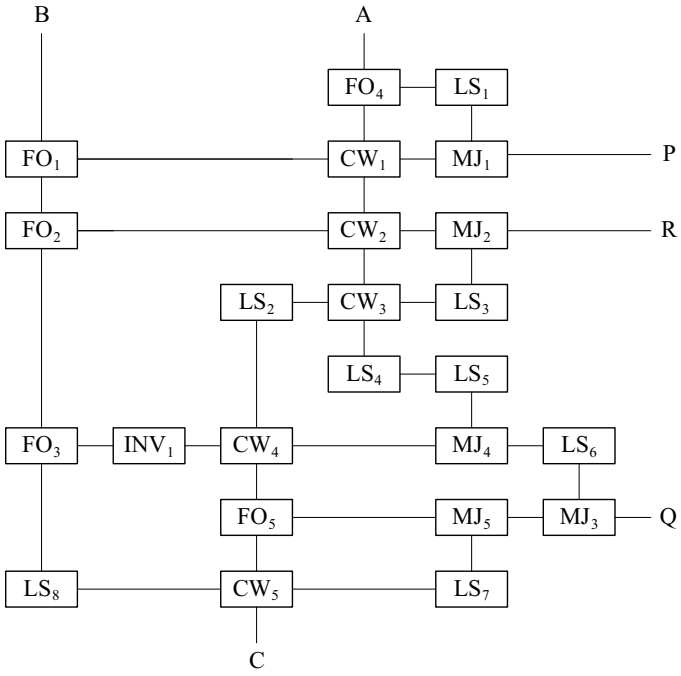


Figure 7.13. Modeling of MX-qca gate QCA layout

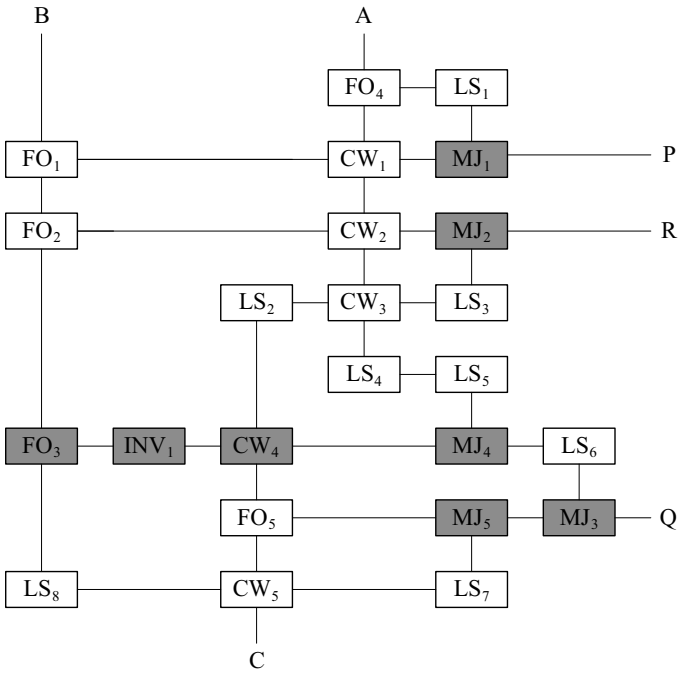


Figure 7.14. QCA layout of Mx-cqca gate testable with only all 0s and all 1s test vectors for any single missing/additional cell defect(the shaded devices represent their fault tolerant counterpart)

CHAPTER 8

CONCLUSIONS AND FUTURE DIRECTIONS

The promising computing paradigm of reversible computing has applications in the world of nanocomputing and is attracting the attention of researchers from architecture, design, synthesis and test perspective. The research efforts reported in this dissertation represent a solid contribution towards the advancement of design, synthesis and test of reversible logic circuits for emerging nanotechnologies. In this dissertation, we have developed design and synthesis methodologies for reversible logic circuits considering metrics of ancilla inputs, garbage outputs, quantum cost and the delay. Further, the fault testing schemes are explored for nanocircuits based on the inherent properties of reversible logic. The approaches presented in this dissertation have wide applicability in the various emerging nanotechnologies. Some of the future directions to improve over this dissertation work, and other interesting research ideas are listed as follows.

