Thesis for the Degree of Doctor of Philosophy

Design and Implementation of Reversible Programmable Devices

Registration No: 174/2015 - 2016

Department of Computer Science and Engineering

University of Dhaka

Dhaka, Bangladesh

December 2019

Thesis for the Degree of Doctor of Philosophy

Design and Implementation of Reversible Programmable Devices

Nazma Tara

Department of Computer Science and Engineering
University of Dhaka
Dhaka, Bangladesh

December 2019

Design and Implementation of Reversible Programmable Devices

By

Nazma Tara

Registration No: 174/2015 - 2016

Supervised by

Moinul Islam Zaber, Ph.D.

Prof. Hafiz Md. Hasan Babu, Ph.D.

Submitted to the Department of Computer Science and Engineering of the Faculty of the Engineering and Technology in University of Dhaka for partial fulfillment of the requirements of the degree of Doctor of Philosophy

Dissertation Committee:

Prof. Md. Abdul Mottalib, Ph.D. (Convener)

Prof. Nazrul Islam Mondal, Ph.D. (External Member)

Moinul Islam Zaber, Ph.D. (Supervisor)

Design and Implementation of Reversible Programmable Devices

By

Nazma Tara

Registration No: 174/2015 - 2016

Supervised by

Moinul Islam Zaber, Ph.D.

Prof. Hafiz Md. Hasan Babu, Ph.D.

Submitted to the Department of Computer Science and Engineering of the Faculty of the Engineering and Technology in University of Dhaka for partial fulfillment of the requirements of the degree of Doctor of Philosophy

Declaration

We declare that this thesis titled "Design and Implementation of Reversible Programmable Logic Devices" and the works presented in it are our. We confirm that:

- The full part of the work is done during Ph.D. research study in University of Dhaka, Bangladesh.
- Any part of this thesis has not previously been submitted for a degree or any other qualification in this University or any other institution.
- We have consulted the published works of others with appropriate references.
- This thesis work is done entirely by us and our contributions and enhancements from other works are stated.

Signed:

_____

Candidate

Countersigned:

_____

Supervisor: Dr. Moinul Islam Zaber

_____

Co-supervisor: Dr. Hafiz Md. Hasan Babu

Abstract

Reversible logic is a computing design, where the ideal implementation would produce zero entropy gain. This unique feature causes the prominent use of reversible computing. More integration capability and regular structure for synthesizing a large number of logic functions made programmable devices enthusiastic to use.

In this thesis, we describe efficient design procedures of two programmable devices namely Programmable Logic Array (PLA) and Field Programmable Gate Array (FPGA) with reversible logic gate.

In the first part of this thesis, we design the reversible Programmable Logic Array (RPLA). Here, we propose an efficient algorithm to design the RPLA with a newly proposed 3 × 3 reversible TB (Tara-Babu) gate, which can realize multi-output ESOP (Exclusive-OR Sum of Product) functions. We present a heuristic algorithm to sort and realize the product terms of ESOP functions to share the internal sub-products to reduce the number of gates in the proposed RPLA. Proposed algorithms make the RPLA more efficient with improvement of 9.83% in terms of the number of gates, 21.3% in terms of the number of garbage outputs and 14.75% in terms of quantum cost than the counter metrics of the existing RPLA averagely. Moreover, we compute the area requirement and the power consumption of the proposed RPLA. We also analyze the performances by using MCNC benchmark functions.

In the last part of this thesis, we design the most significant part of a Field Programmable Gate Array, the Plessey Logic Block with reversible gate. On the way to design the proposed reversible Plessey Logic Block, we design each component such as reversible D-Latch, reversible Decoder, reversible Multiplexer, reversible Master-Slave Flip-Flop, and reversible RAM separately. The proposed design of the individual component is primarily made efficient in terms of the number of

gates, garbage outputs, quantum cost, and delay. In addition, area and power are reduced to ensure the power efficiency of the circuits. Two 4×4 reversible gates, namely HNF (Hafiz-Naz-Flip-Flop) gate and HND (Hafiz-Naz-Decoder) gate are proposed to achieve the optimization goal. Moreover, proposed algorithms, lemmas and theorems certify the novelty of the proposed design. Compared to previous works, the proposed counter-parts of Reversible Plessey Logic Block require less number of gates, garbage outputs, quantum cost, and delay. Finally, the proposed Reversible Plessey (4×2) Logic Block is compared with existing designs. The Comparative results prove the efficacy and novelty of the proposed design showing improvement of 51.62% in terms of number of Transistor, 73.57% in terms of area requirement and 34.12% in terms of power consumption with respect to the corresponding metrics of the best existing design in the literature.

Dedicated to Those People

Who are Honest, Spiritual and Patriot

Who Work for the Welfare of Human Being

Acknowledgements

Praise be to Allah, Lord of the universe, most gracious, most merciful. No one can anything without his blessings.

My deepest thanks and gratitude to Dr. Moinul Islam Zaber and Prof. Dr. Hafiz Md. Hasan Babu, for suggesting the topic of this thesis, and their kind supervision. It is a great honor to work under their supervision. Their guidance, encouragement and confidence in me has transformed me into a better researcher. I am conveying my gratefulness to them for facilitating the work environment also.

I am indebted to the CSE department faculties, especially Prof. Dr. Sabbir Ahmed, Prof. Dr. Md. Abdur Razzaque, Dr. Asif Hossain Khan, Dr. Mamunur Rashid, Abu Ahmed Ferdous and CSE department staffs for helping me to finish the long journey successful. A special thanks to Dr. Ahsan Raja Chowdhury, former faculty of CSE department, who is my mentor in this research area.

It is a great opportunity for me to be a member of the VLSI Research Group, Department of Computer Science and Engineering, University of Dhaka. Along the way, a great support comes directly or indirectly from every single group member. I also thank the members of other groups for their cordial help, support, suggestions during the whole period of research. Thanks go to my fellow researchers Dr. Lafifa Jamal, Dr. Selina Sharmin and other researchers Md. Samsujjoha, Md. Mubin Ul Haque and Zarrin Tasnim Sworna. Their direct support, inspiring words and comments was appreciative.

I must thank the ministry of ICT of Government of the People's Republic of Bangladesh for the innovation fund of research.

Hartiest thanks to my husband S.M. Bashir Ahmed, sons S.M.F. Imbesat Ahmed and S.M.N. Intesar Ahmed who continue assist me till the end of the journey. Their support, understanding, encouragement and sacrifice make this thesis possible. I am also deeply indebted to my mother Rokeya Begum, father Shah Md. Monsur Ali, mother-in-law Rehana Begum and father-in-law S. M. Mansur Ahmed, whose

affection, love, encouragement and prayers of day and night make me able to finish the research. My other family members also show their continuous support during this hard time.

Nazma Tara
December 2019

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 1961, Landauer presented a physical principle [1] of pertaining to the lower theoretical limit of energy consumption. This principle explores that the erasure of a bit consumes $KTln2$ joules of energy while computing, where, K is the Boltzmann constant and T the absolute temperature at which computation is performed. This energy dissipation is very small (at room temperature $2.9 * 10^{-21}$ Joules [2]), which is not negligible but noticeable if Moore's law is effective. The historical trend in microelectronics (according to Moore [3]) shows the number of transistor in a dense integrated circuit doubles about every two years. Investigation in [4] also reveals that the lower limit of estimated energy density of a two-dimensional system of binary switches is about 5-10 million W/cm^2 which is much higher than the energy dissipation of sun surface nearly $6000W/cm^2$. Hence, with the increasing number of transistor in a single IC, the energy reducing capacity of the IC must be ensured in physical and logical design. Therefore, to keep the trend of scaling in modern ICs, reducing the energy dissipation during computation is getting important. Modern fabrication process and material technologies are trying to reduce the energy dissipation of ICs. In this context, in 1973, Bennett [5] has shown reversible computing would be the future trend to trace the energy dissipation problem as his research proclaims that the reversible computing produces zero energy in ideal cases.

Importance of reversible logic is also comprehensible by its usages in quantum computing which can solve some exponential problem in polynomial time [6, 7] and all the quantum computations are necessarily reversible [6, 8]. Therefore, research on reversible is beneficial to the development of future quantum technologies: reversible design methods might give rise to methods of quantum circuit construction, resulting in much more powerful computers and computations. Besides the quantum technologies [9, 10, 11, 12, 13, 14, 15], some other applications of reversible logic are the optical computing [16, 17], particular adiabatic circuit [18], nanotechnology [19, 20], and DNA technology [16]. These technologies exploit reversible logic to reduce power consumption.

The research on reversible logic is expanding toward both design and synthesis. Several researchers have been exploring techniques for synthesis of reversible logic circuits and many interesting contributions have been made [21, 22, 23, 24, 25, 26, 27]. Authors in [28] propose a bidirectional, transformation based algorithm which can synthesize any reversible logic function with minimum number of constant inputs, by using generalized Toffoli gates. Afterward, authors in [29, 30, 31] present some automated reversible and quantum logic circuit synthesis methods based on genetic algorithm (GA) and evolutionary algorithm (EA). They use the global optimization properties of these algorithm to synthesize reversible and quantum circuits for obtaining near optimal circuits. Recently, in [32, 33], interesting contributions have been made toward deriving exact minimal elementary quantum gate realization of reversible combinational circuits. However, in the synthesis of the reversible logic, only few works address the optimization in terms of delay.

Reversible arithmetic units such as adder, subtractors, multiplier, divider which form the essential components of a computing system have also been designed in binary as well as ternary logic as in [34, 35]. Recent work on reversible sequential

circuits are presented in [36].

On the way of designing circuits in reversible logic, most of the constructed circuits are application specific. This application Specific Integrated Circuit (ASIC) is virtually every type of chip that performs a dedicated task. ASIC has limited usage in industry level for lacking flexibility for changes, expensiveness and difficulty to test and debug. To mitigate these problems, digital industry widely use the programmable devices (PLDs). Zero NRE cost, dense architecture with very high performance improvements make PLDs a very attractive alternative to ASICs. This thesis emphasis on the two major categories of PLD, namely Programmable Logic Array (PLA) and Field Programmable Gate Array (FPGA).

PLA has the advantages in terms of regularity over the other conventional circuits like cascade networks [37, 38, 39, 40]. It ensures an understandable and designable regular circuit. Authors in [38] also ensure that the PLA can be used to design binary valued and multiple-valued logic circuits. Different synthesis methods exist which can realize and minimize the PLA [37, 38, 39]. For example, adders are realized using minimized PLA in [41]. ESOP synthesis in PLA gives out better result than SOP realization [42, 43]. In consequence, authors in [44] propose the reversible wave cascade of ESOP synthesis and authors in [45] propose garbage minimization technique. Besides, authors in [46] [47] and [48] continue their research in ESOP realization.

Over-whelmed opportunities of FPGAs and energy saving characteristics of reversible logic have caught the eyes of the researchers to design the FPGAs in power recurring ways and it leads the design of the reversible FPGAs [49, 50, 51, 52]. Authors in [49] focus on the LUT based logic block of FPGA, whereas, others researches focus on Plessey Logic block of FPGA. Their lacking in generality, scalability and efficiency lead the proposed reversible FPGA.

This thesis propose enhanced design methodologies to design PLA and FPGA with reduce cost matrics.

1.1 Methodologies of this Research

While working on this research, some important steps are followed:

- Understanding Reversible Logic and recent research trend in Reversible Logic. Studying on Reversible circuit design procedure, its advantages in creating low power devices over existing irreversible logic design. The basics of quantum computation and quantum circuit synthesis are also studied.
- Understanding the properties and uses of various existing reversible logic gates and construction procedure new gate from the existing one. Studding on how the cost parameters of the reversible circuits are assigned and how the circuits are realized in transistor level.
- Simulation processes of reversible circuits are also studied.
- Understanding Programmable Logic Devices such as PLA, FPGA and then gain knowledge to implement the existing Programmable Logic Devices in reversible way.
- Developing the idea for logic synthesis mechanism in reversible Programmable Logic Devices and design the individual components of reversible FPGA.
- Experiment the proposed circuits with some of the popular simulation mechanisms to make a comparison of the new method with the existing methods.

1.2 Challenges of this Research

The synthesis methods of reversible logic circuits are quite different from the synthesis methods of irreversible logic circuits. Two restrictions on reversible logic synthesis must be followed [53]:

- The fanout of a logic gate must be one.
- A combinational reversible network has to be loop-free.

The first restriction is listed because a fanout structure is not reversible. To cope up with this problem, we use the Feynman gate and Feynman double gate in this thesis. As reversible combinational function is necessarily a finite one-to-one function, the second restriction is considered when reversible circuits are designed. To overcome this restriction in designing reversible sequential circuit, researchers want to assure that the transition function of sequential circuit is constructed by reversible logic [54].
The primary design criteria for efficient reversible logic synthesis are as follows:

- The reversible circuits maximize the gate utilization and thus minimize the number of gates.
- They use as many outputs of every gate as possible, and thus minimize the garbage outputs.
- Efficient circuits also ensures the minimization of quantum cost and delay.
- The reversible circuits should take minimum area and consume less power.

1.3 Contributions of this Research

In this dissertation, we have designed different reversible logic circuits associated with the Reversible Programmable Logic Array and the Reversible Plessey Logic Block of FPGA. We have also proposed design methodology to address the above

important points regarding reversible logic. The dissertation has the following contributions toward the design and synthesis of reversible logic circuits regarding PLA and FPGA.

- The first contribution of this dissertation is the design of three new reversible gates namely the TB (Tara-Babu) gate, HNF (Hafiz-Naz-Flip-Flop) gate, and HND (Hafiz-Naz-Decoder) gate.

  The proposed reversible TB gate is a universal gate as it can implement all basic boolean functions (AND, OR, NOT Ex-OR) by using one TB gate and compound functions (NAND and NOR) by using two cascaded TB gates. In this dissertation, we use this gate to realize multi-output ESOP (Exclusive-OR Sum of Product) functions. Quantum cost of the proposed TB gate is 4 and the delay of this gate is 4Δ which are the least in the literature for generating two AND terms simultaneously.

  We propose the reversible HNF (Hafiz-Naz-Flip-Flop) gate to design different sequential circuits like D-Latch and reversible Master-Slave Flip-Flop and HND (Hafiz-Naz-Decoder) gate to design reversible Decoder circuit. These two gates are also efficient in terms of the parameters such as garbage outputs, quantum cost and delay.

  Construction of a new gate and its quantum circuit is a challenge in reversible logic synthesis as there is no specific guideline. Toffoli or Fredkin synthesis do not give the optimal result all the time. Besides, quantum circuit generation for proposed gate is also laborious. In this thesis we have done this successfully.

- The second contribution is the design of reversible Programmable Logic Array. We have developed of two heuristic algorithms, one is to sort and realize the product terms of ESOP functions and another is to reorder the output

functions. We have simulated the circuits and analyzed the performance of PLAs constructed by the proposed algorithms. The design procedure ensure the proposed circuits are efficient in terms of the number of gates, garbage outputs, and quantum cost.

- The third and the final contribution of this dissertation is the design of reversible Field Programmable Gate Array (FPGA). The significant part of the FPGA is the array of logic blocks which execute the basic logical and arithmetic calculations. Hence, this thesis emphasises on the design of the Plessey logic block of FPGA. On the way to design reversible FPGA, a set of different reversible combinational circuits such as decoder, multiplexer, NAND unit as well as different sequential circuits such as reversible D-Latch, reversible Master-Slave Flip-Flop, and reversible RAM are designed. Then, the Plessey logic block is constructed by incorporating all of these proposed components. The designs are optimal in terms of number of garbage outputs as well as quantum cost and delay. We have also computed the number of transistors, the area requirement and the power consumption of the proposed components and then compared with the existing counterparts which also show the efficiency for each individual case.

We expect that the proposed work will encourage a new paradigm of designing programmable devices based on reversible logic.

1.4 Organization of the Dissertation

We organize the dissertation as follows.

In **Chapter 2**, theoretical aspects of reversible logic are described. Notations, essential definitions, examples and analysis methods are presented here which are associated with reversible logic theory as well as the architectural theory of programmable logic devices.

Chapter 3 focuses on the efficient design of Reversible Programmable Logic Array (RPLA) for ESOP functions. An introduction regarding Programmable Logic Array covers the architectural issues of the device. Then, previous work on RPLA is analyzed and summarized to point out the weaknesses of previous approaches. Proposed section also discusses the procedures to mitigate the problems of the previous designs.

Then the **Chapter 4** illustrates on efficient Reversible Plessey Logic Block of FPGA. This chapter also describe the theoretical aspects of FPGA, indicate the limitation of previous reversible Plessey Logic Block of FPGA and its components. Finally, it discusses the design procedure to enrich the design of reversible Plessey Logic Block of FPGA.

The dissertation concludes in **Chapter 5** which summarizes the contributions described in this thesis and gives some directions of the further research in the reversible logic area and the reversible programmable devices.

Chapter 2

Background Studies

2.1 Introduction

The underneath component of the digital computer is a digital circuit whose behavior is expressed as the binary function comprise with binary variables. Each of these variables has only one truth value '0' or '1' but not both and follows the rules of Boolean algebra. A binary function (Boolean function) can be represented in many ways, among them the truth table method is the simplest one. Let a Boolean function F has n variables $x_1, x_2, x_3, \ldots, x_n$ then the truth table will be as follows:

It is to note that the truth table for single function with n input variables has

TABLE 2.1: Truth table method

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

n+1 columns and 2^n rows.

Example 1. For 3-input 1-output the truth values of the function are shown in Table 2.1.

A multiple output function can also be represented by the truth table having 2^n rows representing the inputs and $(n+m)$ columns, where n is the number of input variables and m is the number of output variables.

Example 2. The 3-input 3-output multiple output Boolean function given in Table 2.2.

TABLE 2.2: Truth table for a 3-input 3-output function

x_1	x_2	x_3	f_1	f_2	f_3
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

The main object in reversible logic theory is the reversible function. The multiple output Boolean function F= $x_1, x_2, x_3, \ldots, x_n$ of n Boolean variables is called reversible if:

1. the number of outputs is equal to the number of inputs;

2. there is a unique one to one and onto relation between inputs and corresponding outputs. In other words, reversible functions are those that perform permutations of the set of input vectors.

Example 3. A 2-input 2-output function given by formula $f_1 = x_1$, $f_2 = x \oplus y$ is reversible. The correctness of this statement can be verified by analyzing the Table 2.3.

TABLE 2.3: Reversible function computing the logical Ex-OR

x	y	x	$x \oplus y$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Example 4. A 2-input single output function given by formula $f_1 = x \oplus y$ is not reversible, since it is not an n-input n-output function. However, it can easily be made reversible by adding output x as given in Example 3.

Example 5. A 2-input 2-output function given by formula $f_1 = x$, $f_2 = x.y$ is not reversible also. Though it is an n-input n-output function, there is no unique input-output mapping. This statement can be verified by analyzing the Table 2.4. The two input combinations (0,0) and (0,1) have the same output combination (0,0). In this case, additional input and output are required to make the function reversible.

TABLE 2.4: Reversible function computing the logical AND

x	y	x	$x.y$
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	1

The functions are represented by the small unit of circuit called gate. An irreversible Ex-OR gate is shown in Fig. 2.1 (a), where A, B and P are the inputs and outputs respectively, whereas, a reversible Ex-OR gate (which is Feynman gate) is shown in Fig. 2.1, where (A, B) and (P, Q) are the inputs and outputs respectively and $P = A$, $Q = A \oplus B$

(a) Irreversible Ex-OR gate (b) Reversible Ex-OR gate.

FIG. 2.1: Ex-OR function representations by block diagram and logical symbol.

2.2 Basic Definitions in Reversible Logic

In this section, we present the definition of the reversible gates and their uses, quantum cost, the challenges of designing reversible circuits, delay computation in reversible circuits.

2.2.1 Reversible Gate

Reversible gate [53] is the unit of the reversible circuit. It holds the property of a bijection function, i.e., both the one-to-one and onto relationship between the inputs and outputs vector are present so that the output can be easily retrive from the unique output for each input. Let, the input vector be I_v, output vector O_v and $I_v = (I_0, I_1, I_2, \ldots I_{n-1}, I_n,)$ and $O_v = (O_0, O_1, O_2 \ldots O_{n-1}, O_n)$. For each particular input, there exits the relationship $I_v \leftrightarrow O_v$.

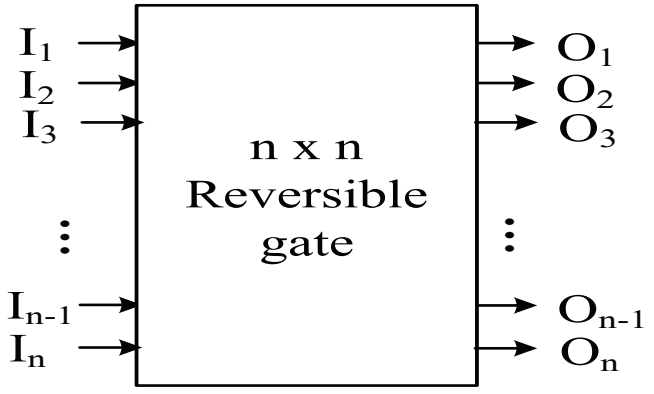


FIG. 2.2: n × n reversible gate

Example 6. Fig. 2.2 shows the block diagram of the $n \times n$ reversible gate.

Example 7. For a specific example, let consider the NOT gate, the only conventional gate which is reversible by its nature. The truth table of NOT gate is shown in Table 2.5, which shows that the output 1 comes from the input 0 and the output 0 comes from the input 1, thus the unique input-output patterns are hold.

TABLE 2.5: Truth table of NOT gate

Input	*Output*
0	1
1	0

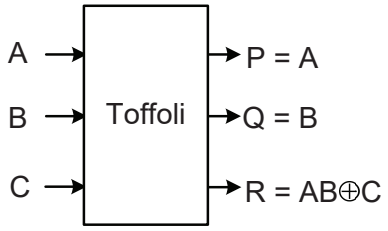


FIG. 2.3: Block diagram Toffoli gate.

TABLE 2.6: Truth table of reversible Toffoli gate

Input			Output		
A	B	C	$P = A$	$Q = B$	$R = AB \oplus C$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	1	0
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	0	1	0

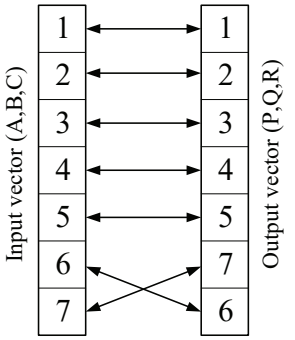


FIG. 2.4: Bijection properties of Toffoli gate.

Example 8. Fig. 2.3 shows the block diagram of the most popular reversible Toffoli gate. Table 2.6 shows the truth table of this gate. Therefore, Fig. 2.4 reveals the bijection relationship collecting the information from the Toffoli's truth table.

The Toffoli gate plays an important role in reversible logic synthesis. The most importantly, it is a universal gate which functions as any type of gate, such as, AND gate, OR gate, NAND gate, and etc. Fig. 2.5 represents three basic functions of a universal gate.

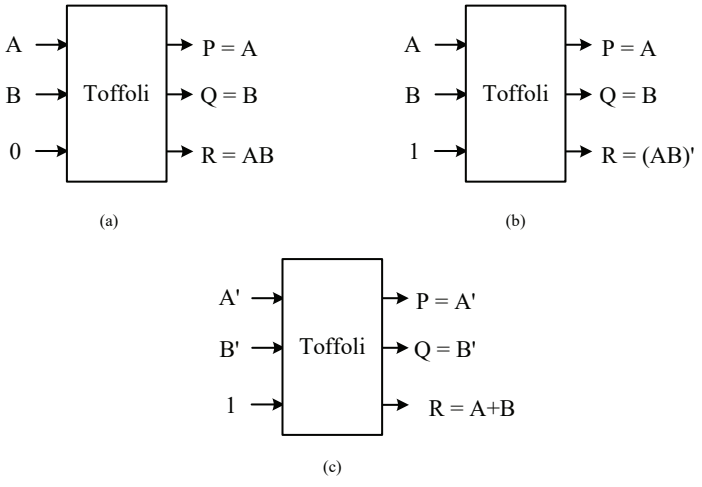


FIG. 2.5: Universal properties of Toffoli gate.

The reversible gates are not uniform in terms of number of input-output, rather than they are constructed such a fashion that they can represent one or more

specific function. Therefore, there are a variety of reversible gate in the literature, such as, Feynman gate [55], Feynman Double gate [55], Toffoli gate [53], Fredkin gate [53], Peres gate [56], Modified Fredkin gate [57], HNG gate [58], MTSG gate [59], etc.

2.2.2 Garbage Output

Garbage output is an additional output that makes an n-input and m-output function reversible [53].

Example 9. Fig. 2.3 shows the block diagram of Toffoli gate. In case of performing only one AND operation, the outputs P and Q are the garbage outputs.

2.2.3 Quantum Cost

Quantum cost is the cost associated with every reversible gate [9, 60]. Every reversible gate has a corresponding quantum circuit which is the combination of the basic quantum gates: NOT, V, V^+ and Ex-OR (CNOT) gate, where, V is a square-root-of NOT gate and V^+ is V's Hermitian. Fig. 2.6(a)-(d) represent all

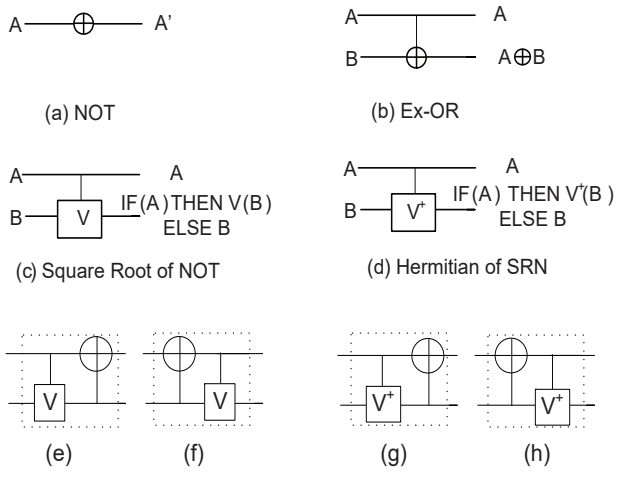


FIG. 2.6: Basic quantum gates and their symmetric patterns.

of the four basic quantum gates, respectively. Each of these basic gates and their symmetric patterns (Figure 2.6 (e)-(h)) has unit cost [9, 61, 62]. The quantum cost

of a reversible gate is the accumulated number of basic gates and their symmetric patterns in the corresponding quantum circuit.

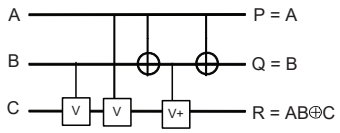


FIG. 2.7: Quantum circuit of Toffoli gate.

Example 10. Fig. 2.7 shows the quantum circuit of the reversible Toffoli gate. It shows the Toffoli gate has the quantum cost 5.

Quantum cost is one of the major cost parameters of the reversible circuit as it relates with the physical implementation cost of that circuit [62]. In designing a perticular circuit, lower value of quantum cost indicates the efficiency of that design. Example 11 can clear this.

