School of Electrical Engineering and Computing

Department of Electrical and Computer Engineering

Design and Synthesis of Reversible Logic

Chua Shin Cheng

This thesis is presented for the Degree of

Doctor of Philosophy

of

Curtin University

March 2016

DECLARATION

To the best of my knowledge and belief, this thesis contains no material previously published by any other person except where due acknowledgement has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature : ______________________

(CHUA SHIN CHENG)

Date : March 2016

ABSTRACT

Power dissipation in modern technologies plays an important role and overheating is a serious issue to consider for both manufacturers and consumers. In the near future, manufacturer would face difficulty in implementing more advanced technology due to the limited operating temperature range for smaller devices such as smartphone, tablet and wearable devices. This issue is mainly caused by the use of the irreversible logic system in current technology devices which each logic gate dissipates heat during computations. To overcome this issue, new logic system, namely reversible logic has been investigated and has been identified to be one of the most promising solutions.

Reversible logic is an emerging research area and has been historically motivated by its theoretical research in low-power computing devices. Recently, it has attracted interest as part of components in these fields: quantum computing, optical computing and nanotechnology. However due to its intrinsic properties, design and synthesis methods used for traditional irreversible logic cannot be carried out on reversible logic. Therefore, new circuit designs and synthesis algorithms are being developed to incorporate the reversible logic based applications.

For a circuit to be reversible, the inputs and outputs of the circuit have to form a unique one-to-one mapping. At times, constant ancilla inputs and garbage outputs are added to create that unique one-to-one mapping to keep the circuit reversible. Parameters such as ancilla input, garbage output, quantum cost and delay play an important role in the evaluation of reversible circuits. Thus for any reversible logic design, it is important to minimize these parameters to achieve better efficiency. To design reversible synthesis algorithm, majority of the benchmark circuits have been optimized in terms of ancilla input and garbage output, hence minimizing gate count and quantum cost are the current focus in the field of reversible logic.

Although the research of reversible logic has achieved major breakthrough in design and synthesis, but various problems still exist and solution remains elusive. The biggest challenges face in the synthesis of reversible logic is that the existing synthesis algorithm is not able to full balance the gate count and quantum cost to achieve optimal solution for all synthesized reversible function. Recently with the new quantum realization of the mixed-polarity Toffoli gate, synthesis algorithm is able to achieve one step further in scaling reversible function better and achieves close optimal result. Although several research works using this gate has been proposed, but further research can still be done to improve existing algorithms. For the research in designing reversible logic based arithmetic application, research attention has always been focused only on quantum cost, ancilla input and garbage output. Parameter such as propagation delay which is crucial in achieving fast executing time which is often neglected. As fast arithmetic is crucial it should not be neglected in any high speed arithmetic design. Thus these issues are addressed towards the finding of this research.

The first contribution of this dissertation is the design of a reversible logic synthesis algorithm. The proposed synthesis algorithm uses search-based technique that synthesizes reversible function into reversible circuits using NCT (NOT, CNOT, Toffoli) based library gates with positive and negative control lines. As taken into account, the proposed synthesis algorithm is designed to provide better quantum cost scaling which most existing synthesis algorithms fail to take care of. As per literature review, this is the first attempt to integrate Toffoli gate with mixed polarity control capability into search-based synthesis algorithm. Through the synthesis results obtained, it is concluded that the proposed algorithm is able to achieve close optimal result using the mixed-polarity Toffoli gate for all three and four variables based reversible functions.

The second contribution is the novel design of a new reversible gate which is developed to serve as a reversible full adder circuit. The proposed reversible gate mainly targets to overcome garbage output issue faced by existing designs while keeping other important parameters equal as compared to the existing designs. The proposed reversible full adder is designed using a mixture of various techniques. It is first designed using our proposed synthesis algorithm. The result obtained is then further optimized in term of quantum cost and propagation delay until its final form is achieved. Experimental result has shown that the proposed reversible full adder design outperforms existing design in terms of ancilla input, garbage output, quantum cost and propagation delay. As full adder circuit is required in almost all digital arithmetic design, the proposed reversible full adder circuit will find its application in corresponding area.

The third contribution is the design of minimal delay reversible BCD adder. The reversible BCD adders are designed to delay the need of the carry input C_{in} and generate the required carry output C_{out} for the next BCD digit faster. This method allows simultaneous run of multiple BCD digits and achieves high speed BCD sum. This objective behind this method is to overcome the poor propagation delay exist in existing reversible BCD adder designs. As fast executing time is essential for arithmetic unit, propagation delay should not be neglected. Hence the proposed BCD adder designs will be beneficial in reversible arithmetic area requiring fast BCD sum. Experimental result has shown that the proposed BCD adder designs outperform existing designs in term of propagation delay while maintaining other important parameter at reasonable range.