- Investigate new reversible gates that can help in optimization of parameters such as ancilla inputs, garbage outputs, quantum cost and delay.
- Investigate the reversible logic design and synthesis of arithmetic and sequential circuits in terms of new metrics such as Nearest Neighbor Cost (NNC). The NNC cost of a reversible gate is defined as the distance between the control and the target lines. Thus, the NNC cost of a reversible gate depends on its realistic physical quantum architectures and will vary according to the implementation [131].
- Development of design methodologies for reversible integer and floating point adder and multipliers such as the recent work in [110, 132].

- Development of design methodologies for binary and multiple-valued reversible barrel shifters such as the recent work in [133, 134].
- Development of design methodologies for reversible storage units such as such as SRAM, registers, register file design, etc.
- Development of a complete reversible CPU having the reversible logic components and the reversible instruction set.
- Development of a comprehensive synthesis tool for reversible arithmetic circuits based on the proposed synthesis framework using proposed Verilog library of reversible gates.
- Investigation of fault testing benefits of reversible logic in nano-CMOS such as 22nm technology and beyond.
- Investigation of design of fault tolerant nanocircuits based on reversible logic and conservative logic.

LIST OF REFERENCES

[1] K. V. R. M. Murali, N. Sinha, T. S. Mahesh, M. H. Levitt, K. V. Ramanathan, and A. Kumar. Quantum information processing by nuclear magnetic resonance:experimental implementation of half-adder and subtractor operations using an oriented spin-7/2 system. *Physical Review A*, 66(2):022313, 2002.

[2] Kai-Wen Cheng and Chien-Cheng Tseng. Quantum full adder and subtractor. *Electronics Letters*, 38(22):1343– 1344, Oct 2002.

[3] A. N. Al-Rabadi. Closed-system quantum logic network implementation of the viterbi algorithm. *Facta universitatis-Ser.: Elec. Energ.*, 22(1):1–33, April 2009.

[4] M. Mohammadi, M. Haghparast, M. Eshghi, and K. Navi. Minimization optimization of reversible bcd-full adder/subtractor using genetic algorithm and don't care concept. *International J. Quantum Information*, 7(5):969–989, 2009.

[5] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, New York, 2000.

[6] X. Ma, J. Huang, C. Metra, F.Lombardi. Reversible gates and testability of one dimensional arrays of molecular QCA. *J. Elect. Testing*, 24(1-3):1244–1245, jan 2008.

[7] C. Taraphdara, T. Chattopadhyay, and J.N. Roy. Machzehnder interferometer-based all-optical reversible logic gate. *Optics and Laser Technology*, 42(2):249–259, 2010.

[8] S. F. Murphy, M. Ottavi, M. Frank, and E. DeBenedictis. On the design of reversible qdca systems. *Technical Report SAND2006-5990, Sandia National Laboratories*, 2006.

[9] L.Chang, D.J. Frank,R.K. Montoye, S.J. Koester, B.L. Ji, P.W. Coteus, R.H. Dennard, W.Haensch. Practical strategies for power-efficient computing technologies. *Proc. of the IEEE*, 98(2):215–236, February 2010.

[10] M.P Frank. Introduction to reversible computing: motivation, progress, and challenges. In *Proc. ACM Int. Conf. Computing Frontiers (CF)*, page 385390, Ischia, Italy, May 2005.

[11] S. Bandyopadhyay and M. Cahay. *Introduction to Spintronics*. CRC Press, Boca Raton, 2008.

[12] Himanshu Thapliyal and M. B. Srinivas. The need of dna computing: reversible designs of adders and multipliers using fredkin gate. In *Proc. SPIE 6050,, 605010*, 2005.

[13] Himanshu Thapliyal and M. B. Srinivas. An extension to dna based fredkin gate circuits: design of reversible sequential circuits using fredkin gates. In *Proc. SPIE 6050, 60500O*, 2005.

[14] G. J. Milburn. Quantum optical fredkin gate. *Phys. Rev. Lett.*, 62(18):2124–2127, May 1989.

[15] N. Kostinski, M.P. Fok, and P.R. Prucnal. Experimental demonstration of an all-optical fiber-based fredkin gate. *Optics letters*, 34(18):2766–2768, 2009.