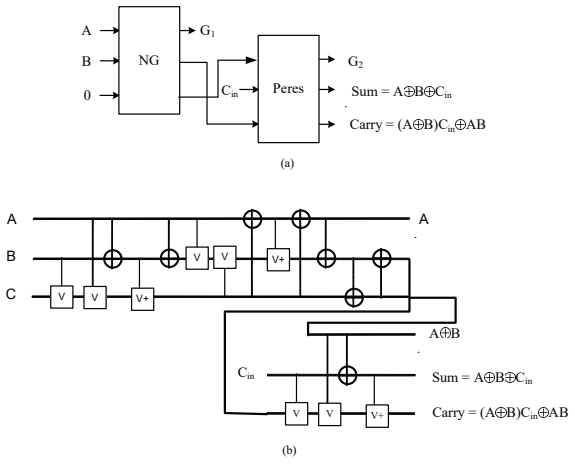


FIG. 2.8: Reversible full-adder1 (a) Block diagram (b) Quantum circuit.

Example 11. Fig. 2.8(b) and Fig. 2.9(b) shows two quantum circuits of the reversible full-adder having quantum cost 17 and 8, respectively. The later one has less quantum cost, hence more efficient.

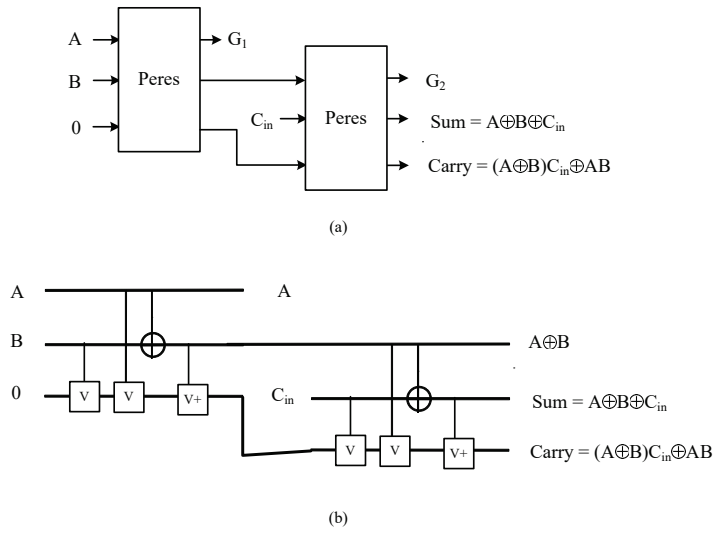


FIG. 2.9: Reversible full-adder2 (a) Block diagram (b) Quantum circuit.

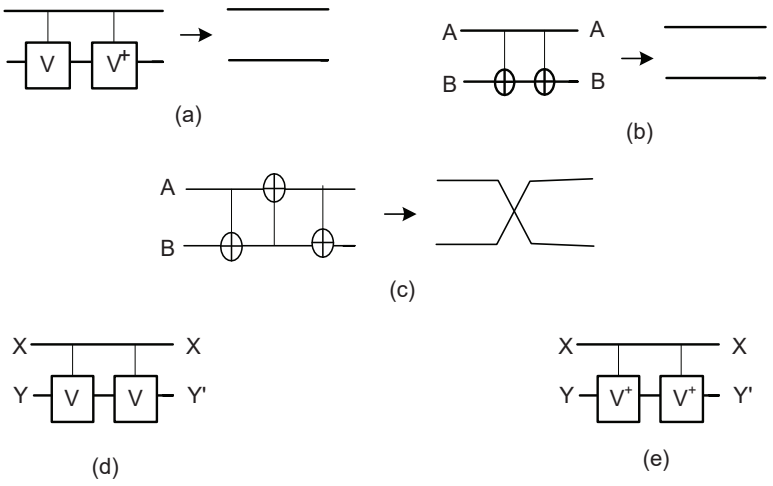


FIG. 2.10: Rules in minimization of quantum circuit.

Minimization of quantum cost is desirable in reversible logic design and for this reason the researchers give importance in reduction of quantum cost as possible. There are several works [9, 61, 62] have been done to simplify the quantum circuit and reduce the cost of the circuit. Fig. 2.10(a)-(c) show some templates to reduce the quantum cost, whereas, Fig. 2.10(d)-(e) show some properties of V and V^+ gate. Section 2.2.5 shows the quantum circuits of some existing gates and the quantum circuits of the proposed gates are shown in the respective proposed

sections (TB gate in Section 3.5.1, HNF gate in Section 4.6.1 and HNF gate in Section 4.6.3).

2.2.4 Delay

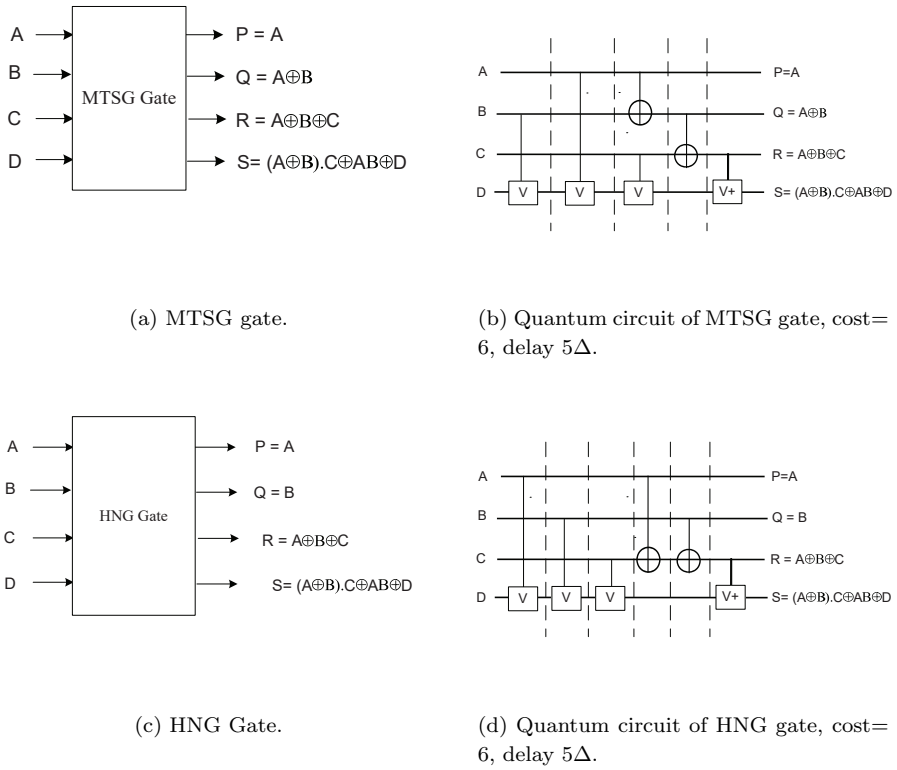
Delay is an important issue in the design of any circuit. In some earlier research [46, 59] the number of the reversible gates (2×2, 3×3 or any order) in the critical path is considered as a unit delay irrespective of their computational complexity. According to this concept, the delay of full-adders of Fig. 2.8(a) Fig. 2.9(a) are same,i.e. 2. However, it is not fair as the number of input and output and the arrangement of the quantum gates in the quantum circuit vary.

Example 12. The block diagrams and quantum circuits of another two full-adders are shown in Fig. 2.11 for delay assessment. Comparison between this Fig. 2.11 with Fig. 2.9 reveals that the construction procedure of a full adder varies not only in terms of number of gate but also on the gate's dimension. Full-adder in Fig. 2.9(a) has two Peres gate, each of which is a three input-output gate, on the other hand, each of the Full-adders in Fig. 2.11(a)& (c) has one reversible gate, which are four input-output gates.

Based on this concept, some researcher considers the logical depth as a measure of the delay [36, 63]. Here, the delay of each 1×1 gate and 2×2 reversible gate is taken as unit delay and denoted as Δ.

Example 13. The full-adder using NG gate(2.8(b)) has delay 18Δ, on the other hand, each of the full-adder using Peres gate(2.9(b)), MTSG gate(2.11 (b) and HNG gate(2.11 (d) has delay 8Δ, 6Δ and 6Δ, respectively.

As any $n \times n$ reversible gate can be designed from 1×1 reversible gates and 2×2 reversible gates, such as CNOT gate, Controlled-V, and Controlled-V^+ gates, the delay of a $n \times n$ reversible gate can be computed by calculating its logical depth when it is designed from smaller 1×1 and 2×2 reversible gates. Still, the quantum cost and delay are not same though they may have same quantitive value.



(a) MTSG gate.

(b) Quantum circuit of MTSG gate, cost= 6, delay 5Δ.

(c) HNG Gate.

(d) Quantum circuit of HNG gate, cost= 6, delay 5Δ.

FIG. 2.11: Delay assessment of reversible gates.

Example 14. The block diagrams and quantum circuits of MTSG gate and HNG gate of Fig. 2.11 for delay shows that the quantum cost of MTSG gate is 6 and the delay is 5Δ. On the other hand, the quantum cost of HNG gate is 6 and the delay is 6Δ.

2.2.5 More about Reversible Gates

Reversible circuits consist of a group of reversible gates. There are several reversible gates in the literature. Some of the reversible gates are described in this section.

2.2.5.1 Feynman Gate (CNOT Gate or Ex-OR Gate)

The Feynman gate (FG) [55] is a 2-input 2-output reversible gate which has the input-output pattern as follows,

$$(A,\ B) \leftrightarrow (P = A,\ Q = A \oplus B)$$

where, A, B denote the input variables and P, Q denote the output variables, respectively. Fig. 2.12(a) and (b) show the block diagram and the quantum circuit of the Feynman gate, respectively. As the quantum circuit of Feynman Gate has only one Ex-OR gate, the quantum cost of FG is 1 and delay is 1 Δ.

Other than as an Ex-OR gate, the Feynman gate copies the input to avoid the fan-out problem in reversible logic (Fig. 2.12(c)). Besides, it generates the complement of an input (Fig. 2.12(d)).

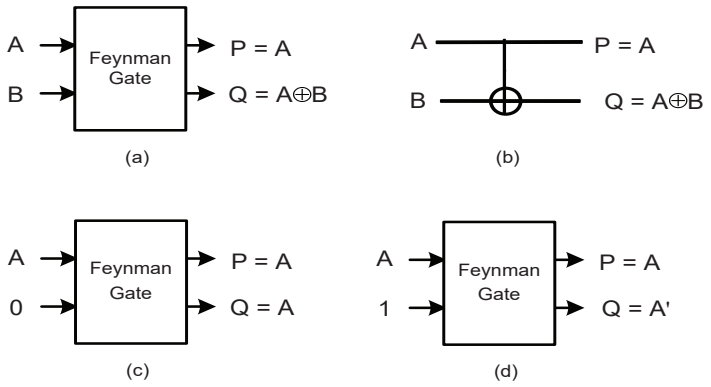


FIG. 2.12: Feynman gate

2.2.5.2 Feynman Double Gate (F2G)

The Feynman double gate (F2G) [55] is a 3-input 3-output reversible gate which has the input-output pattern as follows,

$$(A,\ B,\ C) \leftrightarrow (P = A,\ Q = A \oplus B,\ R = A \oplus C)$$

where, A, B, C denote the input variables and P, Q, R denote the output variables, respectively. Fig. 2.13(a) and (b) shows the block diagram and the quantum circuit of the Feynman double gate, respectively.

F2G is a combination of two FG gates. Like the FG, it also serves the purpose of copying (Fig. 2.13(c)) and generating the complement of the inputs (Fig. 2.13(d)). It has quantum cost 2 and delay 2Δ.

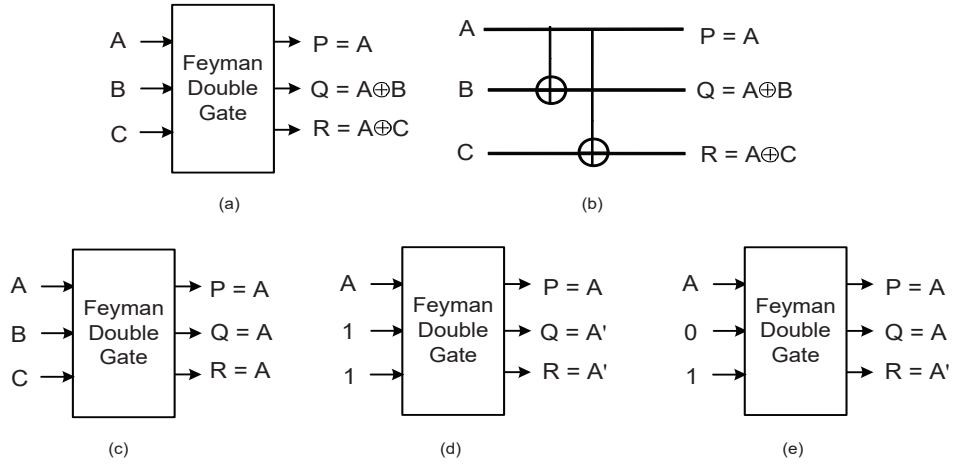


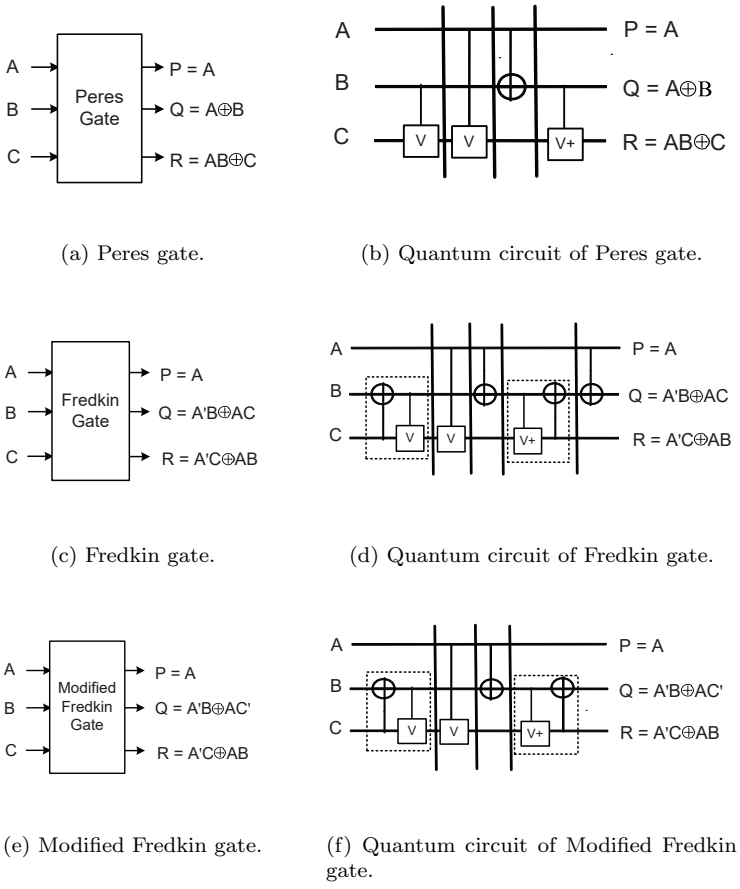
FIG. 2.13: Feynman Double gate

2.2.5.3 Toffoli Gate (TG)

Though the decription of Toffoli gate is ilustrated earlier in Section 2.2.1, the formal definition is given here. A Toffoli Gate (TG) [53] is a 3-input 3-output reversible gate which has the input-output pattern as follows,

$$(A,\, B,\, C) \leftrightarrow (P = A,\, Q = B,\, R = AB \oplus C),$$

A, B, C denote the input variables and P, Q, R denote the output variables, respectively. Fig. 2.3 and Fig. 2.7 show the block diagram and the quantum circuit of the TG, respectively. Quantum circuit of TG needs two Controlled-V gates, one Controlled-V^+ gate and two $Ex-OR$ gates to implement it and all the basic gates are serially placed. As a result, the quantum cost of Toffoli gate is 5 and delay of Toffoli gate is 5Δ.



(a) Peres gate.

(b) Quantum circuit of Peres gate.

(c) Fredkin gate.

(d) Quantum circuit of Fredkin gate.

(e) Modified Fredkin gate.

(f) Quantum circuit of Modified Fredkin gate.

FIG. 2.14: Popular reversible gates.

As the Toffoli gate is a universal reversible gate, it has a great importance in reversible logic synthesis.

2.2.5.4 Peres Gate (PG)

A Peres gate (PG) [56] is a 3-input 3-output reversible gate which has the input-output pattern as follows,

$$(A,\, B,\, C) \leftrightarrow (P = A,\, Q = A \oplus B,\, R = AB \oplus C),$$

where, A, B, C denote the input variables and P, Q, R denote the output variables, respectively. Fig. 2.14(a) shows the Peres gate [56] and Fig. 2.14(b) shows the quantum circuit of the Peres gate (PG).

Peres gate requires two Controlled-V^+ gates, one Controlled-V gate and one $Ex-OR$ gate in its quantum circuit, therefore, the quantum cost of Peres gate is 4. Actually, Peres Gate is the combination of Feynman Gate (FG) and Toffoli Gate (TG) and Peres gate can simultaneously generate two output functions (from Q and R). This gate is popular to construct the full-adder circuits or other circuits which require an Ex-OR operation and the operations done by Toffoli gate, even with the lower cost than that of Toffoli gate.

2.2.5.5 Fredkin Gate (FRG)

A Fredkin gate (FRG) [53] is a 3-input 3-output reversible gate which has the input-output pattern as follows,

$$(A,B,C) \leftrightarrow (P = A,\ Q = A'B \oplus AC,\ R = AB \oplus A'C)$$

where, A, B, C denote the input variables and P, Q, R denote the output variables, respectively.
Fig. 2.14(c) and Fig. 2.14(d) shows the block diagram and the quantum circuit of Fredkin gate, respectively.
Quantum circuit of Fredkin gate consists of two dotted rectangles, one Controlled-V gate, and two Ex-OR gates. Thus, the quantum cost of Fredkin gate is 5. Each basic gates and symmetric patterns of those gate (Fig. 2.6 (e)-(h)) has unit cost and unit delay. The arrangement of the basic quantum gates and their symmetric patterns estimates the delay of Fredkin gate 5Δ.

Fredkin gate also has its importance in reversible literature for generating one output is direct as input and two other outputs as two different Boolean functions. It is also called SWAP gate, as, if the first input is 1, the other two inputs swap in the output. Another use of Fredkin gate is to construct the multiplexer circuits [57].

2.2.5.6 Modified Fredkin Gate (MFRG)

A Modified Fredkin (MFRG) gate [57] is a 3-input 3-output reversible gate which has the input-output pattern as follows,

$$(A,\, B,\, C) \leftrightarrow (P = A,\, Q = A'B \oplus AC',\, R = AB \oplus A'C)$$

where, A, B, C denote the input variables and P, Q, R denote the output variables, respectively. Fig. 2.14(e) and Fig. 2.14(f) are the block diagram and the quantum circuit of Modified Fredkin (MFRG) gate, respectively. It is actually, the modified version of 3-input 3-output Fredkin gate. When A=0, it does the same as Fredkin gate.

Modified Fredkin gate consists of two dotted rectangles, one Controlled-V gate, and one CNOT gates. Hence the quantum cost of Modified Fredkin gate is 4. The arrangement of the quantum gates and their symmetric patterns in the quantum circuit causes the delay of Modified Fredkin (MFRG) gate is 4Δ.

2.3 Physical Implementation Methodologies of Reversible Logic Circuits

Physical implementation of the reversible circuit plays a role to validate the circuit's construction and its working principles. Among the implementation techniques in literature, Charge Recovery Logic (CRL) is based on the explicit reversible pipelined logic gates [64], whereas, Split-level Charge Recovery Logic (SCRL) is based on split-level voltages [65]. Several other implementation techniques are Energy Recovery Logic (RERL) for ultra-low-energy consumption [66, 67] and nMOS Reversible Energy Recovery Logic (nRERL) [68]. The nanoelectronic and optoelectronic implementations of reversible gates are described in [17, 69]. Besides, authors in [57, 70, 71, 72, 73] use a relatively new and extensively used idea which is Transistor Level Realization of reversible circuits.

Authors in [72] describe several approaches for transistor level realization of basic AND and OR operations. One possible way for implementing two transistors AND or OR gate with VDD and GND (Fig. 2.15 (a) and (b)). This paper also describe AND or OR without any VDD and GND (Fig. 2.15 (c) and (d)).

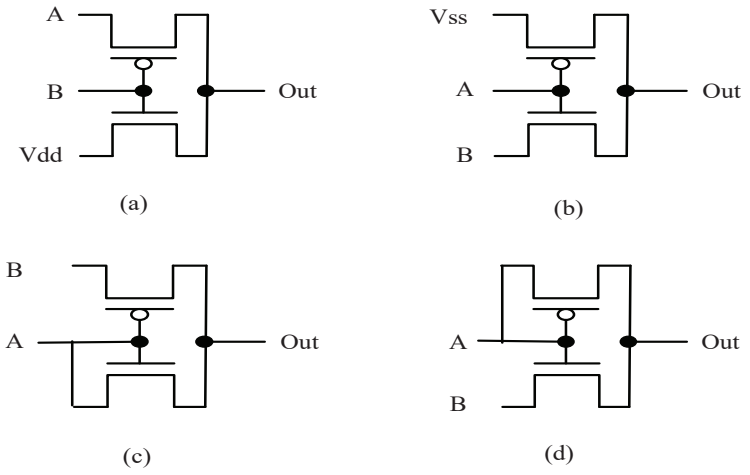
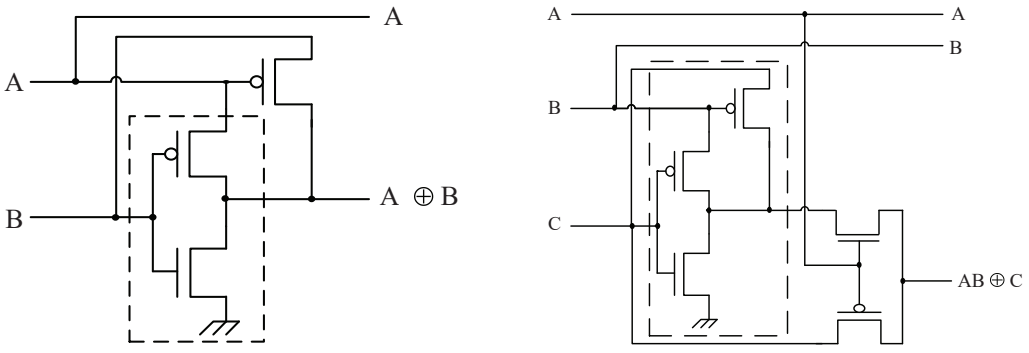


FIG. 2.15: Transistor level realization of AND and OR gate.

Authors in [72] shows the transistor level realization of reversible four-transistor Fredkin gate and six-transistor Toffoli. Later on, authors in [74] presents an improved version of transistor level realization of reversible gates. The transistor-level realization of Feynman and Toffoli gate according to this designs are in Fig. 2.16 (a), and (b). In this thesis, we follow this design procedure for transistor-level realization of the proposed gates given in the Appendix B.

2.4 Overview of Programmable Logic Devices

Reconfigurability of Programmable Logic Devices makes the prominent use of it in designing digital circuits. A logic gate has a fixed function, whereas, PLD has an undefined function. A PLD has to be programmed before to use it. While designing a digital circuit, a question may arise, which one is preferable to use; logic gates or PLD? Each of these devices has its own complexity such as area, power,



(a) Feynman Gate. (b) Toffoli gate.

FIG. 2.16: Transistor level realization of reversible gate [74].

density, and implementation cost. Cosequently, the choice will depend on the particular circuit's requirement. Some application use fixed logic gates while other needs consideration. For example, the design of a simple circuit, which requires few logic gates, may be implemented with fixed logic gate, whereas, a complex circuit, considering the complexity, can be developed by using PLD [75]. According to this author, followings are the characteristics and aptitude of programmable logic:

- A PLD is suited for all designs of different size.
- Prototype designs as well as final applicable designs can use the PLDs.
- The designs that might require modification also can use PLDs.
- It is easy to change the designs made by a PLD without changing the circuit hardware and wiring.
- circuits designed by PLDs can use simple tools like the Boolean logic, the Karnaugh map techniques, and the hardware description languages such as VHDL and Verilog HDL.
- Combinational logic designs, sequential logic designs and memory designs, etc., all suits the PLDs

PLDs have a wide range of application including Glue Logic, State Machines, Synchronization, Decoders, Counters, Bus Interfaces, Parallel-to-Serial conversion, Serial-to-Parallel Conversion, Subsystems and many others. For these extensive varieties of utility, PLD is the right choice for the development and amendment.

2.4.1 Classification of PLDs

There are variety of PLD both in architecture and functionality. Fig. 2.17 shows a pictorial view of the classification of the PLD. Among the different PLDs, PLA and FPGA are the most popular to construct the circuits. This section has a brief description of these two PLDs.

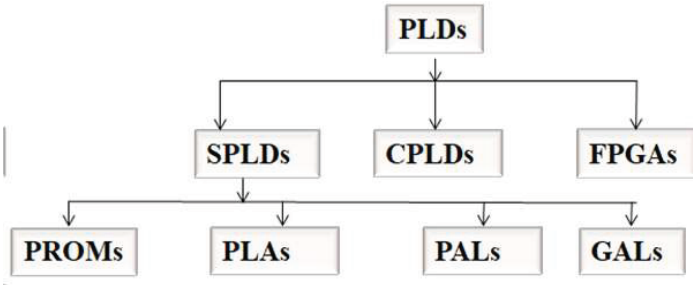
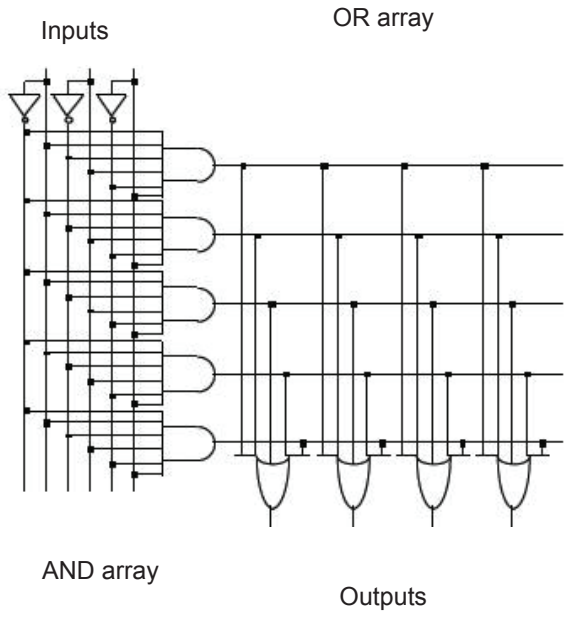


FIG. 2.17: Classification of the Programmable Logic Devices.

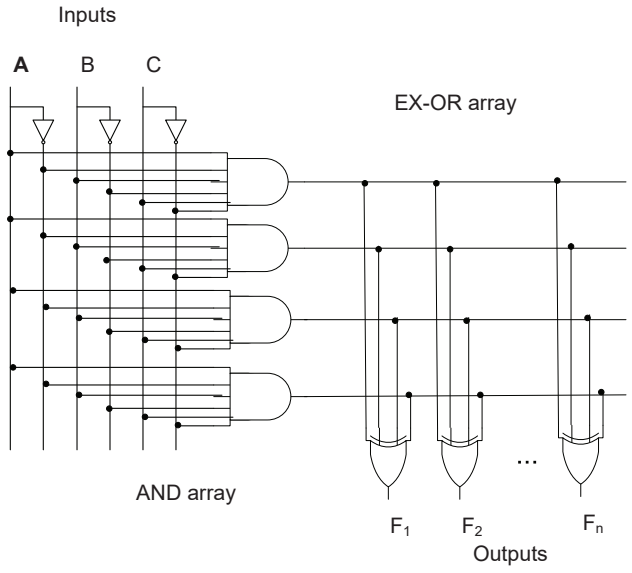
2.4.2 The General Architecture of Programmable Logic Array (PLA)

Architecturally, the PLA has two programmable planes: AND-plane and OR-plane (Fig. 2.18(a)).