The last contribution in this dissertation is the design of low quantum cost reversible BCD adders. As quantum cost remain as the most important parameter to consider for all reversible logic designs. Applications with lower quantum cost value are likely to be more advantageous over the others. As research on reversible logic continues to advance, better tools and reversible gates are available to help achieving lower quantum cost value. Hence a reversible BCD adder design with low quantum cost value is proposed. The BCD adders are designed with the combination of the proposed reversible full adder circuit and other reversible gate such as Peres and TR gate to achieve best quantum cost value. Parallelism technique has been applied to keep propagation delay at minimal. Experimental results are shown that the proposed BCD adder designs have less quantum cost value than other existing designs.

ACKNOWLEDGEMENT

I would like to give thanks to God for providing me the opportunity to pursue my doctoral studies. Throughout the time He constantly reminds me to walk faithfully in His path. When I was weak, He constantly provides me with strength and wisdom to overcome any obstacle faced. Thank you for being there with me throughout my darkest time.

To my dear parents thank you for your unfailing love and kind understanding that make me feel blessed and proud to have them. Thank you for supporting me and being there with me always.

I would like to express my deepest gratitude to my supervisors Professor Ashutosh Kumar Singh, Dr Lenin Gopal and Dr Amandeep S. Sidhu for their valuable advices and guidance throughout the years of my research. Without their continuous support and guidance, I would not have gotten so far.

In addition, I would like to thank Curtin University Sarawak Campus for providing me with the scholarship and facilities. Without it, I would not have the opportunity to take up the doctoral program.

Last but not least, I would like to thank all my friends whom I met and share good moments together throughout the time. I would also like to thank my church especially members from Curtin Ministry (Aunty, Uncle and friends) for their great companion, encouragements, prayers and support. Thank you very much.

PUBLICATIONS

Journal Papers

1. S. C. Chua and A. K. Singh, "Heuristic Synthesis of Reversible Logic–A Comparative Study," Advances in Electrical and Electronic Engineering, vol. 12, pp. 210-225, 2014.

2. S. C. Chua and A. K. Singh, "Design of Quick Reversible Binary Coded Decimal (BCD) Adder," International Journal of Electronics, 2015. (Under review, ISI indexed)

3. S. C. Chua and A. K. Singh, "Search-Based Reversible Logic Synthesis Using Mixed-Polarity Toffoli Gate," Turkish Journal of Electrical Engineering & Computer Sciences, 2015. (Under review, ISI indexed)

4. S. C. Chua and A. K. Singh, "Design of Efficient Reversible BCD Adder Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015. (Under review, ISI indexed)

Conference Papers

1. S. C. Chua, A. K. Singh, and L. Gopal, "Efficient Three Variables Reversible Logic Synthesis Using Mixed-polarity Toffoli Gate," Procedia Computer Science, vol. 70, pp. 362-368, 2015.

2. S. C. Chua, L. Gopal, A. S. Sidhu, and A. K. Singh, "Design of Low Quantum Cost Reversible BCD Adder," in Proceedings of the 5th IEEE International Conference On Control System, Computing and Engineering (ICCSCE2015), Penang, Malaysia, 2015.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

BCD	Binary Coded Decimal
BDD	Binary Decision Diagram
CNOT	Controlled-NOT
DD	Decision Diagram
ESOP	Exclusive OR sum-of-product
FA	Full Adder
GC	Gate count
HA	Half Adder
HD	Hamming Distance
MMD	Miller-Maslov-Miller
m-NCT	Mixed polarity NOT, CNOT, Toffoli based gate
NCT	NOT, CNOT, Toffoli based gate
NCTP	NOT, CNOT, Toffoli, Peres, inverse-Peres based gate
NCTSF	NOT, CNOT, Toffoli, SWAP, Fredkin based gate
PPRM	Positive Polarity Reed-Muller
QC	Quantum cost
RM	Reed-Muller
VHDL	Very High Speed Integrated Circuit Hardware Description Language
XOR	Exclusive OR