[16] R. Landauer. Irreversibility and heat generation in the computational process. *IBM J. Research and Development*, 5:183–191, December 1961.

[17] C.H. Bennett. Logical reversibility of computation. *IBM J. Research and Development*, 17:525–532, November 1973.

[18] Vlatko Vedral, Adriano Barenco, and Artur Ekert. Quantum networks for elementary arithmetic operations. *Phys. Rev. A*, 54(1):147–153, Jul 1996.

[19] V. K. Semenov, G. V. Danilov, and D. V. Averin. Classical and quantum operation modes of the reversible josephson-junction logic circuits. *IEEE Transactions on Applied Superconductivity*, 17:455–461, 2007.

[20] Jie Ren, Vasili K. Semenov, Yuri A. Polyakov, Dmitri V. Averin, and Jaw-Shen Tsai. Progress towards reversible computing with nsquid arrays. *IEEE Transactions on Applied Superconductivity*, 19:961–967, 2009.

[21] Jie Ren and Vasili K. Semenov. Progress with physically and logically reversible superconducting digital circuits. *IEEE Transactions on Applied Superconductivity*, 21(3):780–786, June 2011.

[22] Natalie Kostinski, Mable P. Fok, and Paul R. Prucnal. Experimental demonstration of an all-optical fiber-based fredkin gate. *Opt. Lett.*, 34(18):2766–2768, Sep 2009.

[23] M.P Frank. Approaching the physical limits of computing. In *Proc. ISMVL 2005, The Thirty-Fifth International Symposium on Multiple-Valued Logic*, pages 168–185, Calgary, Canada, May 2005.

[24] S. Kim and S.I Chae. Implementation of a simple 8-bit microprocessor with reversible energy recovery logic. In *Proc. Conf. Computing Frontiers*, pages 421–426, Ischia, Italy, May 2005.

[25] J. Lim, D.-G. Kim, and S.-I. Chae. nmos reversible energy recovery logic for ultra-low-energy applications. *IEEE Journal of Solid-State Circuits*, 35(9):865–875, June 2000.

[26] Ketan N. Patel, John P. Hayes, and Igor L. Markov. Fault testing for reversible circuits. *IEEE Trans. on CAD*, 23:410–416, 2004.

[27] Samuel Gasster. Qubit by qubit: Advancing the state of quantum information science and technology. *Crosslink Emerging Technologies*, 12(1):52–59, 2011.

[28] Saswato R. Das. Key step toward a silicon quantum computer. *IEEE Spectrum*, Oct 2010.

[29] E. Knill, R. Laflamme, and G. J. Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409:46–52, jan 2001.

[30] T. Toffoli. Reversible computing. Technical Report Tech memo MIT/LCS/TM-151, MIT Lab for Computer Science, 1980.

[31] P. Patra. Asymptotically zero power in reversible sequential machines. *Technical report, Dept. of Computer Sciences, Univ of Texas at Austin, CS Tech Report CS-TR-95-14*, 1995.

[32] M. B. Tahoori, J. Huang, M. Momenzadeh, and F. Lombardi. Testing of quantum cellular automata. *IEEE Trans. Nanotechnol.*, 3(4):432–442, December 2004.

[33] R.K. Kummamuru, A.O. Orlov, R. Ramasubramaniam , C.S. Lent, G.H. Bernstein, and G.L. Snider. Operation of a quantum-dot cellular automata (qca),shift registers and analysis of errors. *IEEE Trans. Electron Devices*, 50(9):1906–1913, September 2003.

[34] E. Fredkin and T Toffoli. Conservative logic. *International J. Theor. Physics*, 21:219–253, 1982.

[35] A. Peres. Reversible logic and quantum computers. *Phys. Rev. A, Gen. Phys.*, 32(6):3266–3276, December 1985.

[36] J. A. Smolin and D. P. DiVincenzo. Five two-bit quantum gates are sufficient to implement the quantum fredkin gate. *Physical Review A*, 53:2855–2856, 1996.

[37] W.N. N. Hung, X. Song, G.Yang, J.Yang, and M. Perkowski. Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis. *IEEE Trans. Computer-Aided Design*, 25(9):1652–1663, September 2006.