The AND-plane consists of programmable interconnect along with AND gates. The OR-plane consists of programmable interconnect along with OR gates. In Fig. 2.18(a), there are three inputs to the PLA and three outputs from the PLA. Each of the inputs connects an AND gate with any of the other input by connecting the crossover point of the vertical and horizontal interconnect lines in the AND gate programmable interconnect. Initially, the crossover points are not electrically



(a) AND-OR array.



(b) AND-Ex-OR array.

FIG. 2.18: The architecture of irreversible Programmable Logic Array.

connected, however, configuring the PLA with connects particular crossover points together.

In this figure, the AND gate is connected with a single line to the input. This view is by convention, still, this also means that any of the inputs (vertical lines) can be connected. When the OR-plane Ex-OR, it is called AND-Ex-OR PLA. Fig. 2.18(b) shows the architecture of AND-Ex-OR PLA.

2.4.3 The General Architecture of FPGA

Architecturally, FPGAs consist of an array of programmable logic blocks and a hierarchy of reconfigurable interconnects that allow the blocks to be wired together [76]. Other than the logic block and interconnection, FPGA has input-output blocks for external communication. This logic blocks implements different combinational and sequential logic functions. Manufacturers configure the logic blocks of an FPGA in such a way that they can provide functionality as that of transistor or as complex as that of a microprocessor. The core component of the logic blocks of an FPGA can be a transistor pairs, combinational gates like basic NAND gates or Ex-OR gates, look-up tables, multiplexers or even a wide fan-in AND-OR structure which are connected to each other to implement desired function. Fig. 2.19 shows the architecture of an FPGA with the magnified view of different parts.

As the logic blocks execute the basic logical and arithmetic calculations, they are the most significant parts of an FPGA. There are several type of Logic Blocks in the literature. The Altra Logic Block [77] evolved from the PLA-based architecture consists of wide fan-in AND gates feeding into an OR gate. This configuration has disadvantage as it use pull-up devices that consume static power. An array full of these pull-ups will consume significant amount of power [76].

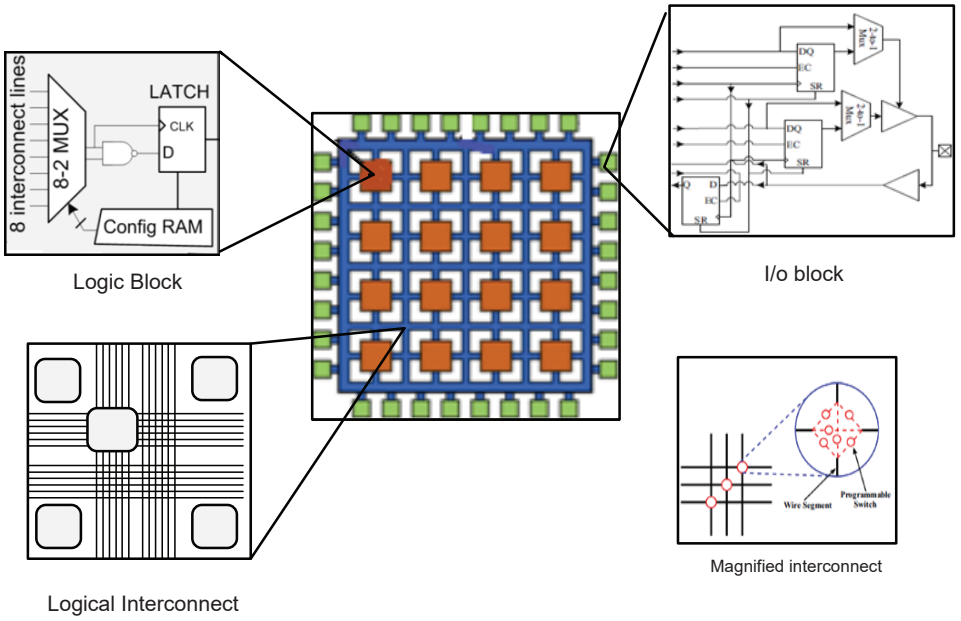


FIG. 2.19: Different parts of an FPGA.

The advantage of a Look-up tables [78, 79, 80] is that they exhibit high functionality. A K-input LUT can implement any function of K inputs and there are 2^{2^K} such functions, however, they are unacceptably large for more than about five inputs, since the number of memory cells needed for a K-input lookup table is 2^K. Though the number of implementable functions increase very fast, they all are not commonly used in logic designs and are also difficult to exploit for a logic synthesis tool. Hence, there is waste of LUTs.

Though the Actel Logic Block [81, 82] and the Quicklogic Logic Block [83] contain multiplexers which also provide large degree of functionality, they achieve at the expense of a large number of inputs.

However, the Plessey Logic Block [84] (Fig. 2.19) based on NAND unit along with other clusters of components (e.g., Random Access Memories, multiplexers and latches) can overcome the former difficulties [78, 79, 80, 84, 85, 86, 87, 88].

In this thesis, we emphasize on the compactness of the physical architecture of Logic Block and Plessey Logic Block is the right choice.

2.5 Summary

This chapter presents a brief study on the basic terminologies on reversible logic. It also includes a short overview on programmable logic devices. Next chapter deals with the design issues of the reversible Programmable Logic Array and then some solutions are proposed to mitigate the deign issues.

Chapter 3

Reversible Programmable Logic Array

3.1 Introduction

In the previous chapter, we gave the descriptions of the basic terminologies regarding reversible logic synthesis and programmable logic devices. In this chapter, we explore an efficient design methodology of the reversible Programmable Logic Array (RPLA).

An elegant solution to the mapping of irregular combinational logic function into a regular structure is provided by Programmable Logic Array. This array logic is based on AND, OR and NOT synthesis to implement SOP or POS, whereas reversible logic prefers Ex-OR operation as well as Exclusive Sum-of-Product (ESOP) synthesis. ESOP synthesis gives out better result than SOP realization where many useful methods are proposed for minimizing multi-output Boolean functions into ESOP form [45, 89]. This chapter has proposed a new approach of designing RPLA for ESOP synthesis.

3.2 Contribution

The key contributions of this Chapter are as follows:

- We propose a low cost 3×3 reversible universal Tara-Babu (TB) Gate.
- Then we develop a heuristic algorithm to sort and realize the product terms.
- We enables the sharing-property of the internal sub-products by using the proposed algorithm to reduce the number of gates, garbage outputs and quantum cost.
- The proposed algorithm also reorders the output functions to make the circuit optimum.
- Finally, we analyze the performance of the proposed algorithm by using MCNC benchmark functions with compared to other existing designs.

3.3 Organization of the Chapter

The organization of rest of the chapter is as follows: in Section 3.4, we explore the existing works in the relevant literature; while Section 3.5.1 proposes the reversible Tara-Babu gate. Then, Section 3.5.2 describes the proposed AND plane, Section 3.5.3 describes the proposed Ex-OR plane and Section 3.6 describes the implementation of the proposed RPLA. We Simulate the proposed circuits and evaluate the performance of the circuits using the proposed algorithm in Section 3.7, whereas, Section 3.8 summarizes the whole chapter.

3.4 Related Works on RPLA

The Boolean function's expression are of three forms, SOP, POS or ESOP form. SOP, POS, and ESOP stand for Sum-of-Products, Products-of-Sum, and Exclusive-Sum-of-Products respectively. A SOP expression is as follows:

$$F_{SOP} = X_1 + X_2 + \cdots + X_n \tag{3.1}$$

Where, X_i is a minterm of one or more literals or product of some minterms, for $1 < i <= n$.
On the other hand, SOP expression is

$$F_{POS} = Y_1 Y_2 \ldots Y_m \tag{3.2}$$

Where Y_j is a maxterm of one or more literals or sum of some maxterms, for $1 < j \leq m$.
ESOP stands for Exclusive-OR-Sum-of-Product, which can be expressed as

$$F_{ESOP} = X_1 \oplus X_2 \oplus \cdots \oplus X_n \tag{3.3}$$

Where X_i is a minterm of one or more literals or product of some minterms, for $1 < i \leq n$ and F is Ex-OR-ed with n terms.

Example 15. Equation 3.4 shows an example of SOP expression, Equation 3.5 is a POS expression, and Equation 3.6 shows an example of ESOP expression .

$$F_1 = AB' + A'BC' + AC' \tag{3.4}$$

$$F_2 = (A' + B).(A + B' + C)(A' + C) \tag{3.5}$$

$$F_3 = A'B' \oplus ABC \oplus C' \tag{3.6}$$

Programmable Logic Array realizes the irregular logic function into a regular structure by using both the SOP and ESOP form. However, among all the forms even in, as shown in the example 15, the ESOP is compact and hence cost effective [90, 91]. We can get the ESOP form from the SOP form by using EXORCISM4 [92]. Authors in [90] generate the minimal ESOP and use to synthesize the reversible logic. They wants to take advantage of shared minterms to minimize the garbage outputs. However, in this way, several other minterms cause more garbage outputs. Factorization of ESOP may benefit from this synthesis approach but has penalties regarding the number of gates and garbage outputs. The minimal form of ESOP is also used in [44]. They use a different technique. This technique focuses on rearranging the products in ascending order according to the decimal value of the product terms. Then, an algorithm is provided in which each step consider a configuration of the Toffoli gate. Though this method overcomes the penalties of the method in [90], it suffers from time complexity described in [46]. However, the design in [46] still follows the conventional architecture (both complement and non-complement lines for copying input variables) of Programmable Logic Array.

Authors in [47] replace the complement and non-complement lines by one line. This design uses Fredkin gates in AND plane of RPLA for AND operations and F2G in Ex-OR plane of RPLA for Ex-OR operations. Though eliminating of one line makes this design enhance, using these two gates elevates the number of garbage and quantum cost. Parallel research on RPLA in [48] also attempt to design the RPLA which minimize the cost, however, maximize the garbage output than [47].

Analyzing all these, this chapter proposes an efficient RPLA for ESOP synthesis which reduces the cost metrics.

3.5 Proposed RPLA

This section describes the proposed design of Reversible Programmable Logic Array (RPLA). On the way to design the RPLA, this dissertation assumes that the functions are already been minimized and available in ESOP form. Still then, different output functions of multi-output ESOP may have some common products and sub-products. Sharing these products or sub-products in RPLA is advantageous. For this reason, we propose a reversible gate namely Tara-Babu (TB) gate in Section 3.5.1. This gate enables the product-sharing property. Section 3.5.2 describes the proposed method to generate the AND-plane, whereas, Section 3.5.3 focuses on the generation of the Ex-OR-plane. Finally, Section 3.6 describes the implementation of the proposed RPLA with these AND-plane and Ex-OR-plane.

3.5.1 Proposed Reversible Tara-Babu (TB) Gate

This section presents the proposed gate namely Tara-Babu (TB) which we use for the AND operation in AND Plane. Let, A,B,and C are the inputs and P,Q,and R are the outputs of TB gate, then the relationship between inputs and outputs are

$$(P, Q, R) \leftrightarrow (A \oplus B, BC' \oplus AC, B'C \oplus AC').$$

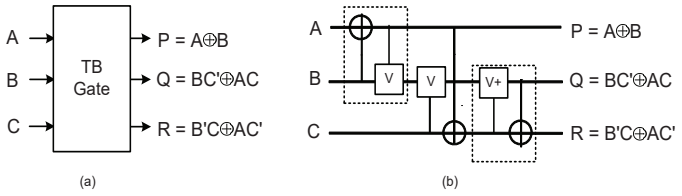


FIG. 3.1: Proposed reversible TB gate : (a)block diagram (b) quantum circuit.

Fig. 3.1 (a) and Fig. 3.1 (b) shows the block diagram and the quantum circuit of TB gate, respectively and Table 3.1 proves the reversibility of the proposed gate. Quantum cost of the proposed TB gate is 4 and the delay of this gate is 4Δ which is the least in the literature for generating two AND terms simultaneously.

TABLE 3.1: Truth table of the proposed reversible TB gate

Input			Output		
A	B	C	$P = A \oplus B$	$Q = BC' \oplus AC$	$R = B'C \oplus AC'$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	1	0
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	1	0	1	0

The proposed reversible TB gate is a universal gate as it can implement all basic boolean functions (AND, OR, NOT Ex-OR) by using one TB gate and compound functions (NAND and NOR) by using two cascaded TB gates. Fig. 3.2 (a) shows the utilization of TB gate as AND and OR gates, Fig. 3.2(b) shows the implementation of NOT gate. Afterward, Fig. 3.2(c) presents the Ex-OR function and Fig. 3.2(d) depicts the copy property to solve the fan-out problem. Finally, Fig. 3.2(e) and Fig. 3.2(f) show the NAND function and the NOR function by cascading two TB gates, respectively. Besides, Fig. 3.3 shows the AND terms generated by the TB gate.

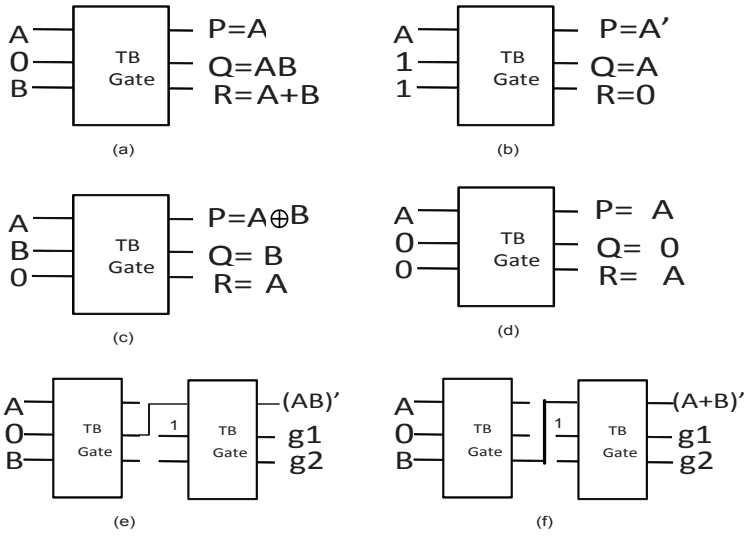


FIG. 3.2: Implementation of all boolean functions using the proposed reversible TB gate.

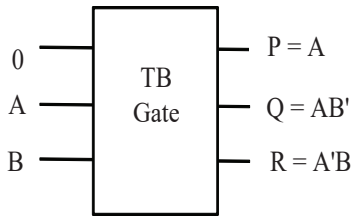


FIG. 3.3: Different AND terms produced simultaneously by the proposed TB gate.

3.5.2 Proposed AND-plane of RPLA

In the AND plane, the combination of the F2G and TB gates generate all forms of the product of two variables and hence eliminate the necessity of dedicated line to complement the input literals. These two gates also propagate the input variables or its complement. (Fig. 3.4(a)-(h)) The elimination of complement line

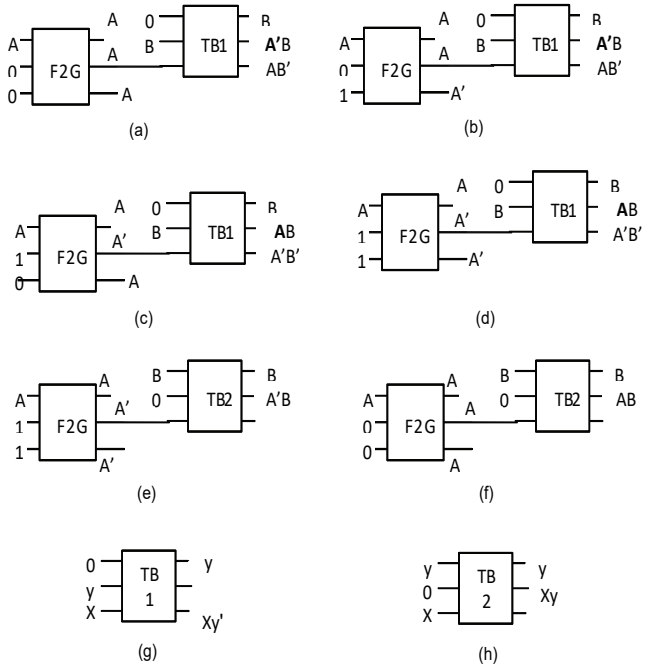


FIG. 3.4: Different combinations of the F2G and TB gates.

and properties of used gates in AND plane reduce the number of gates. Besides, the simultaneous realization of two product terms of first two variables(Fig. 3.4(a)-(d)) enables the sharing property of sub- products, which ensure the further reduction of gates. For the product terms of more than two variables, one TB gate is added

whenever a sigle literal is appended. In this situation, two cases may arise, the product terms may have any of the two forms Xy' or Xy, where, 'X' represent the previous generated terms and 'y' represents the latest appended variable. The configuration of TB gate in Fig. 3.4(e) is used for the former one and Fig. 3.4(f) is used for the later one. The minimization technique for the number of gates also influence the AND plane to design with less garbage and low quantum cost.

Example 16 explores the construction of the RPLA of a multi-output function based on the proposed algorithm (Algorithm 1). In this algorithm, line 3-7 describe the sorting procedure based on the products' frequency and their sharing affinity property, whereas, line 9-23 describe the procedure of product realization. Afterwads, line 18, 19 ensure the sharing phenomena by the proposed gate and the use of DOT indicates no gate is required in that percular case.

Definition 1. The Frequency of a product term is the number of output functions that share the product term.

Definition 2. The cross point in RPLA, in which no gate is used, is termed as **DOT**.

Example 16. Consider the following ESOP functions:

$$\begin{aligned} F_1 &= AB'C \oplus A'B \\ F_2 &= A'B'C \oplus AC \\ F_3 &= AB'C \oplus BC' \oplus A'B \\ F_4 &= AC \\ F_5 &= BC' \oplus A'B \oplus AC \end{aligned} \tag{3.7}$$

In Equ. 3.7, the output functions F_1, F_3 & F_5 share $A'B$, hence, the frequency of $A'B$ is 3, whereas, the frequency of $A'B'C$ is one, as it is only in F_2. Table 3.2 shows the frequency of each product term of the functions in Equ. 3.7. Now following the Algorithm 1 we construct the proposed AND plane of RPLA for multi-output ESOP functions(shown in Fig. 3.5).

Algorithm 1: The proposed algorithm to construct the AND-plane of RPLA

Input : Several Benchmark Functions
Output: Number of gates used and number of garbage outputs produced to realize AND plane of RPLA

begin
  $i = input$, $o = output$
  Sort the products in ascending form based on their frequency of a benchmark function, then their index.
  If Product X and Y have the same frequency and index(y)= index(x) +1, realize product Y prior to the product X if all the conditions hold:
  i. Both X and Y are the part of same output function Op.
  ii. X is the first product of Op and Y is the second product of Op and not shared by any other output function Oq, where p < q < = O, O denotes the number of output functions.
  If Inj is not the first input variable and used in product X, then produce product X in later.
  Set $DOT := 0$,$G_{tc} := 0$, $G_{bc} := 0$; where G_{tc}=no.of total gates and G_{bc}=no.of total garbage outputs; Put all literals into the stack
  while *for each ordered product (P_i)* **do**
    if *Ini is the first variable of Product(P_i)* **then**
      if *In_i or In_i' is used only once in total AND plane* **then**
        DOT + + ; Update Stack;
      else if *In_i or In_i' is used only twice* **then**
        Apply FG gate; G_{tc}++;
      else
        Apply F2G gate; G_{tc}++;
    else if *In_i or In_i' is the second variable of Product (P_i)* **then**
      if *current sub-product is in the form $In_{1'}.In_2$ or $In_1.In_{2'}$ and complementary sub-product is already produced* **then**
        DOT + + ; Update Stack;
      else
        Apply TB Gate; G_{tc}++; Update Stack;
    else
      Apply TB Gate; G_{tc}++; G_{bc}++; Update Stack;
  end while
end

3.5.3 Proposed Ex-OR-plane of RPLA

AND plane generates the product terms of each functions, then Ex-OR plane performs the Ex-OR operations among the corresponding terms of and individual

TABLE 3.2: Frequency table for the ESOP functions in Equ. 3.7

Product	Frequency
$A'B'C$	1
$A'B$	3
$AB'C$	2
AC	3
BC'	2

output function. Suitable choice of the order of the output functions play a significant role in construction of a minimize circuit. Elimination of the fan-out problem is also considered as multiple use of a particular terms may occur. Keeping in mind, these two consideration, we develop a heuristic approach in Algorithm 2 to construct the Ex-OR plane. In this Algorithm, Line 4-5 set a specific condition for ordering. Line 10-11 indicate, if there is a product term which is a first term of a function and is used for single time (frequency 1), the DOT is used indicating no gate is required for this term (There are four DOTs in Fig. 3.6). Line 12, 13 indicate if the first product is shared by other functions (Frequency more than 1), a FG is required (Upper most FG for F_1), whereas, if F_i is the last function that shares any previously used product term, a FG is required as well as a garbage bit is counted (BC' in F_5, right bottom FG in Fig. 3.6). Line 13, 16, 20 returns the number of FG gates and Line 17 returns the number of garbage outputs. Fig. 3.6 shows the constructed Ex-OR plane for Equ. 3.7.

3.6 Implementation of the proposed RPLA

By using Algorithms 1 and 2, the realization of the proposed RPLA is shown in Fig. 3.7. In Algorithm 2, Feynman gate is used to mitigate the fanout problem. According to the proposed algorithms our RPLA requires 14 gates and it produces 6 garbage outputs for the ESOP given in Equ. 3.7.

As the proposed RPLA is constructed by using only the reverible gate, is maintaintains the revesibility of itself. The Fig. 3.7 ensures that the whole circuit is reversible by showing that it preserve the equal number of input and output. In this case, the number of input is 12 and the number of output is also 12.
The Theorem 3.6.1- Theorem 3.6.4 describe the properties of the proposed generalized RPLA.

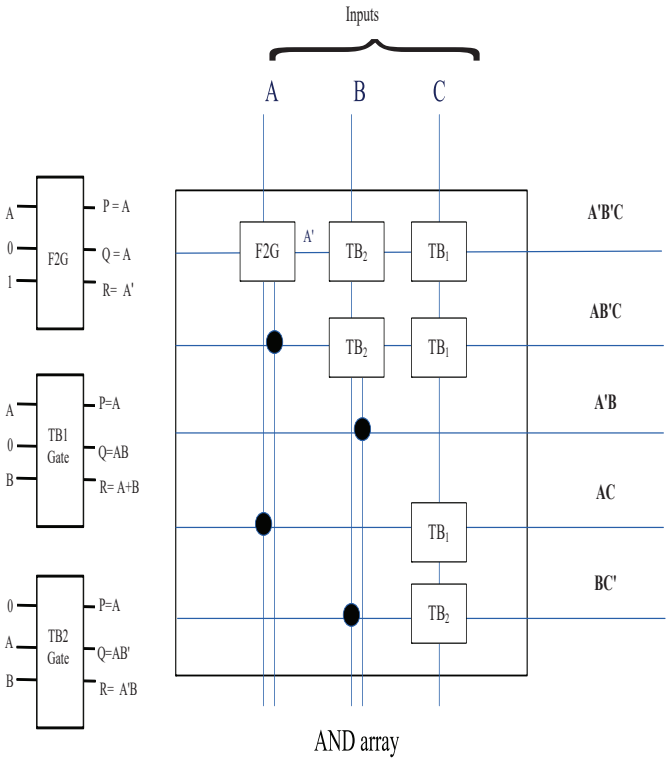


FIG. 3.5: The proposed AND-plane of RPLA for multi-output ESOP functions given in Equ. 3.7.

Algorithm 2: The proposed algorithm to construct the Ex-OR plane of RPLA

Input : Several Benchmark Functions
Output: Number of gate used and number of garbage produced to realize Ex-OR plane of RPLA circuit.

```
1  begin
2      i = input, o = output
3      For each output Functions F_i
4      if F_i has only one product then
5          generate F_i at last
6      Loop
7      For each product P_j of F_i
8      Loop
9      if P_j is the first product in F_i then
10         if Freq(P_j) = 1 then
11         Place DOT; else
12         Place Gate FG;
13         G_t c++;
14     else if  F_i is the last function that shares P_j then
15         Place Gate FG;
16         G_t c++;
17         G_b c++;
18     else
19     Place Gate FG;
20     G_t c++;
21 end
```

Theorem 3.6.1. *The AND plane of RPLA requires no more than* ***q*** *TB gates. Where,* ***q*** *denotes the total number of 2-input AND operations among the literals of the distinct product-terms.*

Proof. Let, there are **q** number of 2-input AND operations for **p** products of a ESOP function. TB gate generates three different 2-input AND operations. Two of them are simultaneously generated. When, we use these two AND terms in the AND plane, AND plane of RPLA requires less number of TB gates. Otherwise, AND plane of RPLA requires **q** number of TB gates.

Therefore, the maximum number of TB gates to construct the AND plane of RPLA is **q**. (Proved) □

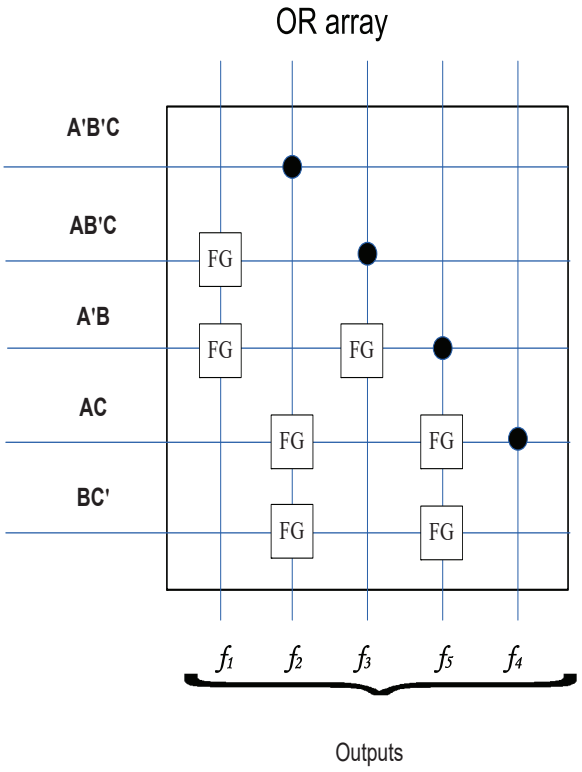


FIG. 3.6: The Proposed Ex-OR plane of RPLA for multi-output ESOP functions given in Equ. 3.7.

Example 17. Fig. 3.5 shows an RPLA which has total **q**=7 AND operations. The AND-plane requires only 6 TB gates by sharing property, which is less than **q**. If there will no scope of sharing, the AND-plane requires 7 TB gate, which is the maximum number of TB gate used and that is equal **q** for this example.