CHAPTER 1 INTRODUCTION

Energy lost during computation is an important issue addressed in every low power digital design. According to Landauer [1], every bit of information is lost each time a logic gate performs a logic operation[1]. This information lost is converted into a form of heat and it is around

$$kTln(2) \text{ Joule} \tag{1.1}$$

Where k is the Boltzmann constant and T is the absolute temperature [1]. At room temperature, the heat generated is not significant, but still it cannot be neglected [2]. Today, the amount of information loss in the form of heat to the surrounding environment of a chip leads to immense challenges in circuit design [3]. As Moore's law predicts the exponential growth of transistor in an integrated system, the heat generated due to the irreversible system used will be noticeable as more transistors are implemented in a die [4, 5]. However according to Bennett [6], it has been shown that the finite amount of energy loss can be avoided in a fully reversible logic implementation because information lost does not occur in the operation of reversible logic.

Reversible logic is specially designed logic gate which utilize quantum logic theories in realizing its logic operation [7]. Since information in reversible logic is fully permutable from its inputs to outputs and vice versa, it achieves lossless information during this transition. Thus this makes reversible logic essential in the future of circuit design. One of the most interesting applications of reversible logic is in the application of quantum computing [8]. A quantum computer is known to be able to solve exponentially hard problems in polynomial time [8-10]. Quantum computer can be viewed as a quantum network composed of quantum logic gates in which each quantum logic gate performs an elementary unitary operation called qubits [11-13]. All unitary operations are reversible and hence quantum networks must be built using reversible logical components [14]. Thus the research on reversible logic is beneficial to the development of future quantum technologies.

Quantum technologies are not the only sectors that will benefit from reversible logic. There are other technologies where reversible logic can be applied such as in adiabatic CMOS design [15, 16], optical computing [17-19], nanotechnology [20-23], etc. Further discussion on these different technologies will not be addressed in this dissertation because each of them involves a vast amount of expertise. However, the irreversible logic implementation is common between these technologies.

The existing logic gate used in digital designs such as the AND gate, OR gate, XOR gate, etc. are irreversible. In order for a circuit to be reversible, there has to be a unique one-to-one mapping between the inputs and outputs [24]. Consider an example of a common XOR gate shown in Figure 1.1 (a), it is obvious that the logic gate is not reversible because there is no unique one-to-one mapping between the inputs and outputs. This is because for the input argument 00 and 11 it give rise to the output 0 while input argument 01 and 10 give rise to the output 1 as seen in Table 1-1 (a). Figure 1.1 (b) shows a reversible XOR gate which has a unique one-to-one mapping between the inputs and outputs such that each of its output vector is assigned to a unique input argument as shown in Table 1-1 (b).

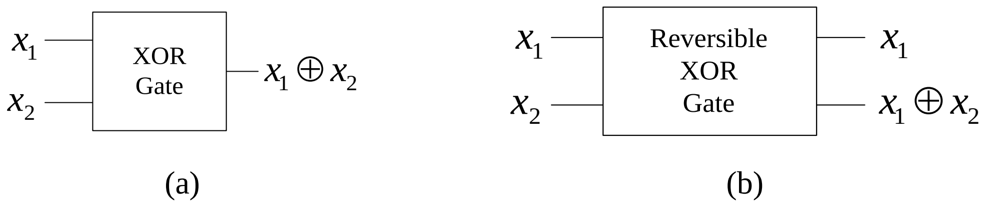


Figure 1.1. (a) Conventional XOR gate (b) Reversible XOR gate

Table 1-1. (a) Truth table for conventional XOR gate (b) Truth table for reversible XOR gate

(a)

	Input x_1, x_2	Output y
0	0 0	0
1	0 1	1
2	1 0	1
3	1 1	0

(b)

	Input x_1, x_2	Output y_1, y_2
0	0 0	0 0
1	0 1	0 1
2	1 0	1 1
3	1 1	1 0

After the discovery of powerful quantum algorithms in the mid-1990s, research on reversible logic synthesis has begun to attract attention [7]. The research on reversible logic is expanding towards two different directions which are design and synthesis. Over these years, lot of interesting synthesis algorithm has been proposed. The earlier proposed synthesis algorithms synthesize reversible function into a set of

NCT based gate library which consist only of NOT, CNOT and Toffoli gates [25-27]. Over the years, other reversible gates such as SWAP, Fredkin, Peres and inverse-Peres gates have also been used in several synthesis algorithms [28-30]. However, most synthesis algorithms prefer to use the original NCT gate library with extended Toffoli size such as using Toffoli4, Toffoli5 or higher Toffoli size to cope with larger functions. Toffoli gate has been explored and extended and recently researchers have started to integrate Toffoli gate with positive and negative control lines into its operation [31, 32]. This type of Toffoli gate with positive and negative polarity control capability is known as the mixed-polarity Toffoli gate. Usage of this gate for reversible logic synthesis has been extensively studied in a number of works such as in [33-36] and had been proven that it provides better synthesis results. Although various research works using this gate have been proposed, still there is room for improvement. Besides, the usage of this gate for designing reversible arithmetic unit, reversible Binary Coded Decimal (BCD) adder has not yet been reported in any literature work. Thus this leads to the contribution of this research.