[38] D. Maslov and D. M. Miller. Comparison of the cost metrics for reversible and quantum logic synthesis. *http://arxiv.org/abs/quant-ph/0511008*, 2006.

[39] Ashis Kumer Biswas, Md. Mahmudul Hasan, Ahsan Raja Chowdhury, and Hafiz Md. Hasan Babu. Efficient approaches for designing reversible binary coded decimal adders. *Microelectron. J.*, 39(12):1693–1703, 2008.

[40] M.H.A. Khan. Design of full-adder with reversible gates. In *Proc. International Conference on Computer and Information Technology*, pages 515–519, 2002.

[41] M. Mohammadi and M. Eshghi. On figures of merit in reversible and quantum logic designs. *Quantum Information Processing*, 8(4):297–318, August 2009.

[42] P. Gupta, A. Agarwal, and N. K. Jha. An algorithm for synthesis of reversible logic ciruits. *IEEE Trans. Computer-Aided Design*, 25(11):2317–2330, Nov 2006.

[43] V. V. Shende, A.K. Prasad, I.L. Markov, and J.P. Hayes. Synthesis of reversible logic circuits. *IEEE Trans. on CAD*, 22:710–722, 2003.

[44] D. Maslov and G. W. Dueck. Reversible cascades with minimal garbage. *IEEE Trans. Computer-Aided Design*, 23(11):1497–1509, November 2004.

[45] G.Yang, X. Song, W.N. N. Hung, and M.A. Perkowski. Bi-directional synthesis of 4-bit reversible circuits. *Computer Journal*, 51(2):207–215, March 2008.

[46] A. K. Prasad, V.V. Shende, I.L. Markov, J.P. Hayes, and K. N. Patel. Data structures and algorithms for simplifying reversible circuits. *ACM JETC*, 2(4):277–293, 2006.

[47] James Donald and Niraj K. Jha. Reversible logic synthesis with fredkin and peres gates. *J. Emerg. Technol. Comput. Syst.*, 4:2:1–2:19, April 2008.

[48] D. Maslov and M. Saeedi. Reversible circuit optimization via leaving the boolean domain. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(6):806 –816, june 2011.

[49] Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi, and Zahra Sasanian. Reversible circuit synthesis using a cycle-based approach. *J. Emerg. Technol. Comput. Syst.*, 6:13:1–13:26, December 2010.

[50] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact synthesis of elementary quantum gate circuits for reversible functions with dont cares. In *Proc. of the Intl Symp. on Multi-Valued Logic*, pages 214–219, Dallas, Texas, May 2008.

[51] D. Große, R. Wille, G.W. Dueck, and R. Drechsler. Exact multiple control toffoli network synthesis with sat techniques. *IEEE Trans. on CAD*, 28(5):703715, 2009.

[52] R. Wille, M. Soeken, and R. Drechsler. Reducing the number of lines in reversible circuits. In *Proc. 47th Design Automation Conference*, pages 647–652, 2010.

[53] J. W. Bruce, M. A. Thornton, L. Shivakumaraiah, P. S. Kokate, and X. Li. Efficient adder circuits based on a conservative reversible logic gate. In *Proc. IEEE Symposium on VLSI, 2002*, pages 83–88, 2002.

[54] M. H. A. Khan and M. A. Perkowski. Quantum ternary parallel adder/subtractor with partially-look-ahead carry. *J. Systems Architecture*, 53(7):453–464, 2007.

[55] S. Offermann, R. Wille, G.W. Dueck, and R. Drechsler. A reversible mips multi-cycle control fsm design. pages 335 – 340, 2010.

[56] S. A. Cuccaro, T. G. Draper, S. A. Kutin, and D. P. Moulton. A new quantum ripple-carry addition circuit. *http://arXiv.org/quant-ph/0410184*, Oct 2004.

[57] Y. Takahashi and N. Kunihiro. A linear-size quantum circuit for addition with no ancillary qubits. *Quantum Information and Computation*, 5(6):440448, 2005.

[58] Y. Takahashi, S. Tani, and N. Kunihiro. Quantum addition circuits and unbounded fan-out. *http://arxiv.org/abs/0910.2530*, Oct 2009.

[59] A. Trisetyarso and R. V. Meter. Circuit design for a measurement-based quantum carry-lookahead adder. *http://arxiv.org/abs/0903.0748*, 2009.