Theorem 3.6.2. *The AND plane of RPLA produces no more than* ***i+q-DOT*** *garbage outputs. Where,* ***q*** *denotes the total number of 2-input AND operations among the literals of the distinct product-terms,* ***i*** *denotes the number of input variables of RPLA, and* ***DOT*** *denotes the number of cross-points in the AND plane of RPLA.*

Proof. The TB gate is a 3 × 3 gate, and each gate generates maximum 2 garbage outputs. One is responsible for propagating the input-variable, another one is an

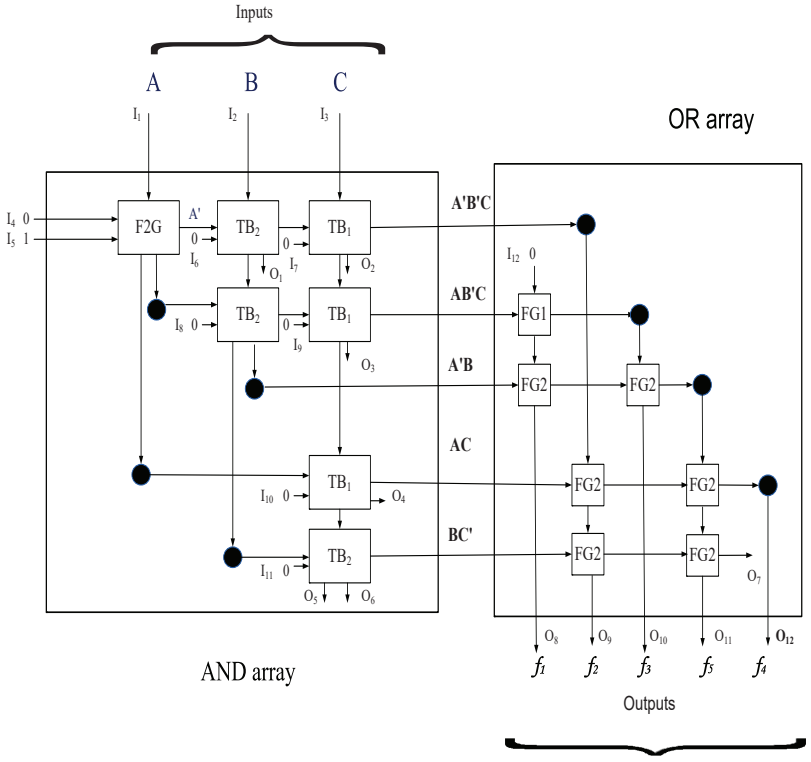


FIG. 3.7: The proposed reversible Programmble Logic Array for Equ. 3.7

unused AND term. If the propagating input variables are no longer used (after last use of input-variables), then they increase the number of garbage outputs by $i - DOT_{in-var}$, where, DOT_{in-var} denotes the DOTs for used propagating the input-variables.

Each unused AND terms of TB gate increase the number of garbage outputs by $q - DOT_{AND}$, where, DOT_{AND} denotes the DOTs for used AND terms.

Sharing the AND terms as well as used propagated input-variables (denoted by

DOTs) decreases the number of garbage outputs.
Hence, the total number of garbage outputs $= i - DOT_{in-var} + q - DOT_{AND}$.
$= i + q - DOT$ Therefore, the AND plane of RPLA produces no more than $i + q - DOT$ garbage outputs. (Proved) □

Example 18. The products of ESOP function given in Equ. 3.7 has total 7 AND operations, and 3 inputs. By adopting the sharing-Property, in the proposed architecture of RPLA, the number of DOTs reaches at 4 and the number of garbage outputs at 6. Thus, the theorem holds for this example.

Theorem 3.6.3. *The Ex-OR plane of RPLA requires no more than $n - DOT$ gates. Where, n denotes the total number of product terms, and **DOT** denotes the number of cross-points in the OR plane of RPLA.*

Proof. Ex-OR plane uses Feynman gates for Ex-OR operations. This reversible gate has two inputs and two outputs. If any of the products are not used further, then a DOT is placed otherwise a gate is used. For example, if there is only two product-terms are Ex-ORed in Ex-OR plane, Ex-OR plane requires only one Feynman gate, and there is one DOT. In this way, the number of Feynman gates in Ex-OR plane of RPLA can be reduced by DOT. As Feynman gates are used for both propagation and Ex-OR operations, the total number of Feynman gates in the Ex-OR plane of RPLA is **n-DOT**.
Therefore, the Ex-OR plane of RPLA requires no more than $n - DOT$ gates. (Proved) □

Example 19. For multi-output function F in Equ. 3.7, the total number of product-terms is 11, and the number of DOT is 4 in Fig. 3.6. Hence, the number of Feynman gates is $n - DOT = 11 - 4 = 7$, which holds the Theorem 3.6.3.

Theorem 3.6.4. *The Ex-OR plane of RPLA produces no more than* ***p-DOT*** *garbage outputs. Where,* ***p*** *denotes the number of distinct products, and DOT be the number of total cross-points in the Ex-OR plane.*

Proof. Ex-OR plane uses Feynman gates for Ex-OR operations. This reversible gate has two inputs and two outputs. Here, one output is the horizontally propagated product-term, and another one is the Ex-ORed of the two consecutive product-terms vertically. If any of the distinct products are not used further, then a DOT is placed otherwise a gate is used. In this way, the number of garbage outputs of Feynman gates in Ex-OR plane of RPLA can be reduced by DOT. Moreover, the Ex-ORed term is propagated vertically to the next Feynman gate to be EX-ORed with another product-term, and finally, we get the output of a particular function. Therefore, the total number of possible garbage outputs in the Ex-OR plane of RPLA is **p-DOT**. (Proved) □

Example 20. Consider Fig. 3.6 for multi-output function F in Equ. 3.7. In Fig. 3.6, the number of distinct products (p) is 5, the number of cross-points (DOT) is 4, and the number of garbage outputs is 1 ($p - DOT = 5 - 4 = 1$), which holds the Theorem 3.6.4.

3.7 Simulation and Performance Evaluation

In this section, the proposed reversible TB gate and the proposed RPLA for a given ESOP function are simulated and the performance is analyzed to discover the efficiency of the proposed design of RPLA.

3.7.1 Simulation Environment

Our proposed algorithms for realization and minimization of RPLA have been written using language C and have tested extensively on windows microcomputer. The simulations of the proposed circuits are done using DSCH 3.5 [93] software on a computer, which has the Intel(R) Core(TM) i7 CPU with 2.50 GHz Clock Speed

and 8.00 GB RAM. During the execution, it was ensured that no other application is running.

3.7.2 Simulation Results

Fig. 3.8 shows the simulation result of TB gate. This waveform reveals that the TB gate works correctly as the waveform reflexes the input-output combinations properly. Moreover, Fig. 3.9 represents the implemented RPLA (for ESOP functions of Equ. 3.7) in DSCH [93]. The simulation result of this proposed RPLA (Fig. 3.10) exposes the functional responses of the Equ. 3.7 .

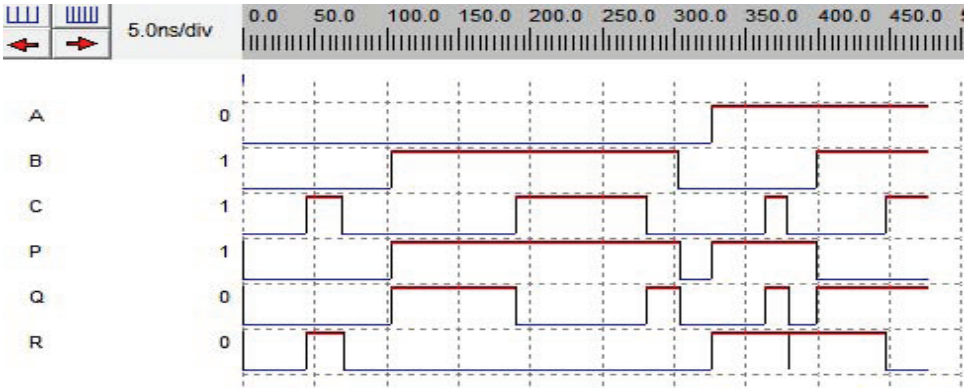


FIG. 3.8: Simulation of the proposed TB gate.

3.7.3 Performance Metrics

We consider five criteria as the metrics of the performance analysis of RPLA. Brief description of these metrics are given below.

- Gates: Reversible gates are used to design a specific circuit. Accumulating all the gates and counting the total number of gates will be found. This is a performance metric of reversible circuits. Less number of required gate certifies the high performance of the circuit.

- Garbage outputs: Garbage output is measured by counting the unused output of a circuit. By accumulating all the garbage outputs, we can find the

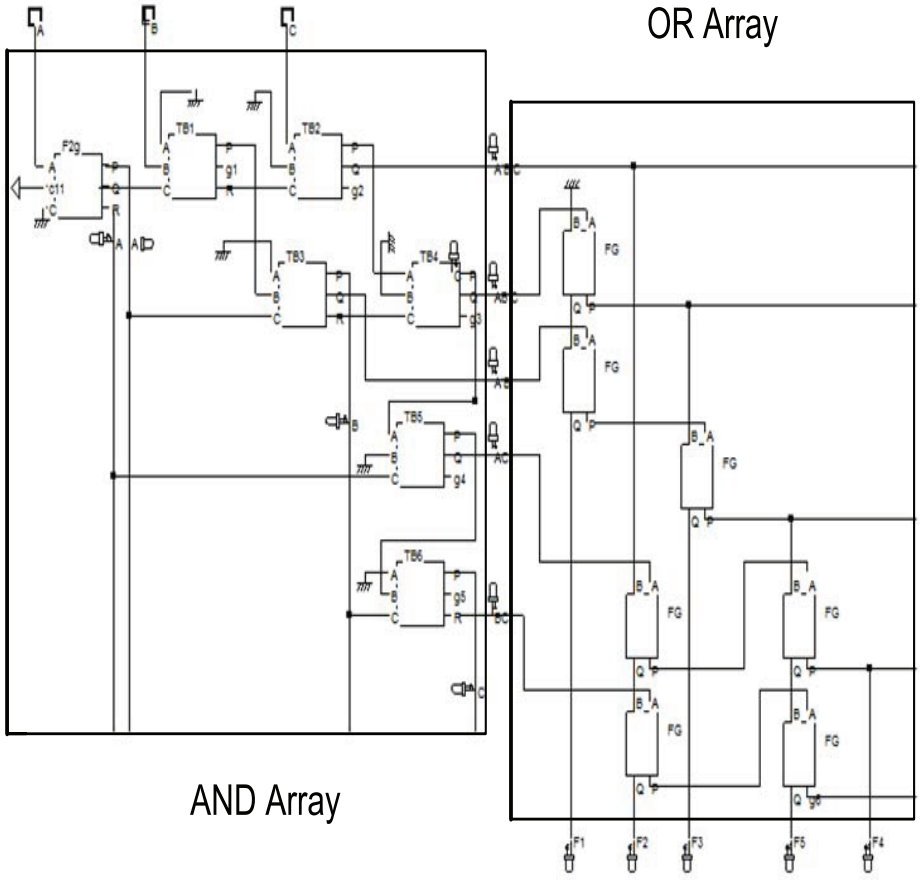


FIG. 3.9: Architecture of the proposed RPLA for ESOP functions given in Equ. 3.7 at physical-lavel in DSCH 3.5 [93].

total number of garbage outputs. The minimum value of garbage outputs proves the efficiency of the design.

- Quantum Cost: It is one of the primary performance metrics of the reversible circuit. The quantum cost of a reversible gate is the cumulative number of NOT, Controlled-V, Controlled-V^+, and $CNOT$ gates required in its implementation.

- Area: The area of a logic circuit means the total area accumulated by the individual circuit element. Microwind [94] can be used to calculate the area of each individual element. As an example, if a circuit consists of n gates

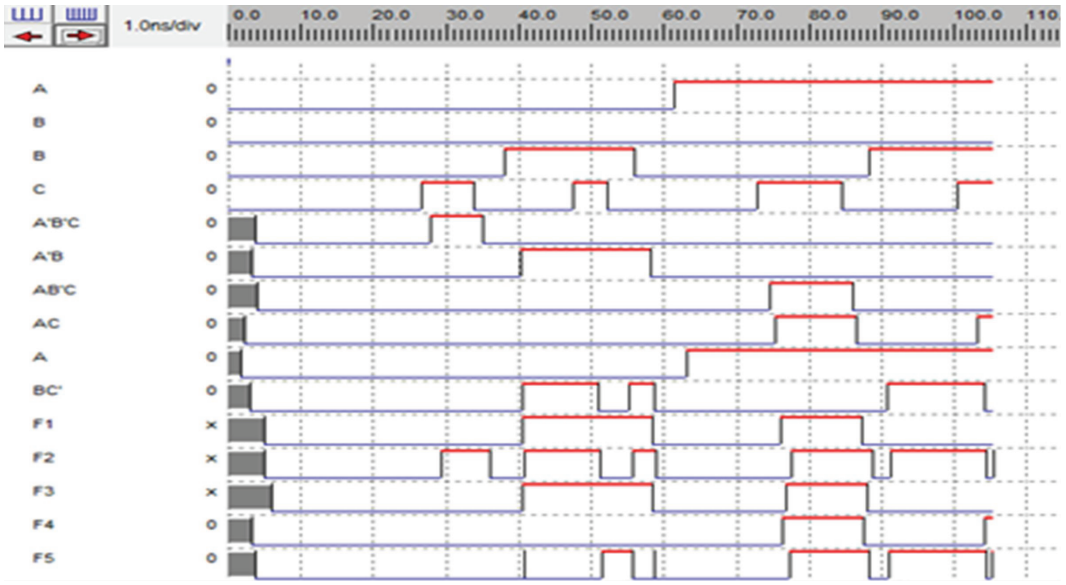


FIG. 3.10: Simulation of the proposed RPLA for ESOP functions (Equ. 3.7).

with area A_1, A_2,..., A_n, then the area (A) of that circuit is as follows-

$$A = \sum_{i=1}^{n} A_i \tag{3.8}$$

Using Microwind [94], we can find the area of a 4:1 multiplexer (which has three Modified Fredkin gate). The area of a Modified Fredkin gate 0.00152 mm^2 and hence, the total area of 4:1 multiplexer is 0.00456 mm^2.

- Power: The power consumption of a logic circuit means the total power consumption by the individual circuit elements. Power of the reversible circuit can be calculated using DSCH 3.5 [93] and Microwind 3.5 [94]. The transistors or pass transistors are used to design the corresponding circuit in DSCH. Then power can be estimated by using Microwind.

3.7.4 Performance Analysis

Table 3.3 shows the experimental results comparing our proposed method with the methods presented in [46, 47] and [48] in terms of the number of gates, the number of garbage outputs and the quantum cost for ESOP functions given in Equ. 3.7. Table 3.4-Table 3.6 show the comparison among the various methods to

TABLE 3.3: Comparison of the proposed and the existing RPLA for the ESOP functions in Equ. 3.7

Method	Gate	Garbage	Quantum Cost
Existing [46]	19	11	47
Existing [47]	18	7	39
Existing [48]	17	19	55
Proposed	14	6	31

realize the RPLA for ESOP expression of some particular benchmark functions. The tables reveals that our method has effective results for these particular functions. Each benchmark function has a different format in respect to the number of inputs, number of outputs, number of product terms and pattern of product terms and there is no straight-forward algorithm to minimize the performance metrics, however, the optimum result from our proposed algorithm outcomes due to the four reasons:

1. The first reason is the elimination of complementary lines, which by default eliminates the use of NOT gate

2. Second, generation of sharable two sub-products

3. Sharing property also eliminate the use of the gate used for copy any literals (as the first input of the shared term need not be copied).

4. Finally, ordering the output functions also has an impact on the reduction of the performances metrics.

For a more specific example, the benchmark function sao2 has 10 inputs, 4 outputs, 28 product terms, and there are 202 AND operations. All the reasons mentioned above directly have an impact on the realization sao2 function.

TABLE 3.4: Comparison of the proposed and the existing RPLAs by using MCNC Benchmark functions in terms of the total number of gate.

Function Name	Existing [46]	Existing [47]	Existing [48]	Proposed
5xp1	170	140	166	136
9sym	439	402	427	396
adr3	69	58	67	58
b12	170	147	159	144
bw	350	296	305	296
duke2	931	914	941	902
z5xp1	171	152	167	148
sao2	291	268	384	161
rd53	55	48	56	46
rd84	321	296	328	286
xor5	9	4	8	4

In case of gate reduction (Table 3.4), in the proposed method, sao2 function needs 10 F2G for the copy operation, 197 TB gates for AND operation, while 5 (five) other AND operations eliminated by sharing sub-products, and finally, it needs 54 FG gates for EXOR operation. On the other hand, the method in [48] requires 24 FG gate for copy, 202 MUX gate for AND operation and 58 FG gate for EXOR operation. Similarly, the method in [46] requires 291 gates and [47] requires 268 gates. The use of F2G, TB gate in AND plane and the product generation procedure, output ordering all together outcome in reduction of the gate in the proposed method.

It is evident that the proposed method does not assure the gate reduction for every benchmark function, for example, for adr3, bw and xor5 the proposed method and the method in [47] gives the same result, and in some cases the improvements are marginal, for example, duke2 has an improvement of only 3.11%, 1.31% and 4.14%

TABLE 3.5: Comparison of the proposed and the existing RPLAs by using MCNC Benchmark functions in terms of the total number of garbage outputs).

Function Name	Existing [46]	Existing [47]	Existing [48]	Proposed
5xp1	80	157	112	102
9sym	330	411	385	354
adr3	50	61	48	40
b12	90	151	132	115
bw	46	293	64	55
duke2	579	923	667	635
z5xp1	85	153	114	97
sao2	243	274	236	115
rd53	23	50	42	33
rd84	269	187	265	115
xor5	4	8	8	4

with respect to [46, 47], and [48]. On the other hand, in some cases the proposed method shows a significant improvement, such as for 5xp1, the proposed method has an improvement of 20% and 18.07%, for sao2 the improvement is 10.31% and 32.03%, and for xor5, the improvement is 50.00%, and 55.56% compared to [48] and [46], respectively.

In case of garbage output reduction(Table 3.5), though the proposed method does not outperform compared to [46] in all cases, our method produces 20% and 57.25% less garbage output for adr3 and rd84, respectively. Then, the proposed method reduces the garbage output up to 56% with an average 19.75% improvement to [48] and up to 50% with an average 32.75% improvement to [47].
Then, Table 3.6 indicates a significant reduction of quantum cost to all existing designs [46, 47, 48]. The proposed method reduces the quantum cost up to 55% with an average 21.22% improvement to [46], up to 50% with an average 14.74% improvement to [47], while, in some cases, the proposed method deteriorates to [47].

In a nutshell, analyzing all the particular benchmark function, we can conclude

TABLE 3.6: Comparison of the proposed and the existing RPLAs by using MCNC Benchmark functions in terms of quantum cost).

Function Name	Existing [46]	Existing [47]	Existing [48]	Proposed
5xp1	508	468	418	402
9sym	1737	1456	1405	1377
adr3	189	176	157	155
b12	562	490	453	546
bw	499	686	446	448
duke2	3361	3024	2735	2704
z5xp1	519	476	425	411
sao2	1099	940	890	862
rd53	162	148	134	126
rd84	1132	992	928	893
xor5	9	8	8	4

TABLE 3.7: Area requirement and power consumption of the different MCNC benchmark circuits by using the proposed method

Name of the benchmark circuits	Area (μm^2)	Power (mW)
5xp1	692.64	96.2
9sym	2427.84	337.2
adr3	266.4	37
b12	764.64	106.2
bw	712.8	99
duke2	4733.28	657.4
z5xp1	721.44	100.2
sao2	1524.96	211.8
rd53	210.24	29.2
rd84	1546.56	214.8
xor5	5.76	0.8

that the proposed algorithm works well to reduce the number of cost metrics for ESOP functions if there is an opportunity of sharing the initial sub-products.

Area needed by the different MCNC benchmark circuits designed by the proposed gates are guaranteed by using Equation 3.8. The power consumption by the circuits are generated using Microwind 3.5 [94] which are shown in Table 3.7.

Based on the information of Table 3.4, Table 3.5 and Table 3.6, some graphical

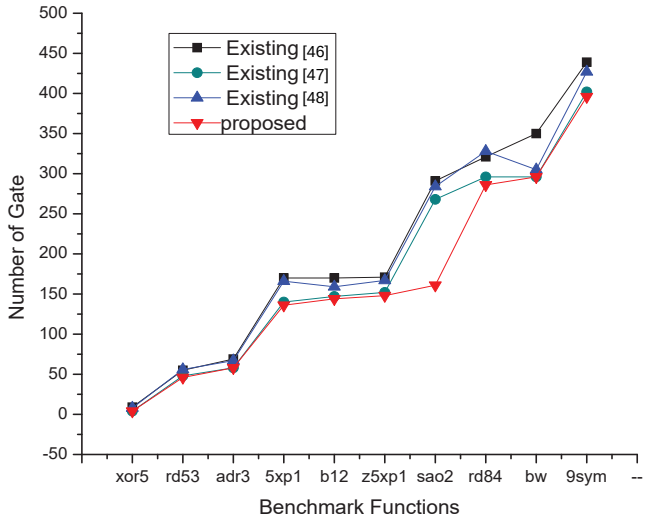


FIG. 3.11: Graphical representations of Benchmark functions vs Number of gates.

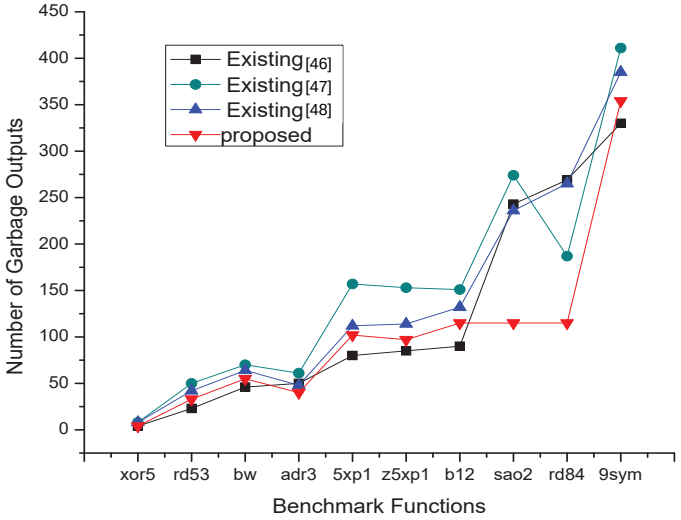


FIG. 3.12: Graphical representations of Benchmark functions vs Number of Garbage outputs.

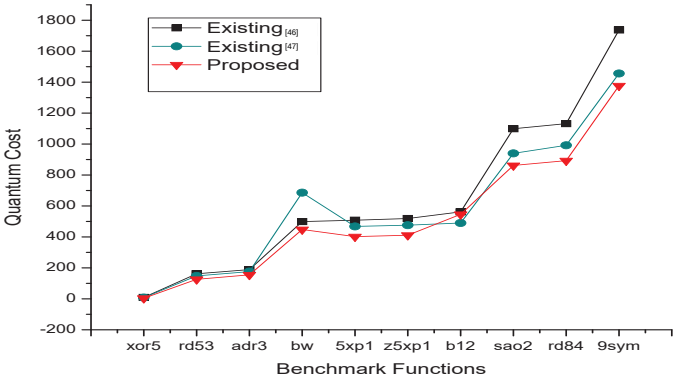


FIG. 3.13: Graphical representations of Benchmark functions vs Quantum cost.

interpretations are drawn in Fig. 3.11, Fig. 3.11,and Fig. 3.13.

These graphs represent the value of performance metrics of the corresponding benchmark functions, where, red triangles on red lines indicate the values of the proposed method. It is clear that, most of the cases, the number of gate, garbage output and the quantum cost are lower than the corresponding metrics of the existing designs. More specifically, for sao2 function, the proposed method has 862 quantum cost, whereas, the method in [48], [47] and [46] have quantum cost 890, 940, and 1099, respectively, which implies the improvement in terms of quantum cost are 3.24%, 9.05%, 27.49% with respect to [48], [47] and [46]. Though the improvement (for different function and for different performance metrics) varies, we can observe the improvement from the graphs and calculate the value from the performance tables.

3.8 Summary

This chapter focuses on the detailed architecture of the proposed RPLA as well as existing design procedures of the RPLA. We discuss the proposed design with its complexities elaborately here. This chapter also provides the simulation of

the proposed designs and performance analysis of the proposed design compared with the other existing designs. The next chapter provides the design issues of reversible Field Programmable Gate Array and we develop some solutions those can mitigate the challenges.

Chapter 4

Reversible Field Programmable Gate Array

4.1 Introduction

In the previous chapter, we focus on the design procedure of the reversible Programmable Logic Array. In this chapter, we emphasis on the design of the reversible Field-Programmable Gate Array.

Field Programmable Gate Arrays (FPGAs) are the preeminent alternative to the Application Specific Integrated Circuit (ASIC) and Mask Programmable Gate Arrays (MPGAs) by providing re-programmability, higher speed, higher density, zero recurring engineering costs and massive parallel operations. Architecturally, FPGA is an array of programmable logic blocks and interconnections. These logic blocks are used to implement the sequential and complex combinational functions. As the blocks can be reconfigured, they influence the speed and density of an FPGA [85]. Look-Up Table (LUT) and Plessey are the most popular SRAM-based logic blocks. The former one has the exponential growth of complexity with respect to inputs. However, the later one overcomes these difficulties through clusters of components e.g., NAND units, Random Access Memories, multiplexers and latches

[76, 84]. Overwhelmed opportunities of FPGAs and energy saving characteristics of reversible logic have caught the eyes of the researchers to design the FPGAs in power recurring ways leads the design of the reversible FPGAs [50, 51, 52]. All of them have designed the FPGA's core part 'Plessey Logic Block' in reversible way by using different number, different type of gates as well as they have used different algorithms to construct the components of the Logic Block. This chapter has proposed an efficient reversible Plessey Logic Block for FPGA.

4.2 Contribution

The key contributions of this chapter are summarized as follows:

- We design each individual component of Plessey logic block of FPGA such as reversible D-Latch, reversible Decoder, reversible Multiplexer, reversible Master-Slave Flip-Flop, and reversible RAM separately.
- We optimize the proposed components in terms of the number of gates, garbage outputs, quantum cost, and delay.
- We also optimize the proposed components in terms of the area and power to ensure the power efficiency.
- We proposed Two 4×4 reversible gates, namely HNF (Hafiz-Naz-Flip-Flop) gate and HND (Hafiz-Naz-Decoder) gate, are proposed to achieve the optimization goal.
- We develope algorithms, lemmas, and theorems to certify the novelty and scalability of the proposed design.
- Finally, the proposed reversible Plessey (4×2) Logic Block of FPGA is compared with the existing designs. The comparative result proves the efficacy and novelty of the proposed design.

4.3 Organization of the Chapter

The rest of the chapter is organized as follows: Section 4.4 explores the existing works in the relevant literature; while Section 4.6.1, 4.6.2, and 4.6.3 propose different components of the reversible Field Programmable Gate Array such as reversible D-Latch, reversible Master-Slave Flip-Flop, and reversible Decoder, respectively. Then, Section 4.6.4 describes the construction procedure and functionality of proposed reversible Random Access Memory. We propose the reversible Multiplexer and NAND unit in Section 4.6.5 and Section 4.6.6, whereas, we assemble the components of reversible Field Programmable Gate Array and describe its properties in Section 4.7. Section 4.8 depicts the simulation and performance evaluations, and finally, Section 4.9 summarizes the whole chapter.

4.4 Related Works on Reversible FPGA

Reversible Plessey Logic Block of FPGA is first attempted in [50]. In this design, authors used reversible Feynman gate (FG), Toffoli gates (TG), BSP gate, and NH gate. The quantum cost of the used gate, such as reversible BSP gate, is not analyzed even. Besides, the quantum circuit of NH gate shows that the quantum cost of NH gate is 6. Therefore, the quantum circuit of NH gate is wrongly designed or the claim of having 4 quantum cost is incorrect.