1.1 Thesis Objective

The aim of this research is to propose efficient methods for design and synthesis of reversible circuits using reversible gates. The key objectives of the research are to:

- Develop new reversible logic synthesis algorithm in terms of gate count and quantum cost using a set of benchmarking circuit.
- Investigate much efficient reversible gates and use it to design new reversible logic arithmetic applications.
- Optimize the performance of the proposed reversible adder circuits in terms of propagation delay.
- Address the usage of mixed-polarity Toffoli gate for designing reversible logic synthesis algorithm and reversible BCD adders.

1.2 Problem Statement

Although the research of reversible logic has achieved major breakthrough in design and synthesis, but various problems still exist and solution remains elusive. The biggest challenges face in the synthesis of reversible logic is that the existing synthesis algorithm is not able to full balance the gate count and quantum cost to achieve optimal solution for all synthesized reversible function. Recently with the new quantum realization of the mixed-polarity Toffoli gate, synthesis algorithm is able to achieve one step further in scaling reversible function better and achieves close optimal result. Although several research works using this gate has been proposed, but further research can still be done to improve existing algorithms. For the research in designing reversible logic based arithmetic application, research attention has always been focused only on quantum cost, ancilla input and garbage output. Parameter such as propagation delay which is crucial in achieving fast executing time which is often neglected. As fast arithmetic is crucial it should not be neglected in any high speed arithmetic design. Thus these issues are addressed towards the finding of this research.

1.3 Thesis Contributions and Overview

In the design and synthesis of reversible logic, there are a few important points that need to be kept in mind. To further understand the efficiency of reversible logic synthesis algorithm, gate count which denotes the amount of reversible gate required to realize a reversible function and quantum cost which denotes the effort to represent the gate function in quantum realization are the most significant parameters. The efficiency of a synthesis algorithm is determined by these two parameters tested out by synthesizing all three variables based reversible functions and reversible benchmark functions [37-39].

For designing reversible logic circuit based applications, the important parameters to be considered are the ancilla inputs (constant logic input), garbage outputs (unused outputs), quantum cost and propagation delay. These parameters are required to keep at possible minimal. Keeping all parameter at lowest is not possible, because optimizing one parameter may often result in the degradation of other important parameters. Hence while designing and synthesizing the reversible logic circuits, some parameters should be considered to achieve optimal value with expenses of other parameters.

In this dissertation, the research contributions toward the design and synthesis of reversible logic are as follows:

- The first contribution of this dissertation is the design of a search-based reversible logic synthesis algorithm. The proposed synthesis algorithm synthesizes reversible function into reversible circuits using positive and negative control Toffoli gates. Based on the recent literature work, it can be concluded that this is the first attempt in the reversible logic circuit design to integrate Toffoli gate with mixed polarity control capability into search-based synthesis algorithm. It has the capability to allow the algorithm to synthesize reversible function in term of quantum cost which is a challenging task that most synthesis algorithms are incapable of.
- The second contribution of this dissertation is the design of a novel reversible gate. The proposed reversible gate is designed mainly to serve as a reversible full adder circuit and is proven to be efficient in terms of ancilla input, garbage output, quantum cost and delay. It can be notified that, the proposed reversible full adder circuit implementation does not use any garbage output. Ancilla input, quantum cost and delay of the proposed design are maintained equally as compared to other existing designs.
- The third contribution of this dissertation is the design of minimal delay reversible BCD adders. In the proposed reversible BCD adder design, the delay introduced between the carry input C_{in} and carry output C_{out} is reduced; hence the next BCD digit is generated faster. In the BCD adder design, parallelism technique is used to achieve the fast BCD sum.
- The last contribution of this dissertation is the design two quantum cost optimized reversible BCD adders. The proposed design uses a variety of reversible gate such as the earlier proposed reversible full adder circuit, Peres gate and TR gate in building the designs. In addition, parallelism technique has been implemented to reduce the overall delay.