[60] R.V. Meter, W.J. Munro, K. Nemoto, and K. M. Itoh. Arithmetic on a distributed-memory quantum multicomputer. *http://arxiv.org/abs/quant-ph/0607160*, 2009.

[61] Y. Takahashi. Quantum arithmetic circuits: a survey. *IEICE Trans. Fundamentals*, E92-A(5):276–1283, 2010.

[62] M.K. Thomsen and R.Glück. Optimized reversible binary-coded decimal adders. *J. Syst. Archit.*, 54(7):697–706, 2008.

[63] H. M.H. Babu and A.R. Chowdhury. Design of a compact reversible binary coded decimal adder circuit. *Elsevier Jour. of Systems Architecture*, 52:272–282, 2006.

[64] M. Mohammadi, M. Eshghi, M. Haghparast, and A. Bahrololoom. Design and optimization of reversible bcd adder/subtractor circuit for quantum and nanotechnology based systems. *World Applied Sciences Journal*, 4(6):787–792, 2008.

[65] R. K. James, K. P. Jacob1, and Sreela Sasi. Reversible binary coded decimal adders using toffoli gates. In *Proc. Advances in Computational Algorithms and Data Analysis, LNEE*, volume 15, pages 117–131, Sep 2008.

[66] P. Picton. Multi-valued sequential logic design using fredkin gates. *MVL Journal*, 1:241–251, 1996.

[67] H. Thapliyal, M. B. Srinivas, and M. Zwolinski. A beginning in the reversible logic synthesis of sequential circuits. In *Proc. the Military and Aerospace Programmable Logic Devices Intl. Conf.*, Washington, September 2005.

[68] J.E Rice. A new look at reversible memory elements. In *Proc. Intl. Symp. on Circ. and Sys. (ISCAS) 2006*, pages 243–246, Kos, Greece, May 2006.

[69] H. Thapliyal and A. P. Vinod. Design of reversible sequential elements with feasibility of transistor implementation. In *Proc. the 2007 IEEE Intl. Symp. on Cir. and Sys.*, pages 625–628, New Orleans, USA, May 2007.

[70] J. E. Rice. An introduction to reversible latches. *Comput. J.*, 51(6):700–709, 2008.

[71] K. Morita. Reversible computing and cellular automata-a survey. *Elsevier Theoretical Computer Science*, 395(1):101–131, 2008.

[72] S.K.Sastry, H.S.Shroff, S. N. Mahammad, and V. Kamakoti. Efficient building blocks for reversible sequential circuit design. In *Proc. the 49th IEEE Intl. l Midwest Symp.on Cir. and Sys.*, pages 437–441, Puerto Rico, August 2006.

[73] A. Banerjee and A. Pathak. On the synthesis of sequential reversible circuit. *arXiv/0707.4233*, 2007.

[74] Min-Lun Chuang and Chun-Yao Wang. Synthesis of reversible sequential elements. *J. Emerg. Technol. Comput. Syst.*, 3(4):1–19, 2008.

[75] Ilia Polian, Thomas Fiehn, Bernd Becker, and John P. Hayes. A family of logical fault models for reversible circuits. In *Proc. the 14th Asian Test Symposium on Asian Test Symposium*, pages 422–427, Kolkata, India, 2005.

[76] J. Zhong and J.C. Muzio. Analyzing fault models for reversible logic circuits. In *IEEE Congress on Evol .Computation*, pages 2422–2427, Vancouver, BC, 2006.

[77] H.Rahaman, D.K. Kole, D.K. Das, and B.B. Bhattacharya. On the detection of missing gate faults in reversible circuits by a universal test set. In *Proc. VLSI Design 2008, 21st International Conference on VLSI Design*, pages 163–168, Hyderabad, India, January 2008.

[78] M. Bubna, N. Goyal, and I. Sengupta. A dft methodology for detecting bridging faults in reversible logic circuits. In *Proc. 2007 IEEE Region 10 Conference , Tencon 2007*, pages 1–4, Taipei, October 2007.