Later on, an improved version of Reversible FPGA is proposed in [51]. In this design, MUX gate, Fredkin gate and Peres gates along with the Feynman gate and Toffoli gates are used. Quantum cost of these gates is 4, 5, 4, 1 and 5, respectively. Though this design is an enhanced method of the former one, the authors did not compare the each component of FPGA comprehensively, not even with the previous design presented in [50]. Only the design of n:1 multiplexer and D-Latch are compared with the counterparts. The authors of [51] compared their design

with only one paper. Therefore, the novelty of this design is not significant.

Afterward, reversible fault-tolerant FPGA is designed in [52]. However, providing fault-tolerant design for every component of a large circuit usually is not an option. It may increase the chip-size, power consumption, and cost. Besides, it will take more time to design, verify, and test. As a consequence, in this chapter, a non-fault-tolerant strategy is widely investigated and implemented to make the architecture of Plessey Logic Block (each component of the whole circuit) efficient regarding every performance metrics.

4.5 Design Methodology

This section presents the design methodology of the reversible circuits. There are several methods in the literature, i.e., direct transformation method, truth table extension method, template matching. Direct transformation approaches are easy to adopt. However, it requires a large number of gates and garbage outputs [54]. Truth table extension methods are not applicable for the large circuit with numerous input-output combinations [36]. Besides, template matching will work well by the availability of the templates [36]. Analyzing all these, in this dissertation, the template method is applied to design individual components of the Plessey Logic Block.

4.6 Proposed Components of the Reversible FPGA

Fig. 2.19 shows the block diagram of the FPGA, where the logic block is the Plessey Logic Block. This logic block consists of four different components: An 8:2 Multiplexer, a RAM, a D-Latch and a NAND unit. In this dissertation, all of these components are designed with reversible logic.

4.6.1 Proposed Reversible D-Latch

Architecturally, a D-Latch is a small memory unit which transfer 'Data' input to the output. The clock signal of D-Latch enables it to make possible the data entry into the latch.

In the recent designs of D-Latch [54, 57, 95], each D-Latch has 6 quantum cost and produce two garbage outputs irrespective their design methodology. We survey and investigate that the garbage output of D-Latch cannot further be optimized, only the quantum cost can further be reduced to 5 by using proposed method.

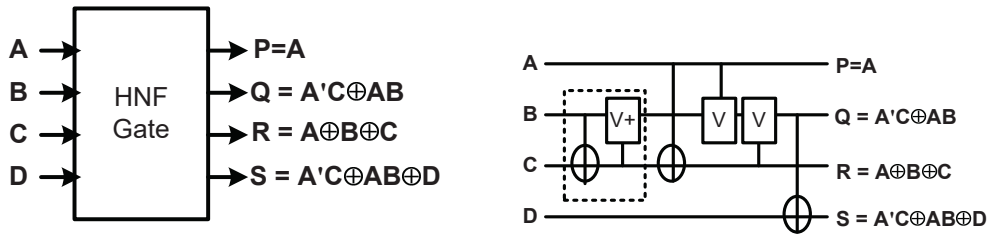
Let, D is the data-input, clk is the clock signal, Q is the output of a D-latch. Then, the characteristic function of the D-Latch (Table 4.1) is $Q^+ = D \cdot clk + clk' \cdot Q$. If the clock (clk) is 1, the data-input D is transfered at the output that is $Q^+ = D$, otherwise, $clk = 0$, and the latch maintains its previous state, i.e, $Q^+ = Q$.

TABLE 4.1: The characteristic table of the D-Latch

Clock (clk)	Data (D)	Previous State Q	Current State Q^+
0	0	0	0
0	1	0	0
0	0	1	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

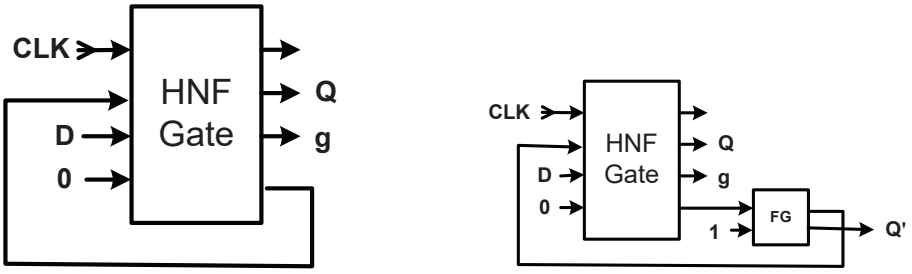
We use the proposed reversible HNF (Hasan-Naz-Flip-Flop) gate to map the characteristic function of the D-Latch. Fig. 4.1(a) and Fig. 4.1(b) show the block diagram and the quantum circuit of the proposed HNF gate, respectively. Table 4.2 proves the reversibility of the proposed HNF gate. The propagation delay of HNF gate is 5Δ and quantum cost of this gate is 5 (Fig. 4.1(b)).

Fig. 4.2(a) shows the Proposed design of the reversible D-Latch using the HNF gate. By putting A = CLK, C= D (Data-input), and D= 0, the proposed HNF



(a) Block diagram (b) Quantum circuit

FIG. 4.1: The proposed reversible HNF gate.



(a) D-Latch with only output Q (b) D-Latch with outputs Q and Q'

FIG. 4.2: The proposed design of D-Latch.

gate works ad the D-Latch, which produce two copies of Q output (second and fourth output of HNF gate), one of these outputs acts as the feedback of the latch and other is the Q^+ output (current status). This circuit requires no other gate to avoid the fan-out. Moreover, this design of D-Latch has only one HNF gate. Thus, the propagation delay of the proposed D-Latch is 5Δ, and it has five quantum cost.

The design in Fig. 4.2(a) does not produce the complement output Q' which we often require in sequential circuits [54]. For this reason, we propose a novel D-Latch (that has both the outputs Q and Q') is with the one HNF gate and the one Feynman gate (Fig. 4.2(b)). In this design, the Feynman gate generates the complement of the output Q.

TABLE 4.2: Truth table of the proposed reversible HNF (Hasan-Naz-Flip-Flop) gate

Input				Output			
A	B	C	D	$P = A$	$Q = A'C \oplus AB$	$R = A \oplus B \oplus C$	$S = A'C \oplus AB \oplus D$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	0	1	0
0	1	0	1	0	0	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	0	1	0
1	0	0	1	1	0	1	1
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	1
1	1	0	0	1	1	0	1
1	1	0	1	1	1	0	0
1	1	1	0	1	1	1	1
1	1	1	1	1	1	1	0

Lemma 1. *A reversible D-Latch with the simultaneous Q and Q' requires 2 gates, 2 garbage outputs, 6 quantum cost, and* 6Δ *delay, respectively, where* Δ *denotes the unit delay.* □

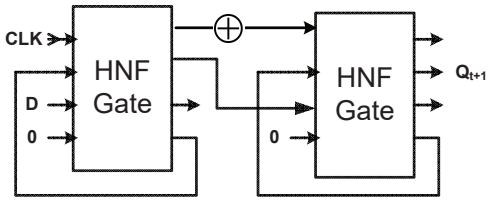
Example 21. The proposed design of D-Latch in Fig. 4.2(b) (with simultaneous Q and Q') has two gates, one is HNF gate and another is FG gate. HNF has 5 quantum cost and Feynman gate has one quantum cost. Therefore, the proposed D-latch has total 6 quantum cost. Additionally, delay of HNF is 5Δ and delay of FG is 1Δ. Serial arrangement of these two gates causes delay in total 6Δ.

The characteristics table of D-Latch (Table 4.1) tells us that, for the four inputs combinations $(0, 0, 0)$, $(0, 1, 0)$, $(1, 0, 0)$, and $(1, 0, 1)$, the output Q^+ is 0, and the other four inputs combinations $(0, 0, 0)$, $(0, 1, 0)$, $(1, 0, 0)$, and $(1, 0, 1)$, the output Q^+ is 1. To maintain the reversibility, there are at least $log_2 4 = 2$ additional outputs (garbage outputs). Between the two Q^+ outputs of the D-Latch (one is

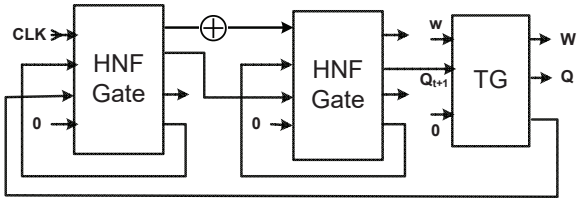
from HNF gate, other is from Feynman gate), one acts as feedback (Fig. 4.2(b)). Hence, the proposed D-Latch produces no additional garbage output.

4.6.2 Proposed Reversible Master-Slave Flip-Flop

Fig. 4.3(a) shows the proposed design of Master-Slave D-Flip-Flop and Fig. 4.3(b) shows the proposed design of write-enable Master-Slave D-Flip-Flop. The methodology of using two latches, one as Master and other as Slave, has been used to design the Master-Slave reversible Flip-Flops. Additionally, a Toffoli gate is used to design write-enable Master-Slave D-Latch.



(a) Master-Slave D-Flip-Flop



(b) Write-enable Master-Slave D-Flip-Flop

FIG. 4.3: The proposed design of reversible Master-Slave D-Flip-Flops

Lemma 2. *A reversible Master-Slave Flip-Flop can be designed with 3 gates and 3 garbage outputs. It also has 11 quantum cost, and* 11Δ *delay, where* Δ *denotes the unit delay.* ■

Example 22. The proposed design of Master-Slave Flip-Flop (Fig. 4.3(a)) has three gates: two HNF gates and one NOT gate. HNF has 5 quantum cost and

NOT gate has one quantum cost, as a result, the proposed Master-Slave Flip-Flop has total 11 quantum cost. Additionally, delay of HNF is 5Δ and delay of NOT is 1Δ. These two gates are arranged in serial and hence, their delay is in total 11Δ. The arrangement of these three gates causes three garbage outputs.

Lemma 3. *A reversible write-enable Master-Slave Flip-Flop can be designed with 4 gates, 3 garbage outputs, 16 quantum cost and* 16Δ *delay, respectively, where* Δ *denotes the unit delay.* ■

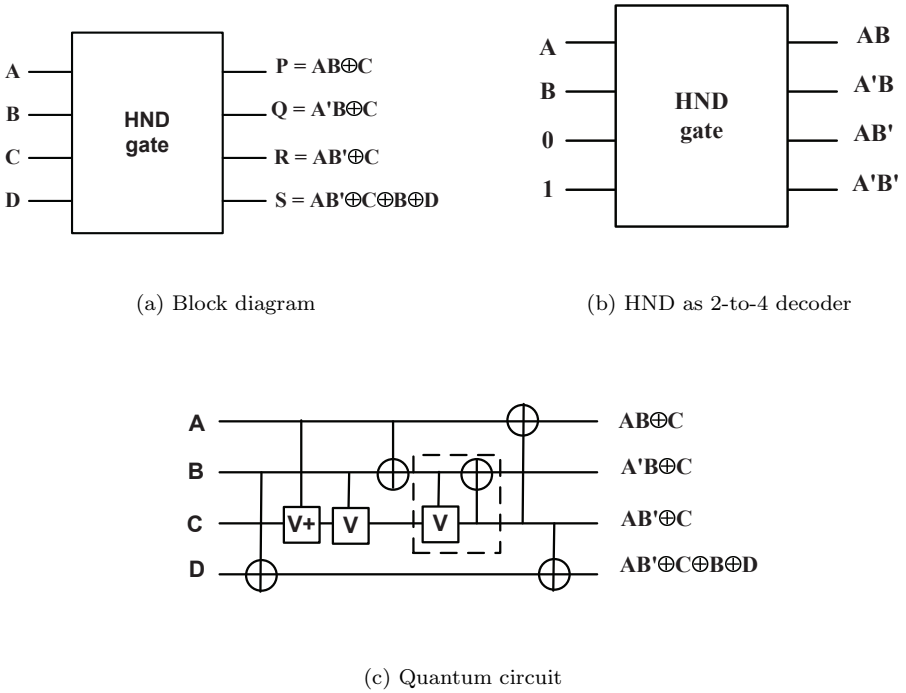
Example 23. From the Fig. 4.3(a), it can be viewed that proposed design of write-enable Master-Slave Flip-Flop has four gates, two HNF gates, one Toffoli gate and one NOT gate. HNF has 5 quantum cost, Toffoli gate also has 5 quantum cost and NOT gate has one quantum cost, as a result, the proposed write-enable Master-Slave Flip-Flop has total 16 quantum cost. Additionally, delay of HNF is 5Δ, delay of Toffoli is 5Δ and delay of NOT is 1Δ. These three gates are arranged in serial, and hence, their delay is in total 11Δ. It can also be viewed that the arrangement of these three gates causes three garbage outputs.

4.6.3 Proposed Reversible Decoder

Architecturally a Decoder is a logic circuit which has n input lines for n bits and 2^n output lines. Only one n-bit combination remains activate at a time.

There exists several designs of the reversible decoder in the literature [51, 52, 96, 97, 98]. Though the design in [96] undergoes from less delay, the requirement of the number of gates is high and also produces a large number of garbage outputs and overheads massive quantum cost. The other designs [51, 97, 98] also have this type of paradigm. To solve this problem and design the decoder circuit in the efficient way, a new 4×4 reversible gate HND (Hafiz-Naz-Decoder) is proposed which requires 7 quantum cost and causes 7 unit in delay measurement. Table 4.3 proves the reversible property of the proposed HND gate. The block diagram of the proposed reversible HND gate and its quantum circuit are shown in Fig. 4.4(a)

and Fig. 4.4(c), respectively.



(a) Block diagram

(b) HND as 2-to-4 decoder

(c) Quantum circuit

FIG. 4.4: The proposed reversible HND gate and its application

A 2-to-4 decoder must generate four logical AND functions $A'B'$, $A'B$, AB', AB. Setting $C = 0$ and $D = 1$, the proposed reversible 2-to-4 decoder gate (HND gate) can generate all four necessary AND functions textcolorblackwithout generating any garbage outputs which are shown in Fig. 4.4(b). Thus, the proposed reversible 2-to-4 decoder has one gate and 0 (Zero) garbage output.

The proposed HND gate is constructed by using two quantum V gates, one quantum V^+ gate and five quantum CNOT gates. The arrangement of these gates results in the proposed reversible 2-to-4 decoder cost 7 and produces delay 7Δ which is least in the literature.

For the higher order decoders such as 3-to-8 and above we use the HND and Fredkin gates. The architecture of the proposed 3-to-8 reversible decoder is in

TABLE 4.3: Truth table of the proposed reversible HND (Hasan-Naz-Decoder) gate

Input				Output			
A	B	C	D	$P = AB \oplus C$	$R = A'B \oplus C$	$Q = AB' \oplus C$	$S = AB' \oplus B \oplus C \oplus D$
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	1	1	1	1
0	0	1	1	1	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	0	0	1	1
1	0	0	1	0	0	1	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	1
1	1	0	1	1	0	0	0
1	1	1	0	0	1	1	0
1	1	1	1	0	1	1	1

Fig. 4.5(a). An algorithm (Algorithm 3) is derived to illustrated the design procedure of the proposed n-to-2^n reversible decoder. In line 5 of the algorithm, the first two control bits (S_0, S_1) are assigned to the HND gate. Line 8 returns the outputs for 2-to-4 decoder. Then in line 10, higher order input is assigned to the Fredkin gates through a recursive call to the previous reversible decoder for $n \geqslant 2$ and lines 17,18 returns the outputs. The complexity of this algorithm is $O(n)$, where n is the number of data bits and $n \geqslant 2$.

According to Algorithm 3, we design the proposed n-to-2^n reversible decoder (Fig. 4.5(b)). Theorem 4.6.1-Theorem 4.6.4 describe the properties of the proposed reversible decoder.

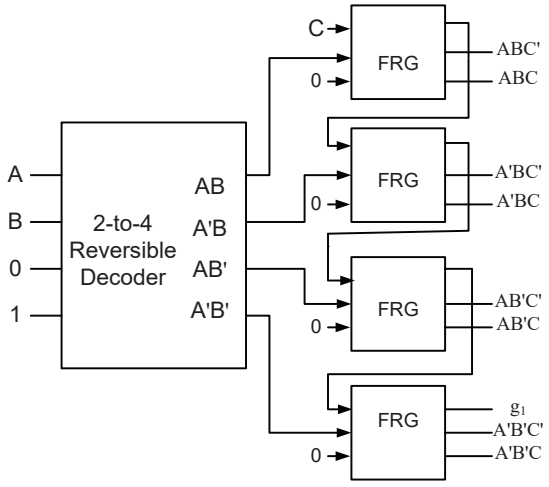
Algorithm 3: The proposed algorithm to construct the reversible n-to-2^n decoder RD(n)

Data: Data input set $S(S_0, S_1, ..., S_{n-1})$

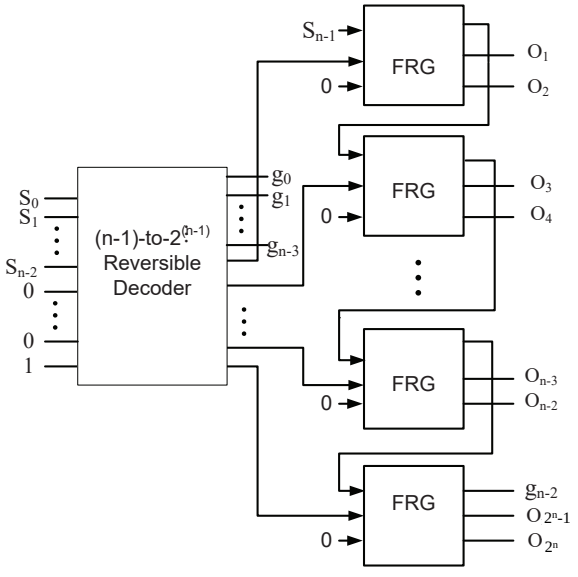
Hafiz-Naz Decoder(HND) gate and Fredkin gate(FRG)

Result: n-to-2^n reversible decoder circuit ($n \geqslant 2$) with gate count RD(n)

```
1  begin initialization;
2  Let, i = input, o = output , Gcount = No. of gate
3  Assign Gcount = 0;
4  for j ← n to 2 do
5      if j = 2 then
6          S_{j-2} → i_1.HND, S_{j-1} → i_2. HND, 0→ i_3. HND, 1→ i_4.HND;
7          Gcount = 1 ;
8          return Gcount, HND.o_1, HND.o_2, HND.o_3 and HND.o_4 → desired
           output ;
9      else
10         S_j → i_1. FRG ;
11         0 → i_3. FRG;
12         call RD(j-1)
13         RD.o_{j-1} → i_2.FRG_j ;
14         Gcount+ = 2^j ;
15     end if
16 end for
17 return FRG_j.o_3 and FRG_j.o_2;
18 return Gcount ;
19 end
```



(a) 3-to-8 decoder



(b) n-to-2^n decoder

FIG. 4.5: The designs of the proposed reversible decoders

Theorem 4.6.1. *An n-to-2^n reversible decoder requires at least $2^n - 3$ reversible gates, where, n denotes the number of bits and $n \geqslant 2$.*

Proof. Fig. 4.15 shows that a 2-to-4 decoder uses only one HND gate.
Thus, the number of gate =1 $(2^2 - 3)$ and the theorem holds for n=2.

Then, Fig. 4.16 shows a 3-to-8 decoder has one HND gate and 4 Fredkin gates, also holds the theorem $((2^3 - 3 = 5))$.
Basically, the Fredkin gate generates two different terms by appending (either the normal form or the complement form of) a new variable with its input term.
Hence, for n-to-2^n reversible decoder, $2^n/2 = 2^{n-1}$ Fredkin gates are added to $n-1$-to-2^{n-1} reversible decoder.
Summing up all the number of gate we get,
$1 + 4 + 8 + \cdots + 2^{n-1}$
$= 1 + 2 + 4 + 8 + \cdots + 2^{n-1}$-2
$=2^n - 1 - 2$
$=2^n - 3$
Therefore, an n-to-2^n reversible decoder requires at least $2^n - 3$ reversible gates. (Proved)

□

Theorem 4.6.2. *An n-to-2^n reversible decoder produces at least n - 2 garbage outputs, where n denotes the number of bits and $n \geqslant 2$.*

Proof. When n is 2, a 2-to-4 reversible decoder is the HND gate (Algorithm 3). The 2-to-4 decoder circuit (HND gate) has 4 primary outputs (O_0, O_1, O_2, O_3) but 2 inputs (S_0, S_1). Therefore, the 2-to-4 reversible decoder should have at least 2 constant inputs to preserve the reversibility.
Now, in the proposed 2-to-4 reversible decoder, there are 2 constant inputs and 2 primary inputs, i.e., the total inputs are 4 and all 4 outputs are primary outputs. Thus, there is no garbage output, which holds Theorem 4.6.2 for $n = 2$.

For the higher order decoder, with the increasing number of bits by one, only one garbage output is added each time.
Thus, the claim in Theorem 4.6.2 satisfies for all n.
Therefore, an n-to-2^n reversible decoder produces at least n - 2 garbage outputs. (Proved)

□

Theorem 4.6.3. *An n-to-2^n reversible decoder has at least $5(2^n-4)+7$ quantum cost, where, n denotes the number of bits and $n \geqslant 2$.*

Proof. The proposed reversible HND gate acts as a 2-to-4 decoder which requires 7 quantum cost i.e, $5(2^2-4)+7=7$.
Thus, the theorem holds for 2-to-4 decoder, where $n=2$.
We know, an n-to-2^n reversible decoder requires at least 2^n-3 reversible gates.(4.6.1)
Among the gates one is HND gate and other (2^n-4) are Fredkin gate. Each Fredkin gate has 5 quantum cost.
Hence, The total quantum cost of an n-to-2^n reversible decoder $= 5(2^n-4)+7$.
Therefore, an $n-to-2^n$ reversible decoder requires at least $5(2^n-4)+7$ quantum cost. (Proved) □

Theorem 4.6.4. *An n-to-2^n reversible decoder has $(7(2^n-4)+7)\Delta$ delay, where, n denotes the number of bits and $n \geqslant 2$, Δ denotes the unit-delay.*

Proof. The proposed reversible HND gate acts as a 2-to-4 decoder which has 7Δ delay (Fig.4.9 (c)) which is equivalent to $5(2^2-4)+7$.
Thus, the theorem holds for 2-to-4 decoder, where $n=2$.
We know, an n-to-2^n reversible decoder requires at least 2^n-3 reversible gates(Theorem 4.6.1).
Among the gates one is HND gate and other (2^n-4) are Fredkin gate. Each Fredkin gate has 5Δ delay.
The HND gate and all the Fredkin gate of the proposed $n-to-2^n$ reversible decoder are in serial (4.5), i.e, the critical path must pass through each of these gates; as a result, the total delay of an n-to-2^n reversible decoder $= 5(2^n-4)+7$.

Therefore, an $n-to-2^n$ reversible decoder requires at least $5(2^n-4)+7$ quantum cost. (Proved) □

4.6.4 Proposed Reversible Random Access Memory

This section presents the construction procedure and the complexities of the proposed reversible Random Access Memory.

The Random Access Memory (RAM), architecturally, is a two-dimensional array of individual memory units. More specifically, there are 2^n rows and m columns, where each row contains m write-enable Master-Slave Flip-Flops. A decoder is needed to address these rows. Section 4.6.3 describes the proposed decoder, whereas, Section 4.6.2 describes the proposed the individual memory unit (the write-enable Master-Slave Flip-Flops). The proposed $2^n \times m$ RRAM also needs Toffoli gates and Feynman gates. The proposed architecture of reversible RAM follows the reversible RAM in [99] as a base. The working principle of the proposed reversible RAM is as follows:

In the $2^n \times m$ RRAM, n is the number of control bits of the proposed decoder (configured HND gate) and 2^n decoded-outputs are the row selection bits. For a particular combination of n input variables of the configured HND gate, only one out of 2^n outputs is *active* at a time. This active output is propagated through the Toffoli gate and selects only one row of the array of memory units. The write bit W specifies whether there is a read or a write operation. When W= 1 or High, then the m data-inputs D_0 to D_{m-1} are written in the m Flip-Flops of the selected row.

When W= 0 or Low, then the m outputs Q_0 to Q_{m-1} are the previously stored bits in the Flip-Flops, and they are read from the m Flip-Flops of the selected row.

The usage of Feynman gates eliminates the fan-out problem. These gates also perform the Ex-OR operations of the outputs of the Flip-Flops in a column.

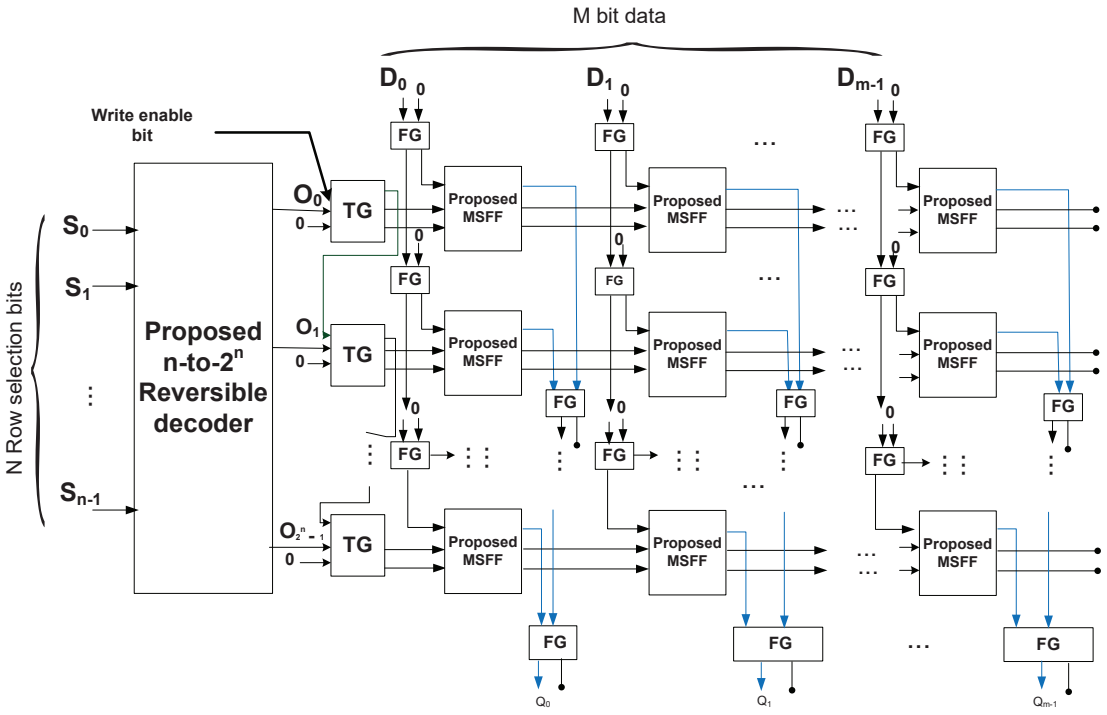


FIG. 4.6: The Proposed architecture of reversible RAM for Plessey FPGA.

The construction procedure of the proposed reversible Random Access Memory for FPGA presented in Algorithm 4 leads the construction of the circuit given in Fig. 4.6. Theorem 4.6.5 - Theorem 4.6.8 define the characteristics of the proposed reversible RAM.