The remainder of this dissertation is organized as follows:

- In Chapter 2, preliminaries on reversible logic are being discussed. It covers reversible functions and circuits, types of reversible gates and reversible function representation model. All the important parameters are explained in details to keep the chapter self-contained.
- In Chapter 3, existing literature work on synthesis of reversible logic, reversible full adder circuit and also reversible BCD adder circuit are being reviewed and discussed. Each of the topics is discussed in a new sub-section to keep the flow in order.
- In Chapter 4, a search-based reversible logic synthesis algorithm using mixed-polarity Toffoli gate has been presented. Search-based reversible logic synthesis algorithm is proven to be able to synthesize any reversible function into reversible circuit. However existing search-based synthesis algorithm are only limited to positive Toffoli gate family. In the proposed algorithm, search-based synthesis technique is integrated with mixed-polarity Toffoli gate. It can be notified that this is the first attempt to use mixed-polarity Toffoli gate for search-based reversible logic synthesis algorithm. Synthesis results are shown that the proposed synthesis algorithm is able to produce result close to the optimal solution.
- In Chapter 5, a new reversible gate is proposed which is capable of realizing the reversible full adder circuit functions efficiently as compared to the existing reversible gates. It can be noted that this is the first attempt in literature for a reversible full adder circuit realization that does not use any garbage output.
- In Chapter 6, a new class of design of reversible BCD adder is proposed with uses of parallelism technique to archive fast BCD sum. In this Chapter, two reversible BCD adder designs are presented such as reversible BCD adder without Carry Input and reversible BCD adder with Carry Input. In each of the proposed design, two reversible logic implementations are presented such as one with using mixed-polarity Toffoli gate and another one without it. The proposed designs are primarily optimized for propagation delay to achieve fast BCD sum. Other important parameters such as the ancilla inputs, garbage outputs and quantum cost are designed at best possible lowest values.
- In Chapter 7, two reversible logic implementations of BCD adder designs are optimized in terms of quantum cost. Quantum cost has always been the most

important parameter to consider for all reversible logic design and synthesis. Other important parameters such as ancilla input, garbage output and propagation delay are not neglected and is kept at best possible lowest value.

- In Chapter 8, conclusions of the research findings are drawn and possible future works and interesting research ideas are outlined to improve or extend the current research works.

CHAPTER 2 BACKGROUND

To keep the dissertation self-contained, basic preliminaries on reversible logic are discussed in this chapter. Section 2.1 and Section 2.2 discuss about the fundamental theories on reversible logic such as reversible function, reversible circuit, ancilla input and garbage output. Section 2.3 provides a detailed explanation on reversible gates and addressed all reversible gates used in this dissertation. Delays on reversible gates are discussed in Section 2.4 and function representation model of reversible logic is discussed in Section 2.5.

2.1 Reversible Function

A logic function $f\,(x_1, x_2, x_3, \ldots, x_n)$ of n Boolean variables is reversible if it has a mapping of all its input assignment to a unique output assignment [40].

Example: Table 2-1 illustrates an example of a three variables reversible function. The function *f* = (1, 3, 5, 7, 6, 2, 0, 4) is permuted over {0, 1, 2, 3, 4, 5, 6, 7} where there exist a one-to-one relationship between the inputs and outputs, such that *f*(0) = 1, *f*(1) = 3, *f*(2) = 5, *f*(3) = 7, *f*(4) = 6, *f*(5) = 2, *f*(6) = 0, *f*(7) = 4. The function is reversible because each of the output is assigned to a unique input with no repeating.

Table 2-1. Reversible function

	Input A, B, C	Output P, Q, R	Permutation Function F
0	0 0 0	0 0 1	1
1	0 0 1	0 1 1	3
2	0 1 0	1 0 1	5
3	0 1 1	1 1 1	7
4	1 0 0	1 1 0	6
5	1 0 1	0 1 0	2
6	1 1 0	0 0 0	0
7	1 1 1	1 0 0	4

2.2 Reversible Circuit

A well-formed reversible circuit is an acyclic combinational logic circuit in which all gates constructed within it are reversible [41]. The usage of fan-out which connects the output of a gate to multiple inputs and looping which connects the output of a gate directly to its input is not allowed in reversible logic [42, 43]. An irreversible function can be converted into a reversible function by adding extra lines to create a one-to-one mapping between the inputs and outputs. Those added extra lines on the input side are known as the ancilla input and are supplied with constant logic value 0 or 1 [44, 45]. The lines at the output side which is neither a useful output nor a regenerated output are known as the garbage output [46, 47]. In a reversible circuit design, ancilla input and garbage output are essential components that help to keep the circuit reversible [43].