[79] J.Mathew, H.Rahaman, B.R. Jose, and D.K. Pradhan. A dft methodology for detecting bridging faults in reversible logic circuits. In *Proc. VLSI Design 2008, 21st International Conference on VLSI Design*, pages 453–459, Hyderabad,India, January 2008.

[80] D. P. Vasudevan, P. K. Lala, and J. P Parkerson. Reversible-logic design with online testability. *IEEE Transactions on Instrumentation and Measurement*, 55(2):406–414, 2006.

[81] S.N. Mahammad and K. Veezhinathan. Constructing online testable circuits using reversible logic. *IEEE Trans. Instrumentation and Measurement*, 59:101 – 109, Jan 2010.

[82] N. Farazmand, M. Zamani, and M. B. Tahoori. Online fault testing of reversible logic using dual rail coding. In *Proc. IEEE International On-Line Testing Symposium*, pages 204–205, May 2010.

[83] G. Swaminathan, J. Aylor, and B. Johnson. Concurrent testing of vlsi circuits using conservative logic. In *Proc. International Conference on Computer Design (ICCD)*, pages 60–65, Cambridge, MA, September 1990.

[84] P. Kartschoke. Implementation issues in conservative logic networks. In *M.S.E.E. Thesis, University of Virginia, Charlottesville VA*, 1992.

[85] G. Swaminathan. Concurrent error detection techniques using parity. In *M.S.E.E. Thesis, University of Virginia, Charlottesville VA*, 1989.

[86] P. Tougaw and C. Lent. Logical devices implemented using quantum cellular automata. *J. Appl. Phys.*, 75(3):1818–1825, November 1994.

[87] P. Tougaw and C. Lent. Dynamic behavior of quantum cellular automata. *J. Appl. Phys.*, 80(8):4722–4736, October 1996.

[88] C.S. Lent and P.D. Tougaw. A device architecture for computing with quantum dots. *Proc. IEEE*, 85(4):541–557, 1997.

[89] J. Huang, M. Momenzadeh, and F. Lombardi. Analysis of missing and additional cell defects in sequential quantum-dot cellular automata. *Integration, the VLSI J.*, 40(1):503–515, January 2007.

[90] Qcadesigner. *http://www.qcadesigner.ca/*.

[91] C.S. Lent, B. Isaksen, and M. Lieberman. Molecular quantum-dot cellular automata. *J. Am. Chem. Soc.*, 125(4):10561063, 2003.

[92] Y. Lu, M. Liu, , and C.S. Lent. Molecular quantum-dot cellular automata: From molecular structure to circuit dynamics. *J. Applied Physics*, 102(3):034311–034311–7, 2007.

[93] M. Crocker, M.T. Niemier, X.S. Hu, and M. Lieberman. Molecular qca design with chemically reasonable constraints. *ACM Trans. On Design Automation of Electronic Sys.*, 4(2):1–21, 2008.

[94] Z. Jin. Fabrication and measurement of molecular quantum cellular automata (qca) device. Master's thesis, University of Notre Dame, 2006.

[95] P.Gupta, N.K. Jha, L.Lingappan. A test generation framework for quantum cellular automata circuits. *IEEE Trans. VLSI Sys.*, 15(1):24–36, January 2007.

[96] M. Momenzadeh, M. Ottavi, F. Lombardi. Modeling QCA defects at molecular level in combinational circuits. In *Proc. DFT in VLSI Systems*, pages 208–216, Monterey, CA, USA, October 2005.

[97] T. Wei, K. Wu, R.Karri and A. Orailoglu. Fault tolerant quantum cellular array (qca) design using triple modular redundancy with shifted operands. In *Proc. the 2005 Conf.on Asia South Pacific Design Automation*, pages 1192–1195, Shanghai, China, January 2005.

[98] T.J. Dysart and P.M. Kogge. System reliabilities when using triple modular redundancy in quantum-dot cellular automata. In *Proc. DFT in VLSI Systems*, pages 72–80, Boston,MA, October 2008.

[99] M. Momenzadeh, J. Huang and F. Lombardi. Defect characterization and tolerance of qca sequential devices and circuits. In *Proc. DFT in VLSI Systems*, pages 199–207, Monterey, CA, USA, October 2005.