Theorem 4.6.5. *A reversible $2^n \times m$ bit RAM for Plessey Logic Block requires at least $2^n(6m+2) - 2m - 3$ garbage output, where n is the number of control bits of decoder and m is the number of bits of input data of RRAM.*

Proof. Proposed reversible $2^n \times m$ RAM consists of an $n \times 2^n$ decoder, 2^n Toffoli gates, $2^n.m$ write-enable Master-Slave D-Flip-Flops and a $2m(2^n - 1)$ Feynman gates (to copy the data-inputs and to propagate the flip-flops outputs).
Summing up the individual number of gates of these components, we can get the

Algorithm 4: The proposed algorithm to construct the proposed reversible $2^n \times m$ bit RAM

Data: proposed n-to-2^n reversible decoder, proposed reversible MSFF, Toffoli gate (TG) and Feynman gate(FG)

Result: $2^n \times m$ bit RAM

```
1  begin Take one n × 2^n reversible decoder, RD(n) ;
2  Let, I = input, O =output
3  for i ← 1 to 2^n do
4      if i = 1 then
5          W ← I_1.TG_i ;
6      else
7          O_1.TG_{i-1} ← I_1.TG_i;
8      end if
9      O_i.RD ← I_2.TG_i For each output O_i of the decoder ;
10     0 ← I_3.TG_i;
11     for j ← 0 to m − 1 do
12         Take one reversible write enable Master-Slave D-Flip-Flop MSFF_j;
13         D_j → MSFF_j , primary data input of the RAM;
14         if j = 1 then
15             O_2.TG_j → I_CP.MSFF_j , CP= Clock input of Flip-Flop;
16             O_3.TG_j → I_W.MSFF_j, W= write enable bit;
17         else
18             O_CP.MSFF_{j-1} → I_CP.MSFF_j;
19             O_W.MSFF_{j-1} → I_W.MSFF_j;
20         end if
21     end for
22 end for
23 end
```

total number of gate for the $2^n \times m$ RRAM.

A $n \times 2^n$ decoder produce $2^n - 3$ gates (Theorem 4.6.1), where n denotes the number of control bits of the decoder and each reversible write-enable Master-Slave D-Flip-Flops requires 4 gates (Lemma 2). Summing up the number of gates, we get the total number of gates

$= 2^n - 3 + 2^n + 4.2^n.m + 2m.(2^n - 1)$

$= 2^n.(6m + 2) - 2m - 3$

Therefore, A reversible $2^n \times m$ bit RAM for Plessey Logic Block requires at least $2^n.(6m + 2) - 2m - 3$ reversible gate. (Proved) □

Theorem 4.6.6. *A reversible $2^n \times m$ bit RAM for Plessey Logic Block produce at least $4.2^n.m + n - m - 1$ garbage output, where n is the number of control bits of decoder and m is the number of bits of input data of RRAM.*

Proof. Proposed reversible $2^n \times m$ RAM consists of an $n \times 2^n$ decoder, 2^n Toffoli gates, $2^n \times m$ write-enable Master-Slave D-Flip-Flops and a $2m(2^n - 1)$ Feynman gates (to copy the data-inputs and to propagate the flip-flops outputs).
Summing up the individual production of garbage output of these components we can get the total number of garbage outputs for the $2^n \times m$ RRAM.
A $n \times 2^n$ decoder produce $n-2$ garbage outputs (Theorem 4.6.2), where n denotes the number of control bits of the decoder.
Each reversible write-enable Master-Slave D-Flip-Flops requires 3 garbage outputs (Lemma 2).
And $2m.(2^n-1)$ Feynman gates produce one $(2^n-1).m$ garbage outputs (Fig. 4.6).
Hence, the total number of garbage outputs
$= n - 2 + 1 + 4.2^n.m + (2^n - 1).m$
$= 4.2^n.m + n - m - 1$

Therefore, A reversible $2^n \times m$ bit RAM for Plessey Logic Block produces at least $4.2^n.m + n - m - 1$ garbage outputs.(Proved) □

Theorem 4.6.7. *A reversible $2^n \times m$ bit RAM for Plessey Logic Block has at least $2^n.(10+18m)-(2m+13)$ quantum cost, where n denotes the number of control bits of decoder, m denotes the number of bits of input data of RRAM and Δ denotes the unit delay.*

Proof. Proposed reversible $2^n \times m$ RAM consists of an $n \times 2^n$ decoder, 2^n Toffoli gates, $2^n.m$ write-enable Master-Slave D-Flip-Flops and a $2m(2^n - 1)$ Feynman gates (to copy the data-inputs and to propagate the flip-flops outputs).

By summing up the individual quantum cost of these components, we can get the total quantum cost for the $2^n \times m$ RRAM.

A $n \times 2^n$ decoder has $5(2^n - 4) + 7$ quantum cost (Theorem 4.6.3), whereas, each Toffoli gate has five (5) quantum cost and each Feynman gate has one (1) quantum cost.

Each reversible write-enable Master-Slave D-Flip-Flops has 16 quantum cost (Lemma 2).

Thus, the total quantum cost

$= 5(2^n - 4) + 7 + 5.2^n + 16.2^n.m + 2m(2^n - 1)$

$= 2^n(10 + 18m) - (2m + 13)$

Therefore, A reversible $2^n \times m$ bit RAM for Plessey Logic Block has at least $2^n(10 + 18m) - (2m + 13)$ quantum cost.(Proved) □

Theorem 4.6.8. *A reversible $2^n \times m$ bit RAM for Plessey Logic Block has at least $(2^n(6 + m) + 15m - 12)$ Δ delay, where n is the number of control bits of decoder, m is the number of bits of input data of RRAM and Δ denotes unit delay.*

Proof. For delay calculation, we need to setup the critical path of the RRAM. The last write-enable Master-Slave D-Flip-Flops (in 2^{nth} Row and m^{th} Column) is responsible for read of write the last bit of input data. For write a bit to (or read a bit from) this memory unit, we must go through an $n \times 2^n$ decoder, 2^n Toffoli gates, m write-enable Master-Slave D-Flip-Flops, and $2m(2^n - 1)$ Feynman gates. An $n \times 2^n$ decoder has $5(2^n - 4) + 7\Delta$ delay (Theorem 4.6.4), whereas, each Toffoli gate has delay of 1Δ and each Feynman gate has delay of 1Δ.

In addition, each reversible write-enable Master-Slave D-Flip-Flop has delay of 16Δ (Lemma 2).

Thus, the total delay

$= (5(2^n - 4) + 7 + 5.2^n + 16.m + 2m(2^n - 1)\ )\ \Delta$

$= (2^n(6 + m) + 15m - 12)\ \Delta$

Therefore, A reversible $2^n \times m$ bit RAM for Plessey Logic Block has at least delay of $(2^n(6 + m) + 15m - 12)\ \Delta$.(Proved) □

4.6.5 Proposed Reversible Multiplexer

This section presents the proposed design of the reversible multiplexer.

Multiplexer causes transmission of a large number of information units over a smaller number of channels. Architecturally, a digital multiplexer is a logic circuit that puts one out of several inputs to a single output. A set of select inputs controls the inputs. Generally, a multiplexer has m inputs and n select input, where, $m = 2^n$.

A 2:1 multiplexer is the smallest unit of the proposed architecture of reversible multiplexer. The characteristics function of a multiplexer is $s_0' I_0 \oplus s_0 I_1$, which can be determined by several gates [49, 50, 52, 100]. In this thesis, Modified Fredkin gate is used as a 2:1 reversible multiplexer as it has the least quantum cost in the literature.

Let, I_0 and I_1 are the inputs and S_0 is the select input of a 2:1 multiplexer. When S_0=0, then I_0 input transmit to the output Y and when S_0=1, then I_1 input transmit to the output Y. Fig. 4.7 shows the architecture of reversible 2:1 multiplexer by using Modified Fredkin gate.

The quantum cost and delay of this reversible 2:1 multiplexer is 4 and 4Δ, respectively. Moreover, the number of garbage output is 2.

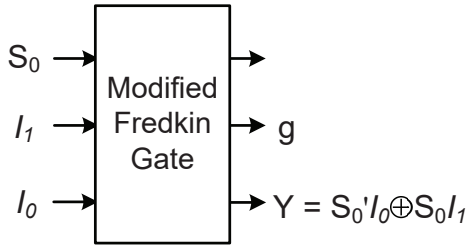
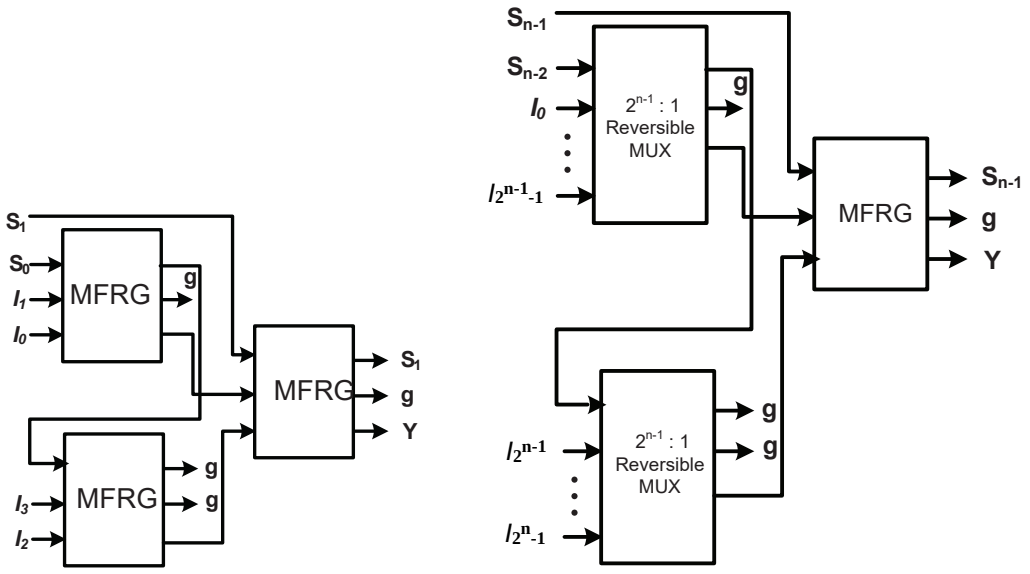


FIG. 4.7: Modified Fredkin gate as reversible 2:1 multiplexer.

TABLE 4.4: Function of S_0 and S_1 select lines

S_0	S_1	**Output (O)**
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



(a) 4:1 reversible multiplexer.

(b) $2^n : 1$ multiplexer.

FIG. 4.8: the proposed designs of reversible multiplexers.

The reversible 4:1 multiplexer (2^2 : 1 multiplexer) has four input, two select inputs, and one output. Fig. 4.8(a) shows the proposed design of reversible $4 : 1$ multiplexer, where I_0, I_1, I_2, and I_3 are the inputs, S_0 and S_1 are the select inputs. The bit combination of select inputs controls the function of $4 : 1$ multiplexer (Table 4.4). Three Modified Fredkin gates are used in this design (4.8(a)). So, the quantum cost of the proposed reversible $4 : 1$ multiplexer is 12 (3×4) and delay of the proposed reversible $4 : 1$ multiplexer is 12 (3×4) Δ, respectively, whereas, the garbage output is 3.

As the consequence of the design of reversible multiplexers, a $2^n : 1$ multiplexer can be constructed using two $2^{n-1} : 1$ reversible multiplexers and one 2:1 reversible multiplexer shown in the Fig. 4.8(b). The properties of $2^{n-1} : 1$ reversible multiplexer are given in Lemma 4.

Lemma 4. *A reversible* $2^n : 1$ *multiplexer for Plessey Logic Block can be designed with* $2^n - 1$ *gate which produces* $2^n + n - 1$ *garbage outputs. It also requires* $4(2^n - 1)$ *quantum cost and delay of* $4(2^n - 1)\Delta$. *Where, n denotes the number of selection input, and* Δ *denotes the unit-delay.*

4.6.6 Proposed Reversible NAND unit

The rest of the components of Plessey Logic Block is a NAND unit. The reversible Peres gate is ideal to construct a NAND unit. The Peres gate in Fig. 4.9 acted as NAND unit.

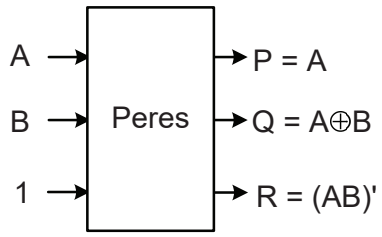


FIG. 4.9: The proposed reversible NAND unit of Plessey Logic Block.

4.7 Implementation of the Reversible Plessey Logic Block

In Chapter 2, the block diagram of the Plessey Logic Block is given (Fig. 2.19). Architecturally, the Field Programmable Gate Array is a set of logic blocks connected with programmable wires. Each end of a wire segment typically has a switch attached. The routing resources such as track and routing channels are responsible

to route among the blocks. In each of the logic block, there are multiplexer, configurable RAM, NAND unit, and D-Latch. The multiplexers are controlled by the RAM cells and the outputs of two 4:1 multiplexers are assigned as the inputs to the NAND unit. The output of NAND unit and the configurable bit from the RAM go to D-Latch. The inputs to each multiplexer are connected to either the output of the previous NAND Gate in the row or the output of the NAND gate above or below this logic block, whichever is closer, the other blocks are connected similarly [101].

In the prior sections, proposed reversible architectures for the logic elements (reversible Multiplexer, reversible RAM, reversible D-Latch and reversible NAND unit) of Plessey Logic Block of FPGA are described. In this section, an algorithm (Algorithm 5) is defined to combine all the components of the logic block which is shown in Fig. 4.10.

The general properties of this proposed circuit are illustrated in Theorem 4.7.1-Theorem 4.7.3.

Theorem 4.7.1. *A reversible Plessey Logic Block of FPGA requires at least* $(r_{gt}+9)$ *reversible gates, where,* r_{gt} *denotes the number of gates required to design the RRAM.*

Proof. Plessey Logic Block consists of an 8:2 multiplexer, a D-Latch, a NAND unit and a $2^n \times m$ bit RAM. Summing up the individual requirement of gates of these components we can get the total requirement of gate for the logic block. A 2^n: 1 multiplexer needs $2^n - 1$ gates (Lemma 4), where n denotes the number of selection bits of the multiplexer. Putting n = 2, we get that a 4:1 multiplexer requires 3 gates. Thus an 8:2 multiplexer (two 4:1 multiplexer) needs 3 + 3= 6 gates.

The reversible D-Latch requires 2 reversible gates (Lemma 1) and the RRAM requires $2^n(6m+2) - 2m - 3$ reversible gates (Theorem 4.6.5).

Let, $2^n(6m+2) - 2m - 3 = r_{gt}$.

A NAND unit requires a single Peres gate.

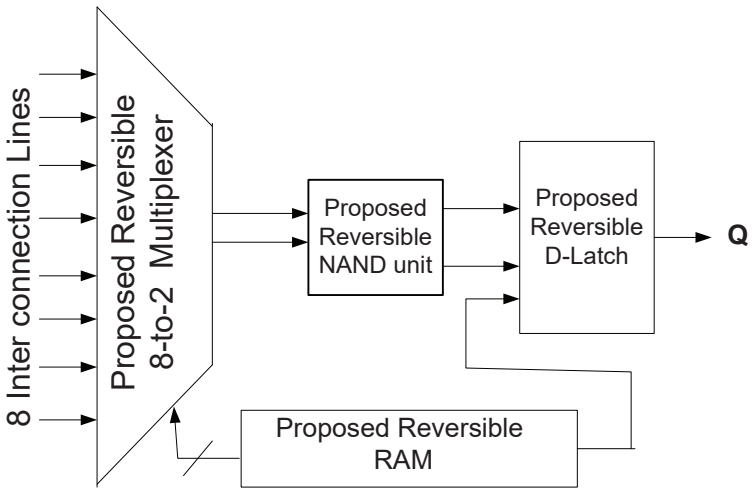


FIG. 4.10: Proposed Reversible Plessey Logic Block of FPGA

Hence, a Plessey Logic Block can be constructed with at least $(r_{gt} + 6 + 2 + 1) = (r_{gt} + 9)$ reversible gates. □

Algorithm 5: The proposed algorithm to construct the proposed reversible Plessey Logic Block of FPGA

Data: 8-bit interconnection

Result: Q, which will be sent to the next logic block

1 begin:

2 Take selection bits from RRAM to 8:2 multiplexer.

3 Multiplex the select bits of line 2 by 8:2 multiplexer.

4 Set $i_1.Peres \leftarrow o_1.multiplexer$, $i_2.Peres \leftarrow o_2.multiplexer$, $i_3.Peres \leftarrow 1$

5 Set $i_1.D - Latch \leftarrow o_1.Peres$, $i_1.D - Latch \leftarrow o_1.Peres$ Set $i_1.D - Latch \leftarrow$ one control bit from.RRAM which will make the D-Latch transparent if the latch is not needed

6 return Q;

7 end

Theorem 4.7.2. *A reversible Plessey Logic Block of FPGA produces at least* $(r_{gb}+13)$ *garbage outputs, where* r_{gb} *denotes the number of garbage outputs generated from RRAM.*

Proof. Likewise Theorem 4.7.1 we can also prove the Theorem 4.7.2 by cumulating the garbage outputs of the components of the logic block.
A 2^n: 1 multiplexer needs $2^n + n - 1$ garbage (Lemma 4), where n is the number of selection bits of the multiplexer. Putting n= 2, we get a 4:1 multiplexer produces $5(= 2^2 + 2 - 1)$ garbage outputs. Thus an 8: 2 multiplexer (two 4:1 multiplexer) needs 10 garbage outputs. The reversible D-Latch can be designed by 2 garbage outputs (Lemma 1), NAND unit(3×3 Peres gate) produces 1 garbage and at least $4.2^n.m + n - m - 1$ garbage outputs are generated by the RRAM (Lemma 4.6.6).
Let $4.2^n.m + n - m - 1 = r_{gb}$.
Hence, the proposed Plessey Logic Block produces at least $(r_{gb} + 10 + 2 + 1) = (r_{gb} + 13)$ garbage outputs. (Proved) □

Theorem 4.7.3. *A reversible Plessey Logic Block requires at least* $(r_{qc} + 34)$ *quantum cost, where* r_{qc} *be the quantum cost of RRAM.*

Proof. A 4:1 multiplexer requires three Modified Fredkin gate, each of which has 4 quantum cost.
Thus an 8: 2 multiplexer (two 4:1 multiplexer) needs $24(= 2 \times 3 \times 4)$ quantum cost.
The reversible D-Latch (with Q and Q') has 6 quantum cost (Lemma 1) and the RRAM has $2^n(10 + 18m) - (2m + 13)$ quantum cost (Lemma 4.6.7).
Let, $2^n(10 + 18m) - (2m + 13) = r_{qc}$.

Hence, along with a NAND unit which is a single Peres gate of 4 quantum cost, the proposed logic block has at least $(r_{qc} + 24 + 6 + 4) = (r_{qc} + 34)$ quantum cost. (Proved) □

4.8 Simulation and Performance Evaluation

In this section, the proposed reversible HND gate, HNF gate and the proposed reversible components of Plessey Logic Block are simulated and the performance

is analyzed to discover the efficiency of the proposed components.

4.8.1 Simulation Environment

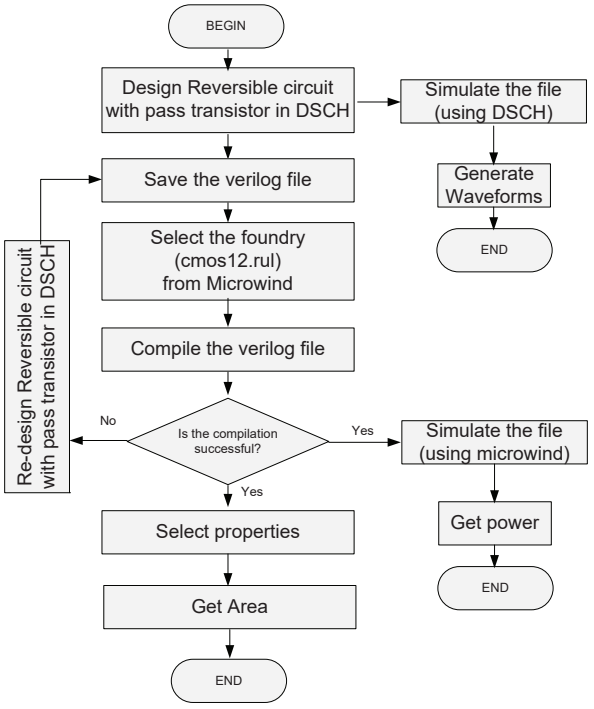


FIG. 4.11: Simulation flow to get the waveforms, the area and the power of the proposed reversible components.

The simulations of the proposed circuits are done using DSCH 3.5 [93] and Microwind 3.5 [94] softwares on a computer, which has the Intel(R) Core(TM) i7 CPU with 2.10 GHz Clock Speed and 2.00 GB RAM. We designed the reversible circuits with the pass transistors, simulated them to get the waveforms in DSCH. Transistor implementation of the proposed circuits followed the procedure proposed in [74]. Then, Microwind was used to get the power and area. The simulation flow used in this work is shown in Fig. 4.11.

4.8.2 Simulation Results

The simulations of the proposed components (reversible D-Latch, reversible write enable Master-Slave Flip-Flop, reversible Multiplexer, reversible 2-to-4 Decoder

and reversible 3-to-8 Decoder) are presented in Fig. 4.12 - 4.16, respectively.

The simulation result of the proposed reversible D-Latch is shown in Fig. 4.12, where CLK indicates clock input, D is for data input, Q is the non-complementary output of D-Latch, and the Q^+ notation denotes the negation of Q. Fig. 4.12 shows that when $CLK = 1$, the value of D becomes the output Q. When $CLK = 0$, the latch maintains its previous state.

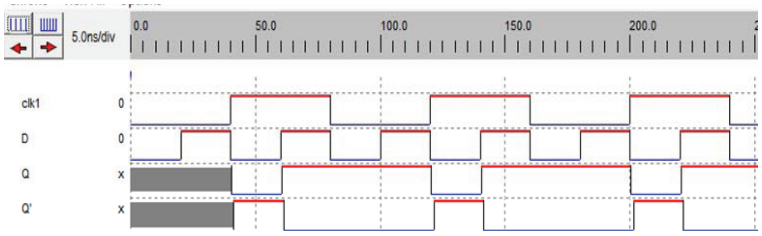


FIG. 4.12: Simulation of the proposed reversible D-Latch.

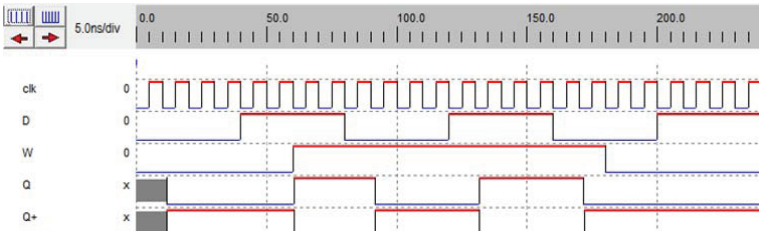


FIG. 4.13: Simulation of the proposed reversible write enable Master-Slave Flip-Flop.

Then, Fig. 4.13 shows the simulation result of the write-enable Master-Slave Flip-Flop, where CLK, D, Q, and Q^+ denotes the same as the corresponding notations of D-Latch. Additionally, W denotes the write bit. When $W = 1$, depending on the value of CLK, the value of Q varies like the Q of D-Latch discussed in the previous paragraph.

In addition, the simulation result of reversible 4:1 multiplexer is drawn in Fig. 4.14, where based on the combination of two selection bits (S_1 and S_0), the output denoted by O_{mux} is assigned any of the four inputs (I_0, I_1, I_2, and I_3) which proves the correctness of the design.

Finally, the simulation result of the 2-to-4 Decoder and 3-to-8 Decoder are shown in Fig. 4.15 and 4.16. Based on the binary combination of two control bits (s_1 and s_0) of 2-to-4 Decoder, the 4 outputs (denoted by O_0, O_2, ..., O_3) are generated which proves the exactness of the design. For example, when $S_0 = 0$, and $S_1 = 1$, the output O_1 is activated. Similarly, based on the binary combination of three control bits (S_2, S_1 and S_0) of 3-to-8 Decoder, the eight outputs denoted by (O_0, O_1, ..., O_7) are generated which proves the correctness of the design. For example, when $S_0 = 1$, $S_1 = 0$, and $S_2 = 0$, the output O_4 is activated.

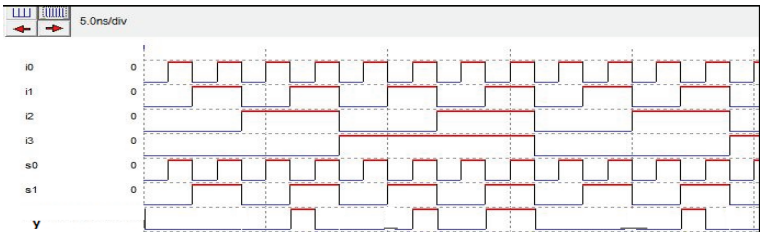


FIG. 4.14: Simulation of the proposed reversible 4 : 1 Multiplexer.

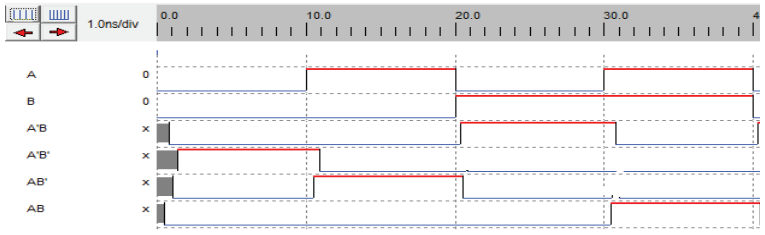


FIG. 4.15: Simulation of the proposed reversible 2-to-4 Decoder.

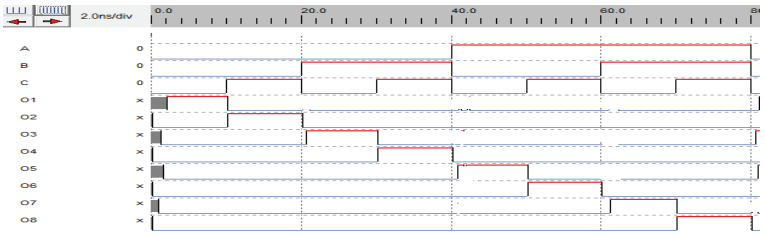


FIG. 4.16: Simulation of the proposed reversible 3-to-8 Decoder.

4.8.3 Performance Metrics

We consider six criteria as the metrics of the performance analysis of reversible FPGA. Brief description of these metrics are given below.

- Gates: Reversible gates are used to design a specific circuit. Accumulating all the gates and counting the total number of gates will be found. This is a performance metric of reversible circuits. Less number of required gate certifies the high performance of the circuit.

- Garbage outputs: Garbage output is measured by counting the unused output of a circuit. By accumulating all the garbage outputs, we can find the total number of garbage outputs. The minimum value of garbage outputs proves the efficiency of the design.

- Quantum Cost: It is one of the primary performance metrics of the reversible circuit. The quantum cost of a reversible gate is the cumulative number of NOT, Controlled-V, Controlled-V^+, and $CNOT$ gates required in its implementation.

- Delay: For delay calculations in the reversible circuits, the reversible gates in the critical path are replaced with their corresponding quantum gates and the total delay of the critical path determines the delay of the circuit. The delay of each 1×1 or 2×2 reversible gate is taken as unit-delay and denoted as Δ.

- Area: The area of a logic circuit means the total area accumulated by the individual circuit element. Microwind [94] can be used to calculate the area of each individual element. As an example, if a circuit consists of n gates with area A_1, A_2,..., A_n, then the area (A) of that circuit is as follows-

$$A = \sum_{i=1}^{n} A_i \tag{4.1}$$

Using Microwind [94], we can find the area of a 4:1 multiplexer (which has three Modified Fredkin gate). The area of a Modified Fredkin gate 0.00152 mm^2 and hence, the total area of 4:1 multiplexer is 0.00456 mm^2.

- Power: The power consumption of a logic circuit means the total power consumption by the individual circuit elements. Power of the reversible circuit can be calculated using DSCH 3.5 [93] and Microwind 3.5 [94]. The transistors or pass transistors are used to design the corresponding circuit in DSCH. Then power can be estimated by using Microwind.

4.8.4 Performance Analysis

Performance analysis of the different components of the Plessey Logic Block is given in this section.

4.8.4.1 Performance of Proposed Reversible D-Latch

TABLE 4.5: Comparison of the proposed and the existing reversible D-Latches with simultaneous Q and Q' output

	Gates	Garbage	Delay (Δ)	Quantum Cost
Proposed	2	2	6	6
Existing [52]	2	2	7	7
Existing [50]	2	2	12	11
Existing [36, 102, 103]	3	1	8	7
Existing [104]	2	1	14	10
Existing [96]	3	2	15	11

There are a variety of ways the researchers design the D-Latch [36, 50, 52, 96, 102, 103, 104] . The designs use reversible gates which are in different number and different type. The proposed D-Latch is constructed using one HNF gate (which has 5 quantum cost and 5d delay) and one Feynman gate (which has 1 quantum cost and 1d delay). The gate choice and the construction procedure guarantees the enrichment of the proposed D-Latch in terms of delay and the quantum cost for all compared design, while, it shows improvement in terms of the gate for

some designs [96]. For example, the proposed D-Latch has 6 quantum cost and 6 delay, whereas, the best existing design has 7 quantum cost and 7 delay and the design [96] (among the compared D-Latch) has 11 quantum cost and 15 delay.

A comparison of the proposed design of reversible D-Latch (producing simultaneous Q and Q') with the existing designs of reversible D-Latch is also presented in Table 4.5. It Illustrates that the proposed design reduces the number of gates by 33.33% $[((3-2)/3) \times 100\%]$ compared to [36, 102, 103] and [96], respectively . In addition, it achieves 14.28%, 25%, 50%, 57.14%, and 60 % improvement in the reduction of delay compared to [52], [36, 102, 103], [50], [104], and [96], respectively. In case of the reduction of quantum cost, the proposed design achieves 14.28% improvement compared to [52] and [36, 102, 103], 40% improvement compared to [104], and 45.45% [50] and [96], respectively, while producing two garbage outputs.

4.8.4.2 Performance of Proposed Reversible Master-Slave Flip-Flop

In addition, from Table 4.6, we find that the proposed design of Master-Slave Flip-Flop outperforms the existing designs in terms of the number of gates (the improvements are 66.66%, 40%, 25%, and 40%), delay (the improvements are 65.71%, 15.38%, 8.33%, and 15.38%) and quantum cost (the improvements are 71.05%, 15.38%, 8.33%, and 15.38%) with respect to [36, 54, 103, 104], respectively, while the design is also enhanced in terms of garbage outputs (the improvements are 16.66%, 33.33 %, and 50% with respect to [54, 103, 104] , respectively). Similarly, it can be found that the proposed Master-Slave Flip-Flop is better than the existing Master-Slave Flip-Flops designed for the Plessey Logic Block of FPGAs in term of the number of gates (the improvements are 57.14%, 40%, and 57.14% with respect to [50], [51], and [52], respectively), garbage outputs (the improvements are 77.77% and 71.4% with respect to [50] and [52], respectively), delay (the improvements are 26.66% and 47.62% with respect to [51] and [52],

respectively) and quantum cost (the improvements are 56%, 26.66 % and 52.17% with respect to [50], [51] and [52], respectively).

TABLE 4.6: Comparison of the proposed and the existing reversible Master-Slave Flip-Flops. ('-' indicates not calculated)

	Gates	Garbage	Delay (Δ)	Quantum Cost
Proposed	3	2	11	11
Existing [52]	7	7	21	23
Existing [51]	5	2	15	15
Existing [50]	7	9	-	25
Existing [103]	4	3	12	12
Existing [36]	5	3	13	13
Existing [104]	5	4	13	13
Existing [54]	9	12	35	38

4.8.4.3 Performance of Proposed Reversible Decoder

The proposed HND gate is able to generate all the four terms of the 2-to-4 decoder. This HND gate or 2-to-4 decoder has 7 quantum cost and 7d delay, which is the least in the literature. Based on this concept, the performance comparison of the proposed decoders and the existing 2-to-4 decoders [51, 52, 96, 97, 98] and 3-to-8 decoders [51, 52, 98] have been shown in Table 4.7 and Table 4.8, respectively.
Table 4.7 shows that the proposed 2-to-4 decoder outperforms the existing designs in terms of the number of gates (the improvements are 66.67%, 66.67%,90% , 66.67% and 75%), delay (the improvements are 41.67%, 41.67%, 22.22%, 66.67% and 14.28%) and quantum cost (the improvements are 41.67%, 41.67%, 75% , 53.33% and 41.67%) with respect to [51, 52, 96, 97, 98] , respectively. It is also found that the improvements of the proposed 2-to-4 decoder in terms of garbage output is outstanding, i.e., 100% improvement compared to [51, 52, 96, 97, 98] as proposed design is garbage free, i.e., number of garbage is 0(zero) (Fig. 4.4(b))[This is already discussed in the Section 4.6.3].

On the other hand, Table 4.8 demonstrates the efficiency of the proposed 3-to-8 decoder. It can be viewed that the proposed 3-to-8 decoder is also more feasible than the existing ones in terms of the number of gates (improvements are 28.57%, 28.57% and 54.54%) and garbage outputs (improvements are 66.67% and 83.33%) compared to [51, 52] and [98] , respectively, while this proposed design of 3-to-8 decoder defeats the designs proposed in [51, 52] (improvement is 15.62%) in terms of both delay and quantum cost.

TABLE 4.7: Comparison of the proposed and the existing reversible 2-to-4 decoders

2-to-4 decoder	Gates	Garbage	Delay (Δ)	Quantum Cost
Proposed	1	0	7	7
Existing [51, 52]	3	2	12	12
Existing [98]	4	2	8	12
Existing [97]	3	2	21	15
Existing [96]	10	8	9	28

TABLE 4.8: Comparison of the proposed and the existing reversible 3-to-8 decoders

3-to-8 decoder	Gates	Garbage	Delay (Δ)	Quantum Cost
Proposed	5	1	27	27
Existing [51, 52]	7	3	32	32
Existing [98]	11	6	32	32

4.8.4.4 Performance of Proposed Reversible Multiplexer

Subsequently, performances of different reversible multiplexers are depicted in Table 4.9, which articulates the supremacy of the proposed design of multiplexer having 33.33%, 57.14% and 50% improvements in terms of the number of gates, having 54.54 %, 54.54 % and 50% improvement in terms of garbage output compared to [50], [100] and [49], respectively. Besides, this design is significant one as it reduces delay of 20% compared to [52], while delay in [49, 50, 100] is undefined. It also overcomes the challenges of reducing quantum cost having 20% reduced

quantum cost compared to[52] and 78.94% reduced quantum cost compared to [50, 100], respectively.

TABLE 4.9: Comparison of the proposed and the existing reversible 4 : 1 multiplexer ('-' indicates not calculated).

Multiplexers	Gate	Garbage	Delay (Δ)	Quantum cost
Proposed	3	5	12	12
Existing [52]	3	5	15	15
Existing [50]	9	11	-	57
Existing [49]	7	11	-	57
Existing [100]	6	10	-	-

4.8.4.5 Performance of Proposed Reversible RAM

The efficient design of the individual components also makes the design of the RRAM efficient. By combining the respective cost parameters of the individual components of the proposed RRAM, we got the total cost of a 4×2 RRAM, which is compared with the existing reversible RAMs in Table 4.10 as an example of the efficacy of the proposed RRAM. This table shows that the proposed design of RRAM surpasses the limitations of the existing designs in terms of the number of gates (improvements are 45.33% , 46.75%, 45.33%, 21.15% and 25.45%) and garbage (improvements are 62.03%, 64.71%, 62.03%, 6.25% and 28.57%) with respect to [51, 52, 99, 105, 106], respectively , while the proposed design defeats other designs in terms of delay (improvements are 7.14%, 21.21% and 27.78% with respect to [52, 105, 106] , respectively) and in terms of quantum cost (improvements are 22.91%, 23.58% and 26.78 % with respect [51], [99], and [52, 106] respectively).

4.8.4.6 Other Performance analysis of Plessey Logic Block

Beyond the previous performance analysis, comparison of different elements of Reversible Plessey Logic Block with existing designs in terms of the number of transistors, area, and power in Table 4.11 - 4.13, respectively, show the supremacy

TABLE 4.10: Comparison of the proposed and the existing reversible 4 × 2 RAMs.

	Gates	Garbage	Delay (Δ)	Quantum Cost
Proposed	49	31	52	167
Existing [52]	77	85	56	239
Existing [51]	75	79	46	227
Existing [105]	52	32	66	177
Existing [106]	55	42	72	239
Existing [99]	75	79	46	229

of the proposed designs. Finally, in Table 4.14, proposed Reversible Plessey Logic Blocks (4 × 2) is compared with existing corresponding designs, which ensures the efficiency of the proposed design showing improvement in each parameter (improvements are 51.62%, 80.83%, and 54% with respect to [52], whereas, improvements are 58.23%, 73.57%, and 34.12% with respect to [50], in terms of the number of the transistors, area, and power, respectively).

TABLE 4.11: Comparison of the proposed and the existing reversible elements of reversible Plessey Logic Block in terms of the number of transistors

	Proposed	Existing [52]	Existing [50]
2-to-4 Decoder	8	20	20
3-to-8 Decoder	24	36	36
D-Latch	14	16	26
Write enable Master-Slave Flip-Flop	30	60	68
2 : 1 Multiplexer	7	4	50
4 : 1 Multiplexer	21	12	118
NAND unit	8	12	14
4x2 RAM	304	700	664

The efficiency of the proposed reversible Plessey Logic Block of FPGA is easily conceivable as the individual components have already been compared with the existing ones. The optimum result in each design case ensures the efficiency in the cumulative result also. The enhancement of the proposed design of the components of Plessey Logic Block are also presented graphically in Fig. 4.17 - 4.21.

TABLE 4.12: Comparison of the proposed and the existing reversible elements of reversible Plessey Logic Block in terms of area

	Proposed are	Existing [52]	Existing [50]
2-to-4 Decoder	0.0389	0.24869	0.24869
3-to-8 Decoder	0.4909	0.70125	0.70125
D-Latch	0.030148	0.13555	0.076917
Write enable Master-Slave Flip-Flop	0.0632	0.42906	0.278574
2 : 1 Multiplexer	0.00152	0.11314	0.285665
4 : 1 Multiplexer	0.00456	0.33942	0.542305
NAND unit	0.016154	0.13555	0.021954
4x2 RAM	0.618356	4.49229	2.622238

Unit of area is (mm^2)

TABLE 4.13: Comparison of the proposed and the existing reversible elements of reversible Plessey Logic Block in terms of power

	Proposed	Existing [52]	Existing [50]
2-to-4 Decoder	2.149	3.72	0.24869
3-to-8 Decoder	5.213	6.784	0.70125
D-Latch	1.38	2.934	3.906
Write enable Master-Slave Flip-Flop	5.808	10.97	8.64
2 : 1 Multiplexer	1.229	0.766	4.80625
4 : 1 Multiplexer	2.236	2.298	9.45625
NAND unit	1.139	2.934	1.17
4x2 RAM	54.268	129.232	82.515

Unit of power is (mW)

TABLE 4.14: Comparison of the proposed and the existing reversible Plessey Logic Block (4×2) in terms of the number of transistors, area & power

	No. of Transistor	Area (mm^2)	Power (mW)
Proposed	416	1.263738	73.422
Existing [52]	860	6.59495	159.638
Existing [50]	996	4.777593	111.4434
Improvement w.r.t.[52](%)	51.62%	80.83%	54%
Improvement w.r.t.[50](%)	58.23%	73.57%	34.12%

In Fig. 4.17, comparison of different reversible D-Latches with simultaneous Q and Q' output is given. This bar graph shows that the proposed decoders are the best in terms of all parameters (the number of gates, garbage outputs, delay, and quantum cost). Fig. 4.18 represents the performance analysis of the proposed and existing Master-Slave Flip-Flops. The bar graphs ensure the significant efficiency

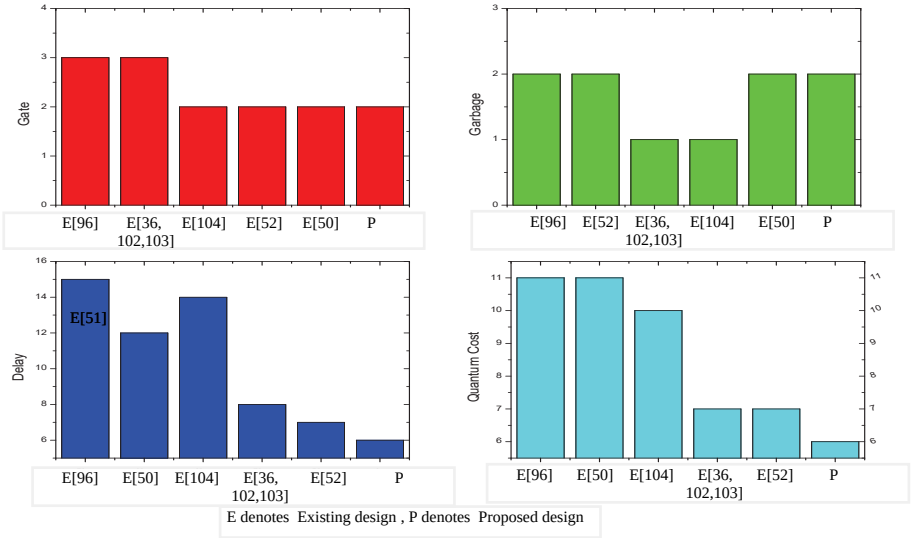


FIG. 4.17: Performance analysis of different reversible Flip-Flops with simultaneous Q and Q' output

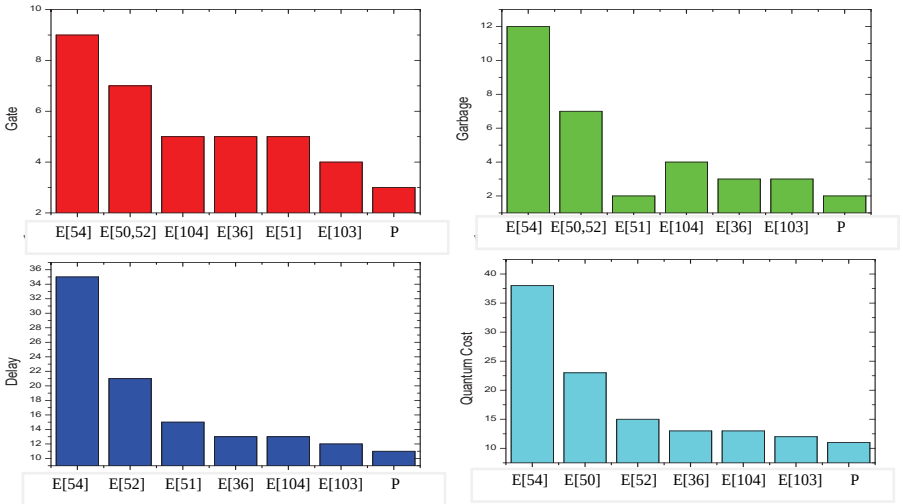


FIG. 4.18: Performance analysis of different reversible Master-Slave Flip-Flops

of the proposed design in terms of the all performance metrics..

In addition, bar graph (Fig. 4.19) illustrates the comparisons of different reversible

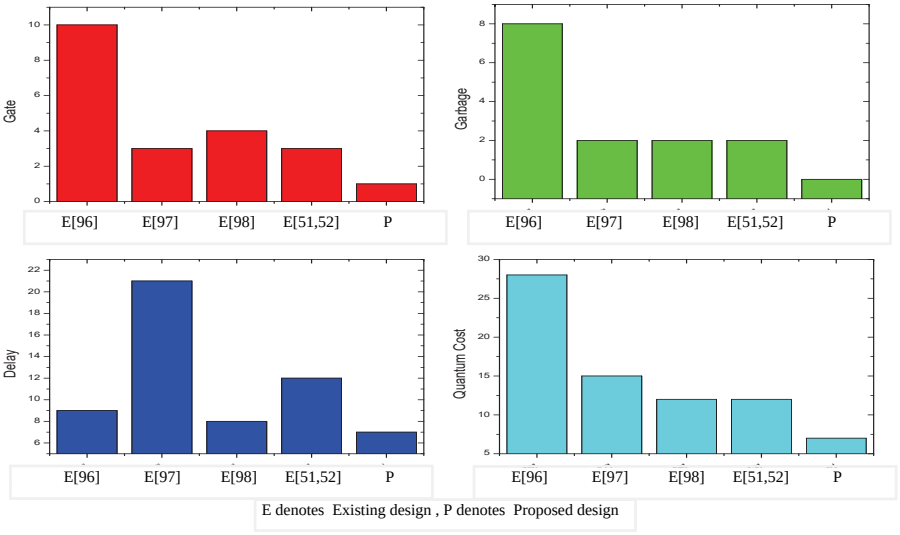


FIG. 4.19: Performance analysis of different reversible 2-to-4 decoders.

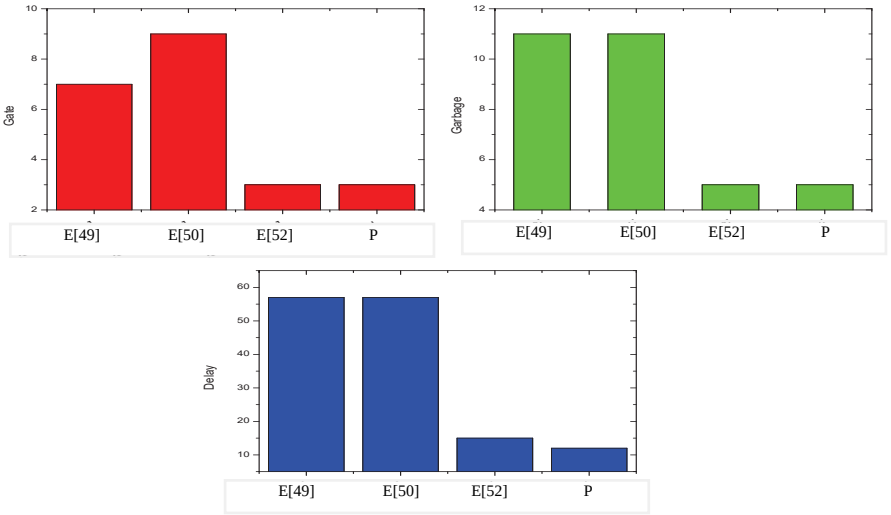


FIG. 4.20: Performance analysis of different reversible 4 : 1 multiplexers.

2-to-4 decoders which ensure the nobility of the proposed 2-to-4 decoder. Afterward, the effectiveness of the proposed 4:1 multiplexer is unraveled by the comparison with existing reversible 4:1 multiplexers shown in Fig. 4.20. Finally, bar graphs shown in Fig. 4.21 represent the cost evaluation of the proposed reversible 4×2 RAM. Efficient design of each component makes the RAM efficient, i.e.,

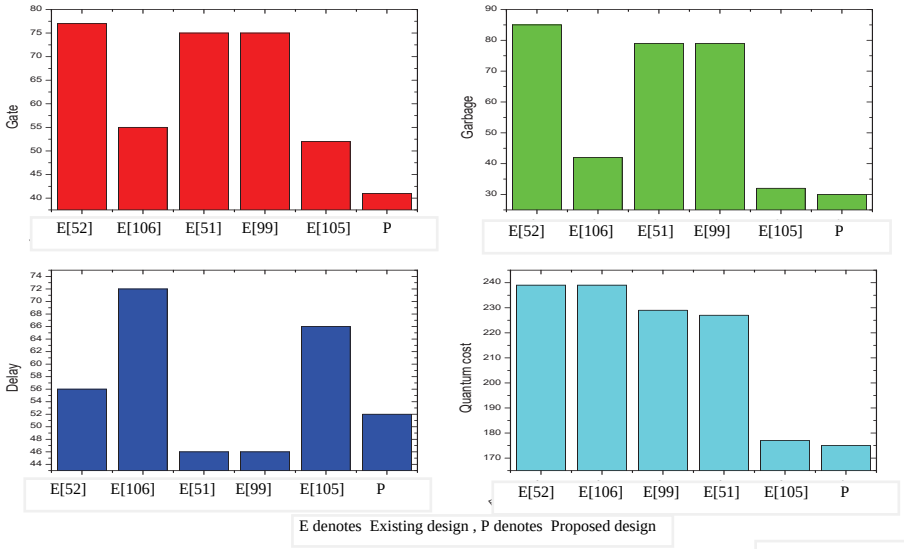


FIG. 4.21: Performance analysis of different reversible 4×2 RAMs

it requires less number of gates, produces less number of garbage outputs and quantum cost. In few cases, the existing RAMs have less delay than the proposed RAM, but the gap is very low.

From the extensive analysis of the proposed designs and existing designs, it is the fair claim that our proposed design of the Plessey Logic Block of FPGA is the conqueror.

4.9 Summary

In this chapter, reversible Field Programmable Gate Array is proposed. First of all, the architecture of Plessey logic block and motivation toward this device is presented. Analysis of existing designs of reversible Plessey logic block is briefly described. Then, to diminish the excessive cost metrics of the previous designs, new design approaches are elaborately presented in this chapter. Several lower bounds are given in terms of the number of gates, quantum cost, and delay. Scalability, correctness, and supremacy are ensured by adopting theorems, simulations, and performance analysis.

Chapter 5

Conclusions

5.1 Summary of Research

The reversible logic provides a basis for quantum computation with its applications and solves the power dissipation problem of conventional irreversible circuits. This dissertation focused on the reversible realization of two of the mostly used Programmable Logic Devices (PLDs) namely Programmable Logic Array (PLA) and Field Programmable Gate Array (FPGA). The proposed circuits, in this thesis, are expected to be efficient that ensure the cost-effectiveness, scalability, and generality.

In the first part of this thesis, we proposed a compact Reversible Programmable Logic Array (RPLA). In this part, we propose a new reversible gate namely TB gate to generate two sharable sub-products at a time. Then, two developed algorithms realize multi-output ESOP (Exclusive-OR Sum of Product) function by using the TB gate along with existing Feynman and Feynman Double gates. The combination approach of these gates eliminated the dedicated lines for complemented forms of input variables. It also enabled the simultaneous realization of sharable product terms. Reorder of the output function further included in the algorithm. These three features ensure the reduction of gates, garbage outputs,

and quantum cost.

Tables of comparative analysis for a sample circuit and the benchmark circuits show the cost-effectiveness and guarantee the generality. We also calculate the area and power of the benchmark circuits. The proposed design algorithm reduces the number of gates by 13.5%, 5.6% and 10.4% with respect to [46], [47] and [48], respectively. Then, it reduces garbage outputs by 7%, 38% and 19.8% with respect to [46], respectively, whereas, it declines the quantum cost 19% than [46], and 10.5% than [47]. Finally, we show the graphical interpretations from the performance analysis to give a clear view of the improvement of the proposed algorithm.

In the second part of the thesis, we presented the design of Reversible Plessey Logic Block of FPGA which had the lower number of performance metrics (the number of gates, garbage outputs along with the quantum cost, delay, area, and power). Plessey logic block has a NAND unit, RAM, Multiplexer, and Latches. The RAM requires an n-to-2^n Decoder and a Master-Slave Flip-Flop. In this part, we propose three algorithms to design a compact reversible n-to-2^n Decoder, a $2^n \times m$ reversible RAM and the reversible Plessey Logic Block of FPGA, respectively. Here, we also proposed the reversible D-Latch, reversible multiplexer, and write-enable master-slave Flip-Flop. The proposed circuits are more scalable and general as we proposed all the component in terms of n and m. Here we present several theorems and lemmas which illustrate the lower bounds on the number of gates, garbage outputs, and quantum cost of the individual component as well as the complete reversible Plessey Logic Block. These theorems and lemmas prove the efficiency and supremacy of the proposed designs. The examples, followed these theorems, clarify the representative analysis in details. Afterward, we apply pass-transistor logic using Microwind simulator for calculating power and area. The simulation waveforms also show the correctness of the proposed circuits. Comparative tables have the evidence that the proposed circuits require less number of

gates, garbage outputs, and quantum cost. Besides, the tables show the efficiency in terms of delay as well as the number of transistors, area, and power of the designed circuits. Finally, the graphical results clearly state the supremacy of the proposed circuits.

5.2 Future Work

Although our proposed methodologies for reversible Programmable Logic Array and reversible Field Programmable Gate Array achieve a better result as they show better performance in terms of every cost parameter, further theoretical and experimental extensions are possible.

Based on the opportunity of sharing the initial sub-products and eliminating the complimentary line ensure the compactness of the reversible Programmable Gate Array. The algorithm of RPLA also ensure the reordering the output functions to enhance the performance; however, how the heterogeneous format of the benchmark function impacts on the performance of the PLA is not exposed clearly, i.e., how the number of input-outputs or the product terms influence the performance is absent. Moreover, in this thesis, the performances of some particular benchmark functions are chosen and there is no indication of how the number of gate, garbage outputs or quantum cost varies where the sharing property is unable. Therefore, in future, we want to explore more elaborately the correlation between the functions' properties and the algorithmic properties with the simulation process. We can also focus on further reduction mechanism such as decomposition or genetic algorithm.

In this thesis, we only emphasis on the design of a logic block of an FPGA. In future, we may also focus on the other parts of the FPGA. Architectural perspective of reversible RAM as well as the whole logic block is also under the contemplation

in further work for the Plessey logic block. Other type of logic blocks are also under consideration of forthcoming exploration.

Bibliography

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM journal of research and development*, vol. 5, no. 3, pp. 183–191, 1961.

[2] D. Maslov, G. W. Dueck, and D. M. Miller, "Templates for toffoli network synthesis," in *International Workshop on Logic Synthesis*, California, USA, May 2003, pp. 320–326.

[3] G. E. Moore, "Cramming more components onto integrated circuits, electronics, 38: 8 (1965)," *URL: ftp://download. intel. com/research/silicon/-moorespaper. pdf*, vol. 16, 2005.

[4] V. V. Zhirnov, R. K. Cavin, J. A. Hutchby, and G. I. Bourianoff, "Limits to binary logic switch scaling-a gedanken model," *Proceedings of the IEEE*, vol. 91, no. 11, pp. 1934–1939, 2003.

[5] C. H. Bennett, "Logical reversibility of computation," *IBM journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[6] M. A. Nielsen and I. L. Chuang, "Quantum computation and quantum information," 2000.

[7] N. Gershenfeld and I. L. Chuang, "Quantum computing with molecules," *Scientific American*, vol. 278, no. 6, pp. 66–71, 1998.

[8] J. Preskill, "Lecture notes for physics 229: Quantum information and computation," *California Institute of Technology*, vol. 16, 1998.

[9] J. A. Smolin and D. P. DiVincenzo, "Five two-bit quantum gates are sufficient to implement the quantum fredkin gate," *Physical Review A*, vol. 53, no. 4, pp. 28–55, 1996.