[100] S. Bhanja, M. Ottavi, S. Pontarelli and F. Lombardi. Novel designs for thermally robust coplanar crossing in qca. In *Proc. the 2006 Design Automation and Test in Europe*, pages 786–791, Munich, Germany, March 2006.

[101] S Sultana, S Al Imam, K Radecka. Testing qca modular logic. In *Proc. the 13th IEEE Intl. Conf. on Electronics, Cir. and Sys.*, pages 700–703, Nice, France, December 2006.

[102] H. Thapliyal and N. Ranganathan. Design of efficient reversible binary subtractors based on a new reversible gate. In *Proc. the IEEE Computer Society Annual Symposium on VLSI*, pages 229–234, Tampa, Florida, May 2009.

[103] H. Thapliyal, N. Ranganathan, and R. Ferreira. Design of a comparator tree based on reversible logic. In *Proc. the 10th IEEE International Conference on Nanotechnology*, pages 1113–1116, Seoul, Korea, August 2010.

[104] H. Thapliyal and N. Ranganathan. A new design of the reversible subtractor circuit. In *Proc. the 11th International Conference on Nanotechnology*, pages 1430–1435, Portland, Oregon, August 2011.

[105] Ivan Oliveira, Roberto Sarthour Jr., Tito Bonagamba, Eduardo Azevedo, and Jair C. C. Freitas. *NMR Quantum Information Processing*. Elsevier Science, 2007.

[106] D. Maslov, G.W. Dueck, D.M. Miller, and C. Negrevergne. Quantum circuit simplification and level compaction. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(3):436–444, 2008.

[107] H. Thapliyal and N. Ranganathan. A new reversible design of bcd adder. In *Proc. Design Automation and Test in Europe (DATE 2011)*, pages 1180–1183, Grenoble, France, March 2011.

[108] H. Thapliyal and N. Ranganathan. Design of efficient reversible logic based binary and bcd adder circuits. *To appear ACM Journal of Emerging Technologies in Computing Systems*, 2011.

[109] T. Matsunaga and Y.Matsunaga. Customizable framework for arithmetic synthesis. In *Proc. the 12th Workshop on Synthesis And System Integration of Mixed Information technologies (SASIMI 2004)*, pages 315–318, Yokahama, 2004.

[110] M. Nachtigal, H. Thapliyal, and N. Ranganathan. Design of a reversible single precision floating point multiplier based on operand decomposition. In *Proc. the 10th IEEE International Conference on Nanotechnology*, pages 233–237, Seoul, Korea, August 2010.

[111] H. Thapliyal and N. Ranganathan. Design of reversible latches optimized for quantum cost,delay and garbage outputs. In *Proc. the 23rd International Conference on VLSI Design*, pages 235–240, Bangalore, India, January 2010.

[112] H. Thapliyal and N. Ranganathan. Design of reversible sequential circuits optimizing quantum cost, delay and garbage outputs. *ACM Journal of Emerging Technologies in Computing Systems*, 6(4):14:1–14:35, December 2010.

[113] H. Thapliyal and Mark Zwolinski. Reversible logic to cryptographic hardware: A new paradigm. In *Proc. The 49th IEEE International Midwest Symposium on Circuits and Systems*, pages 342–346, August 2006.

[114] D. Maslov. Reversible logic benchmarks, http://webhome.cs.uvic.ca/ dmaslov/, 2009.

[115] M. M. Mano. *Digital Design*. Prentice Hall, 2002.

[116] H. Thapliyal and N. Ranganathan. Reversible logic-based concurrently testable latches for molecular qca. *IEEE Trans. Nanotechnol.*, 9(1):62–69, January 2010.

[117] H. Thapliyal and N. Ranganathan. Bit conserving logic as a potential integration platform for hybrid molecular and nanoscale cmos-based architectures. In *Proc. 2009 Nanoelectronic Devices for Defense and Security (NANO-DDS) Conference*, Fort Lauderdale, September 2009.

[118] H. Thapliyal and N. Ranganathan. Conservative qca gate (cqca) for designing concurrently testable molecular qca circuits. In *Proc. the 22nd International Conference on VLSI Design*, pages 511–516, Delhi, India, January 2009.

[119] H. Thapliyal and N. Ranganathan. Concurrently testable fpga design for molecular qca using conservative reversible logic gate. In *Proc. the 2009 International Symposium on Circuits and Systems*, pages 1815–1818, Taipei, May 2009.

[120] H. Thapliyal and N. Ranganathan. Reversible logic based concurrent error detection methodology for emerging nanocircuits. In *Proc. the 10th IEEE International Conference on Nanotechnology*, pages 217–222, Seoul, Korea, August 2010.

[121] M. Ottavi, L. Schiano, and F. Lombardi. HDLQ: a HDL environment for QCA design. *ACM J. Emerging Tech.*, 2(4):243–261, October 2006.

[122] K.Kim, K. Wu, and R. Karri. Quantum-dot cellular automata design guideline. *IEICE Trans. Fundamentals*, E89-A(6):1607–1614, 2006.

[123] S. M. Kang and Y. Leblebici. *CMOS Digital Integrated Circuits Analysis and Design,2nd ed.* McGraw-Hill, New York, 1999.

[124] E. Ahmed and J. Rose. The effect of lut and cluster size on deep-submicron fpga performance and density. *IEEE Trans. VLSI*, 12(3):288–298, March 2004.

[125] S. Srivastava, S. Sarkar, and S. Bhanja. Estimation of upper bound of power dissipation in qca circuits. *IEEE Trans.on Nano.*, 8(1):116–127, January 2009.

[126] R. Zhang, K. Walus, W. Wang, and G. A. Jullien. A method of majority logic reduction for quantum cellular automata. *IEEE Trans. Nanotechnol.*, 3(4):443–450, December 2004.

[127] Reversible logic benchmarks. *http://webhome.cs.uvic.ca/ dmaslov/*.

[128] S. Mitra. Diversity techniques for concurrent error detection. In *PhD Thesis, Department of Electrical Engineering, Stanford University*, 2000.

[129] H. Thapliyal and N. Ranganathan. Testable reversible latches for molecular QCA. In *Proc. IEEE NANO 2008*, pages 699–702, Arlington,TX, August 2008.

[130] M. Pedram, Q.Wu, and X.Wu. A new design for double edge triggered flip-flops. In *Proc. Asia South Pacific Design Automation Conf.*, page 417421, Yokahama, 1998.

[131] Robert Willey, Mehdi Saeedi, and Rolf Drechsler. Synthesis of reversible functions beyond gate count and quantum cost. *http://arxiv.org/abs/1004.4609*, 2010.

[132] M. Nachtigal, H. Thapliyal, and N. Ranganathan. Design of a reversible floating-point adder architecture. In *Proc. the 11th IEEE International Conference on Nanotechnology*, pages 451–456, Portland, Oregon, August 2011.

[133] S. Kotiyal, H. Thapliyal, and N. Ranganathan. Design of a ternary barrel shifter using multiple-valued reversible logic. In *Proc. the 10th IEEE International Conference on Nanotechnology*, pages 1104–1108, Seoul, Korea, August 2010.

[134] S. Kotiyal, H. Thapliyal, and N. Ranganathan. Design of a reversible bidirectional barrel shifter. In *Proc. the 11th IEEE International Conference on Nanotechnology*, pages 463–468, Portland, Oregon, August 2011.

ABOUT THE AUTHOR

Himanshu Thapliyal received the B.Tech. degree in Computer Engineering from G.B. Pant University, India, in 2004, and the M.S. degree in VLSI and embedded systems from IIIT Hyderabad, India, in 2006. He is currently working toward the Ph.D. degree in the Department of Computer Science and Engineering, University of South Florida, Tampa. He has authored or coauthored more than 40 articles in refereed conferences and journals and is a co-owner of a U.S. patent issued recently. In 2009, he received the Distinguished Graduate Achievement Award from the Graduate and Professional Student Council at USF for outstanding research and academic achievement. In 2010, the IEEE Computer Society awarded him with the Richard E. Merwin Scholarship in recognition of his contributions to student chapter activities, academic achievement and for serving as a student ambassador. In 2011, he received 2010 UPE/CS Award for Academic Excellence from the Upsilon Pi Epsilon Honor Society for the Computing Sciences and the IEEE computer Society. His recent work on reversible logic has been featured in MIT Technology Review, ACM TechNews, New Scientist Magazine, insideHPC, Softpedia News, etc.