[10] M. Perkowski, M. Lukac, P. Kerntopf, M. Pivtoraiko, M. Folgheraiter, Y. W. Choi, J.-w. Kim, D. Lee, W. Hwangbo, and H. Kim, "A hierarchical approach to computer-aided design of quantum circuits," in *6th International Symposium on Representations and Methodology of Future Computing Technologies*, Trier, Germany, 2003, pp. 201–209.

[11] M. D. Price, S. S. Somaroo, A. E. Dunlop, T. F. Havel, and D. G. Cory, "Generalized methods for the development of quantum logic gates for an nmr quantum information processor," *Physical Review A*, vol. 60, no. 4, pp. 27–77, 1999.

[12] M. Price, S. Somaroo, C. Tseng, J. Gore, A. Fahmy, T. Havel, and D. G. Cory, "Construction and implementation of nmr quantum logic gates for two spin systems," *Journal of Magnetic Resonance*, vol. 140, no. 2, pp. 371–378, 1999.

[13] J. Kim, J.-S. Lee, S. Lee, and C. Cheong, "Implementation of the refined deutsch-jozsa algorithm on a three-bit nmr quantum computer," *Physical Review A*, vol. 62, no. 2, p. 022312, 2000.

[14] J. Kim, J.-S. Lee, and S. Lee, "Implementing unitary operators in quantum computation," *Physical Review A*, vol. 61, no. 3, p. 032312, 2000.

[15] J.-S. Lee, Y. Chung, J. Kim, and S. Lee, "A practical method of constructing quantum combinational logic circuits," *arXiv preprint quant-ph/9911053*, 1999.

[16] P. Picton, "A universal architecture for multiple-valued reversible logic," *MVL Journal*, vol. 5, pp. 27–37, 2000.

[17] ——, "Optoelectronic multi-valued conservative logic," *International Journal of Optical Computing*, vol. 2, pp. 19–29, 1991.

[18] W. C. Athas and L. Svensson, "Reversible logic issues in adiabatic cmos," in *Proceedings of IEEE Workshop on Physics and Computation, PhysComp'94.* IEEE, 1994, pp. 111–118.

[19] R. C. Merkle, "Reversible electronic logic using switches," *Nanotechnology*, vol. 4, no. 1, p. 21, 1993.

[20] ——, "Two types of mechanical reversible logic," *Nanotechnology*, vol. 4, no. 2, p. 114, 1993.

[21] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2317–2330, 2006.

[22] V. V. Shende, A. K. Prasad, I. L. Markov, and J. P. Hayes, "Synthesis of reversible logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 6, pp. 710–722, 2003.

[23] G. Yang, X. Song, W. N. Hung, and M. A. Perkowski, "Bi-directional synthesis of 4-bit reversible circuits," *The Computer Journal*, vol. 51, no. 2, pp. 207–215, 2008.

[24] 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 Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 2, no. 4, pp. 277–293, 2006.

[25] J. Donald and N. K. Jha, "Reversible logic synthesis with fredkin and peres gates," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 4, no. 1, p. 2, 2008.

[26] D. Maslov and M. Saeedi, "Reversible circuit optimization via leaving the boolean domain," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 6, pp. 806–816, 2011.

[27] M. Saeedi, M. S. Zamani, M. Sedighi, and Z. Sasanian, "Reversible circuit synthesis using a cycle-based approach," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 6, no. 4, p. 13, 2010.

[28] D. M. Miller, D. Maslov, and G. W. Dueck, "A transformation based algorithm for reversible logic synthesis," in *Proceedings of IEEE Design Automation Conference, 2003.*, 2003, pp. 318–323.

[29] M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, C. H. Yu, K. Chung, H. Jeech, B.-G. Kim, and Y.-D. Kim, "Evolutionary approach to quantum and reversible circuits synthesis," *Artificial Intelligence Review*, vol. 20, no. 3-4, pp. 361–417, 2003.

[30] M. Mohammadi and M. Eshghi, "Heuristic methods to use don't cares in automated design of reversible and quantum logic circuits," *Quantum Information Processing*, vol. 7, no. 4, pp. 175–192, 2008.

[31] M. Mohamadi, M. Eshghi, and K. Navi, "Optimizing the reversible full adder circuit," *IEEE EWDTS, Yerevan*, pp. 312–315, 2007.

[32] D. Große, R. Wille, G. W. Dueck, and R. Drechsler, "Exact multiple-control toffoli network synthesis with sat techniques," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 5, pp. 703–715, 2009.

[33] ——, "Exact synthesis of elementary quantum gate circuits for reversible functions with don't cares," in *38th International Symposium on Multiple Valued Logic (ISMVL)*, 2008, pp. 214–219.

[34] J. Bruce, M. A. Thornton, L. Shivakumaraiah, P. Kokate, and X. Li, "Efficient adder circuits based on a conservative reversible logic gate," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, 2002, pp. 83–88.

[35] M. H. Khan and M. A. Perkowski, "Quantum ternary parallel adder/subtractor with partially-look-ahead carry," *Journal of Systems Architecture*, vol. 53, no. 7, pp. 453–464, 2007.

[36] H. Thapliyal and N. Ranganathan, "Design of reversible sequential circuits optimizing quantum cost, delay, and garbage outputs," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 6, no. 4, pp. 1–4, 2010.

[37] T. Sasao and H. Terada, "Multiple-valued logic and the design of programmable logic arrays with decoders," in *Proceedings of the 9th International Symposium on Multiple-Valued Logic*, Bath, England, May 1979, pp. 27–37.

[38] T. Sasao, "On the optimal design of multiple-valued plas," *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 582–592, 1989.

[39] T. Sasao and P. Besslich, "On the complexity of mod-2l sum pla's," *IEEE Transactions on Computers*, vol. 39, no. 2, pp. 262–266, 1990.

[40] T. Sasao, "Input variable assignment and output phase optimization of pla's," *IEEE Transactions Computers*, vol. 33, no. 10, pp. 879–894, 1984.

[41] A. Weinberger, "High-speed programmable logic array adders," *IBM Journal of Research and Development*, vol. 23, no. 2, pp. 163–178, 1979.

[42] T. Sasao, "Exmin2: a simplification algorithm for exclusive-or-sum-of-products expressions for multiple-valued-input two-valued-output functions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 5, pp. 621–632, 1993.

[43] A. Mishchenko and M. Perkowski, "Logic synthesis of reversible wave cascades," in *International Workshop on Logic Synthesis*, Louisiana, USA, June 2002, pp. 197–202.

[44] D. Maslov and G. W. Dueck, "Reversible cascades with minimal garbage," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 11, pp. 1497–1509, 2004.

[45] ——, "Garbage in reversible design of multiple output functions," in *6th International Symposium on Representations and Methodology of Future Computing Technologies*, Trier, Germany, 2003, pp. 162–170.

[46] A. R. Chowdhury, R. Nazmul *et al.*, "A new approach to synthesize multiple-output functions using reversible programmable logic array," in *5th IIEEE nternational Conference on Embedded Systems and Design and 19th International Conference on VLSI Design*, Hyderabad, India, January 2006, pp. 6–11.

[47] R. Rahman, L. Jamal, and H. M. H. Babu, "Design of reversible fault tolerant programmable logic arrays with vector orientation," *International Journal of Information and Communication Technology Research*, vol. 1, no. 8, 2011.

[48] S. K. Mitra, L. Jamal, M. Kaneko, and H. M. Hasan Babu, "An efficient approach for designing and minimizing reversible programmable logic arrays," in *Proceedings of the ACM Great Lakes symposium on VLSI*. Salt Lake City, Utah, USA: ACM, 2012, pp. 215–220.

[49] S. S. M.M.A. Polash, "Design of a lut-based reversible field programmable gate array." *Journal of Computing*, vol. 2, pp. 103–108, 2010.

[50] A. S. M. Sayem, M. M. A. Polash, and H. M. H. Babu, "Design of a reversible logic block of field programmable gate array," in *Silver Jubilee Conference on*

Communication Technologies and VLSI Design (CommV'09), TamilNadu, India, October 2009.

[51] A. S. M. Sayem and S. K. Mitra, "Efficient approach to design low power reversible logic blocks for field programmable gate arrays," in *IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, vol. 4, Shanghai, China, June 2011, pp. 251–255.

[52] M. Shamsujjoha, H. M. H. Babu, and L. Jamal, "Design of a compact reversible fault tolerant field programmable gate array: A novel approach in reversible logic synthesis," *Microelectronics journal*, vol. 44, no. 6, pp. 519–537, 2013.

[53] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3, pp. 219–253, 1982.

[54] M.-L. Chuang and C.-Y. Wang, "Synthesis of reversible sequential elements," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 3, no. 4, p. 4, 2008.

[55] R. P. Feynman, "Quantum mechanical computers," *Foundations of physics*, vol. 16, no. 6, pp. 507–531, 1986.

[56] A. Peres, "Reversible logic and quantum computers," *Physical review A*, vol. 32, no. 6, pp. 32–66, 1985.

[57] H. Thapliyal and A. P. Vinod, "Design of reversible sequential elements with feasibility of transistor implementation," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Louisiana, USA, May 2007, pp. 625–628.

[58] M. Haghparast and K. Navi, "A novel reversible bcd adder for nanotechnology based systems," *American Journal of Applied Sciences*, vol. 5, no. 3, pp. 282–288, 2008.

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

[60] S. Lee, S.-J. Lee, T. Kim, J.-S. Lee, J. Biamonte, and M. Perkowski, "The cost of quantum gate primitives." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 12, 2006.

[61] W. 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 Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 9, pp. 1652–1663, 2006.

[62] D. Maslov, C. Young, D. M. Miller, and G. W. Dueck, "Quantum circuit simplification using templates," in *Proceedings of the IEEE Computer Society conference on Design, Automation and Test in Europe (DATE '05)*, vol. 2, Munich, Germany, March 2005, pp. 1208–1213.

[63] M. Mohammadi and M. Eshghi, "On figures of merit in reversible and quantum logic designs," *Quantum Information Processing*, vol. 8, no. 4, pp. 297–318, 2009.

[64] S. G. Younis and T. F. Knight Jr, "Practical implementation of charge recovering asymptotically zero power cmos," in *Proceedings of the 1993 symposium on Research on integrated systems*. MIT Press, 1993, pp. 234–250.

[65] S. G. Younis, "Asymptotically zero energy computing using split-level charge recovery logic." Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab, Tech. Rep., 1994.

[66] J. Lim, K. Kwon, and S.-I. Chae, "Reversible energy recovery logic circuit without non-adiabatic energy loss," *Electronics Letters*, vol. 34, no. 4, pp. 344–345, 1998.

[67] J. Lim, D.-G. KIM, and S.-I. Chae, "Reversible energy recovery logic circuits and its 8-phase clocked power generator for ultra-low-power applications," *IEICE transactions on electronics*, vol. 82, no. 4, pp. 646–653, 1999.

[68] 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*, vol. 35, no. 6, pp. 865–875, 2000.

[69] S. Bandyopadhyay, A. Balandin, V. Roychowdhury, and F. Vatan, "Nano-electronic implementations of reversible and quantum logic," *Superlattices and Microstructures*, vol. 23, no. 3-4, pp. 445–464, 1998.

[70] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, "Online testable reversible logic circuit design using nand blocks," in *19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, Cannes, France, November 2004, pp. 324–331.

[71] H. M. H. Babu, M. I. Zaber, M. M. Rahman, and M. R. Islam, "Implementation of multiple-valued flip-flips using pass transistor logic [flip-flips read flip-flops]," in *IEEE Euromicro Symposium on Digital System Design (DSD)*. Rennes, France: IEEE, 2004, pp. 603–606.

[72] H. Thapliyal and A. P. Vinod, "Transistor realization of reversible tsg gate and reversible adder architectures," in *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Jeju, Korea (South), 2006, pp. 418–421.

[73] D. P. Vasudevan, P. K. Lala, and J. P. Parkerson, "Cmos realization of online testable reversible logic gates," in *IEEE Computer Society Annual Symposium on VLSI(ISVLSI'05)*, Tampa, Florida, USA, May 2005, pp. 309–310.

[74] S. Nowrin, P. Nazneen, and L. Jamal, "Design of a compact reversible read-only-memory with mos transistors," *International Journal of VLSI design & Communication Systems (VLSICS)*, vol. 6, no. 5, pp. 69–84, 2015.

[75] I. Grout, *Digital Systems Design with FPGAs and CPLDs —— Introduction to Programmable Logic*. Elsevier, 2008.

[76] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of field-programmable gate arrays," *Proceedings of the IEEE*, vol. 81, no. 7, pp. 1013–1029, 1993.

[77] S. C. Wong, H. So, J. H. Ou, and J. Costello, "A 5000-gate cmos epld with multiple logic and interconnect arrays," in *Proceedings of the IEEE Conference on Custom Integrated Circuits*, San Diego, California, USA, May 1989, pp. 5–8.

[78] T. Matsunaga, S. Kimura, and Y. Matsunaga, "Power and delay aware synthesis of multi-operand adders targeting lut-based fpgas," in *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design (ISLPED'11)*, Kyushu, Japan, August 2011, pp. 217–222.

[79] J. Cong and Y. Ding, "Combinational logic synthesis for lut based field programmable gate arrays," *ACM Transactions on Design Automation of Electronic Systems*, vol. 1, no. 2, pp. 145–204, 1996.

[80] N. B. Bhat and D. D. Hill, "Routable technology mapping for lut fpgas," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD '92)*, Cambridge, Massachusetts, USA, October 1992, pp. 95–98.

[81] M. Ahrens, A. El Gamal, D. Galbraith, J. Greene, S. Kaptanoglu, K. Dharmarajan, L. Hutchings, S. Ku, P. McGibney, J. McGowan *et al.*, "An fpga family optimized for high densities and reduced routing delay," in *Proceedings of the IEEE Conference Custom Integrated Circuits*, Boston, Massachusetts, USA, May 1990, pp. 31–5.

[82] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. A. El-Ayat, and A. Mohsen, "An architecture for electrically configurable gate arrays," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 2, pp. 394–398, 1989.

[83] J. a. Birkner, A. Chan, H. Chua, A. Chao, K. Gordon, B. Kleinman, P. Kolze, and R. Wong, "A very-high-speed field-programmable gate array using metal-to-metal antifuse programmable elements," *Microelectronics Journal*, vol. 23, no. 7, pp. 561–568, 1992.

[84] Plessey, "Plessey semiconductor era60100," Preliminary data sheet, 1989.

[85] V. Betz and J. Rose, "How much logic should go in an fpga logic block," *IEEE Design & Test of Computers*, vol. 15, no. 1, pp. 10–15, 1998.

[86] R. J. Francis, "A tutorial on logic synthesis for lookup-table based fpgas," in *Proceedings of the 1992 IEEE/ACM international conference on Computer-aided design (ICCAD '92)*, Santa Clara, California, USA, 1992, pp. 40–47.

[87] H. Gru, R. Hartenstein *et al.*, "Chameleon: A workstation of a different color," *Field-Programmable Gate Arrays: Architectures and Tools for Rapid Prototyping*, pp. 152–161, 1993.

[88] V. Betz and J. Rose, "Cluster-based logic blocks for fpgas: Area-efficiency vs. input sharing and size," in *Proceedings of the IEEE Conference on Custom Integrated Circuits (CICC '97)*, Santa Clara, California, USA, may 1997, pp. 551–554.

[89] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, and B. Massey, "Regularity and symmetry as a base for efficient realization of reversible logic circuits," in *International workshop on logic synthesis*, Tahoe City, USA, 2001, pp. 245–252.

[90] M. H. Khan and M. Perkowski, "Multi-output esop synthesis with cascades of new reversible gate family," in *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technologies*, Trier, Germany, 2003.

[91] ——, "Logic synthesis with cascades of new reversible gate families," in *6th International Symposium on Representations and Methodology of Future Computing Technology,*, Trier, Germany, March 2003, pp. 43–55.

[92] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products," in *International Reed-Muller Workshop*, Mississippi, USA, August 2001, pp. 242–250.

[93] "Microwind dsch - schematic editor and digital simulator," http://www.microwind.net/dsch.ph, (Accessed: 2017-02-30).

[94] "Microwind2.exe," http://www.microwind.net/microwind2.exe, (Accessed: 2017-02-30).

[95] H. Thapliyal and M. Zwolinski, "Reversible logic to cryptographic hardware: a new paradigm," in *49th IEEE International Midwest Symposium on Circuits and Systems*, Puerto Rico, USA, August 2006, pp. 342–346.

[96] K. Buch, "Low power fault tolerant state machine design using reversible logic gates," in *Conference on Military and Aerospace Programmable Logic Devices*, Maryland, USA, September 2008.

[97] S. N. Mahammad and K. Veezhinathan, "Constructing online testable circuits using reversible logic," *IEEE transactions on instrumentation and measurement*, vol. 59, no. 1, pp. 101–109, 2010.

[98] M. Tayari and M. Eshghi, "Design of 3-input reversible programmable logic array," *Journal of Circuits, Systems, and Computers*, vol. 20, no. 02, pp. 283–297, 2011.

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

[100] S. Gugnani and A. Kumar, "Synthesis of reversible multiplexers," *International Journal of Scientific & Engineering Research*, vol. 4, no. 7, pp. 1859–1863, 2013.

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

[102] J. E. Rice, "An introduction to reversible latches," *The Computer Journal*, vol. 51, no. 6, pp. 700–709, 2008.

[103] M. S. Al Mamun, I. Mandal, and M. Hasanuzzaman, "Design of universal shift register using reversible logic," *International Journal of Engineering and Technology*, vol. 2, no. 9, pp. 1620–1625, 2012.

[104] J. E. Rice, "A new look at reversible memory elements." in *IEEE International Symposium on Circuits and Systems (ISCAS'06)*, Island of Kos, Greece, May, 2006.

[105] S. D. Kumar and S. N. Mahammad, "A novel sram cell design using reversible logic," in *3rd IEEE International Conference on Eco-friendly Computing and Communication Systems (ICECCS 2014)*. Mangalore,India: IEEE, December 2014, pp. 1–4.

[106] M. Morrison, M. Lewandowski, R. Meana, and N. Ranganathan, "Design of static and dynamic ram arrays using a novel reversible logic gate and decoder," in *11th IEEE Conference on Nanotechnology*, Oregon, USA, 2011, pp. 417–420.

Appendix A

Various Existing Reversible gates

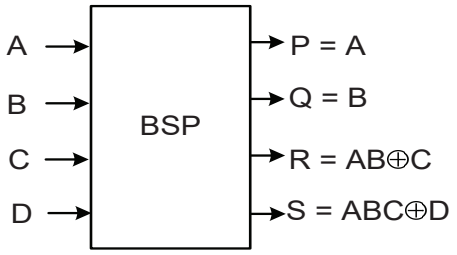
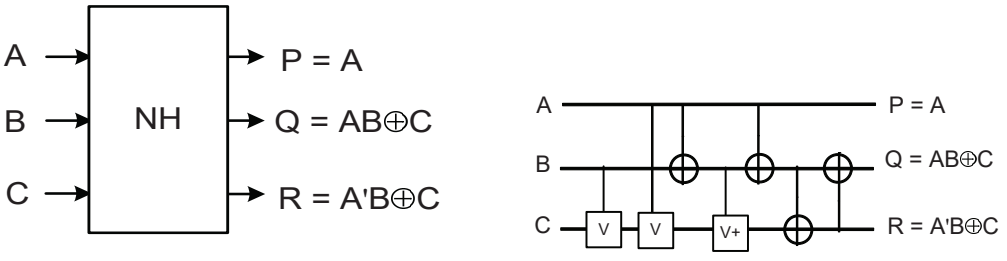


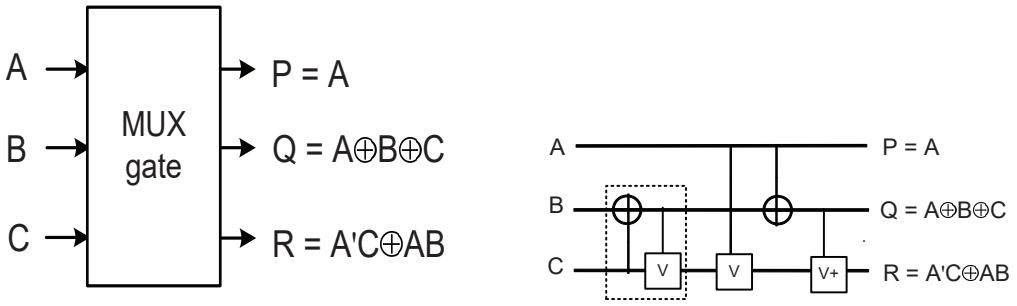
FIGURE A.1: Block diagram of BSP gate



(a) Block diagram.

(b) Quantum circuit.

FIGURE A.2: NH gate



(a) Block diagram. (b) Quantum circuit.

Figure A.3: MUX gate

Appendix B

Transistor Realization of Proposed Reversible gates

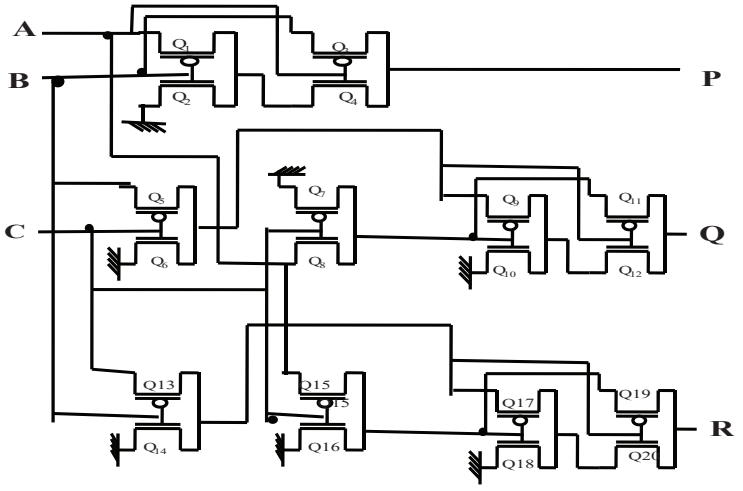


FIGURE B.1: Transistor realization of the proposed TB gate

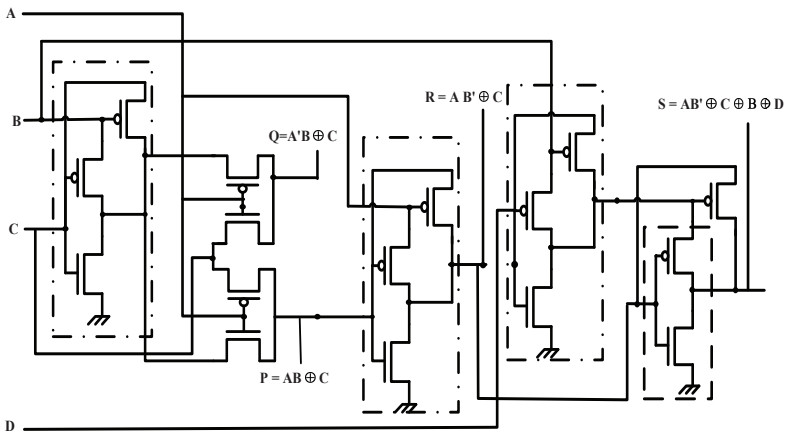


Figure B.2: Transistor realization of the proposed HND gate

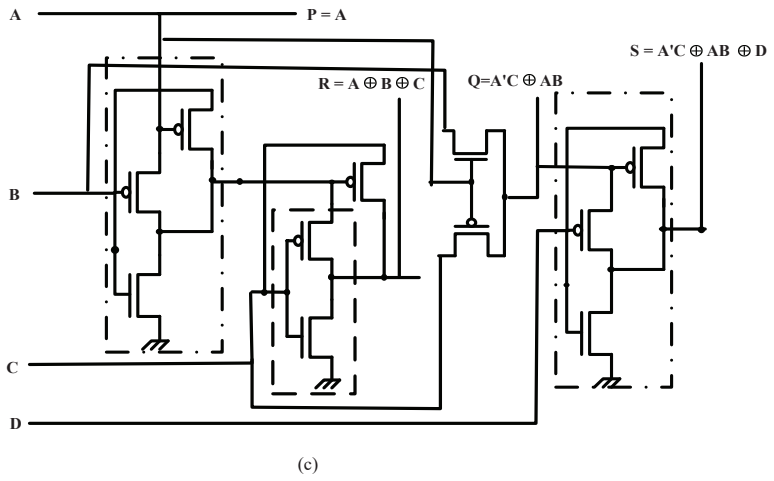


Figure B.3: Transistor realization of the proposed HNF gate

Appendix C

List of Acronyms

AND	Boolean operation (&, concatenation)
	with properties 0&0 = 0&1 = 1&0 = 0,1&1 = 1;
CNOT	Controlled NOT gate, also known as the Feynman gate;
Ex-OR	Boolean operation ($\oplus$) with properties $0 \oplus 0 = 1 \oplus 1 = 0$, $0 \oplus 1 = 1 \oplus 0 = 1$;
ESOP	Exclusive OR Sum-of-Products;
FG	Feynman gate
F2G	Feynman double gate
FPGA	Field Programmable Gate Array;
FRG	Fredkin gate
HND	Hafiz-Naz-Decoder
HNF	Hafiz-Naz-Flip-Flop
LUT	Look-Up table;
MFRG	Modified Fredkin gate
MPGA	Mask Programmable Gate Array;
MTSG	Modified TSG gate
TB	Tara-Babu
NOT	Boolean operation ($'$) with properties $(0)' = 1$, $(1)' = 0$;
PG	Peres gate
PLA	Programmable Logic Array;

RPGA	Reversible Programmable Gate Array;
RPLA	Reversible Programmable Logic Array;
RAM	Random Access Memory;
TG	Toffoli gate

Appendix D

List of Publications

International Journal Papers (SCI-indexed)

1. Nazma Tara, Hafiz Md. Hasan Babu and Lafifa Jamal, "Power Efficient Optimum Design of the Reversible Plessey Logic Block of a Field-Programmable Gate array", Elsevier Journal of Sustainable Computing: Informatics and Systems, volume 16, pp 76-92, December 2017.

2. Nazma Tara and Hafiz Md. Hasan Babu, "Synthesis of Reversible PLA Using Products Sharing", Springer Journal of Computational Electronics, Volume 15, Issue 2, pp. 420-428, June 2016.

3. Zarrin Tasnim Sworna, Mubin Ul Haque, Nazma Tara, Hafiz Md Hasan Babu and Ashis Kumar Biswas. "Low-power and area efficient binary coded decimal adder design using a look up table-based field programmable gate array", IET Circuits, Devices & Systems, Volume 10, Issue 3, pp. 163-172, 2016.

International Conference Papers

1. Nazma Tara, Hafiz Md Hasan Babu, and Nawshi Matin, "Logic Synthesis in Reversible PLA." 29th International Conference on VLSI Design and 15th International Conference on Embedded Systems (VLSID), Kolkata, India, pp.110-115, January 2016.

2. Nazma Tara, Lafifa Jamal and Hafiz Md. Hasan Babu, "An Efficient Approach to Design Compact Reversible Programmable Logic Array", IEEE Woman in Engineering Conference on Electrical and WIECON-ECE 2015, Dhaka, Bangladesh, pp. 135-138, December 2015.

3. Ankur Sarker, Tanvir Ahmed, S. M. Mahbubur Rashid, Shahed Anwar, Lafifa Jamal, Nazma Tara, Md. Masbaul Alam, and Hafiz Md. Hasan Babu, "Realization of Reversible logic in DNA computing", 11th IEEE conference on Bioinformatics and Bioengineering, Taichung, Taiwan, pp. 261-265, 2011.