Example: To demonstrate the process of converting an irreversible function into a reversible function, logic function of a half-adder is considered in this example. Table 2-2 illustrates the function of a half-adder, where A and B are the inputs to be summed and S is the sum and C is the carry. The function is not reversible as the function contains repeated output vectors (mark with a * sign). In order for the function to be reversible, an extra line is added to create the unique one-to-one mapping between the inputs and outputs. Table 2-3 illustrates the function after extra line has been added (Input D is an ancilla input of constant value 1 and output G is the garbage output). Now the function is reversible as it has a clear one-to-one mapping between the inputs and outputs.

Table 2-2. Half-adder function

Input A, B	Output C, S	
0 0	0 0	
0 1	0 1	*
1 0	0 1	*
1 1	1 1	

Table 2-3. Possible reversible half-adder function

Input D, A, B	Output G, C, S
1 0 0	1 0 0
1 0 1	0 0 1
1 1 0	1 0 1
1 1 1	1 1 1

2.3 Reversible Gates

A reversible gate realizes a reversible function and has equal amount of input and outputs which form a unique one-to-one mapping between the inputs and outputs. If a reversible gate has k inputs and outputs, it is called a k bits gate or a gate with k lines. All reversible gates are associated with a cost called quantum cost which denotes the effort required to realize the function of the gate [31]. Quantum cost of a reversible gate is measured by the number of elementary quantum gates required to realize the function [48]. The elementary quantum gates generally consist of the NOT gate, the CNOT gate, the Controlled-V, Controlled-V+, Controlled-W and Controlled-W+ gates which are the basic building block of any reversible gate [48-50]. All elementary quantum gates have a quantum cost of 1 [48, 51].

In the sub-sections below, the commonly used reversible gates are discussed in the research field. Few well known reversible gates form the basic NCT gate library such as the NOT, CNOT, Toffoli and Toffoli4 gates. The NCT gate library with positive and negative polarity control line such as mixed-polarity Toffoli and mixed-polarity Toffoli4 gates form the m-NCT gate library when combine together with the NCT gate library. There are other commonly used reversible gate such as the SWAP gate with Fredkin gate and also the Peres gate with inverse-Peres which forms the NCTSF and NCTP gates family respectively.

2.3.1 The NOT Gate

The NOT gate is a one bit reversible gate, it has a quantum cost of 1 [52]. The gate maps a Boolean pattern x_1 to $\overline{x_1}$. Figure 2.1 shows the accepted symbol of the gate with its function.

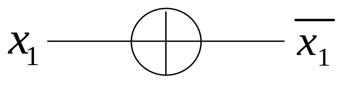


Figure 2.1. NOT gate

2.3.2 The CNOT Gate

The CNOT gate or the Feynman gate is a two bit reversible gate, it has a quantum cost of 1 [53]. The gate maps a Boolean pattern (x_1, x_2) to $(x_1,\ x_1 \oplus x_2)$. Figure 2.2 shows the accepted symbol of the gate with its function [52].

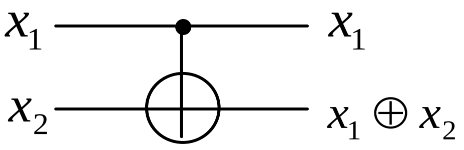
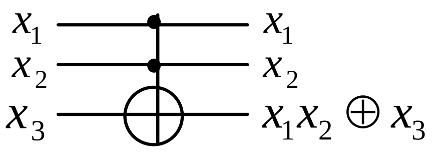


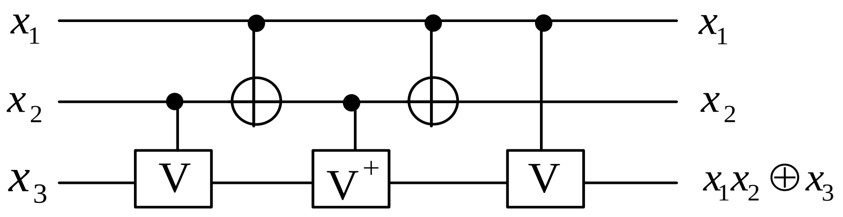
Figure 2.2. CNOT gate

2.3.3 The Toffoli Gate

The Toffoli gate is a three bit reversible gate, it has a quantum cost of 5 [52, 53]. The gate maps a Boolean pattern (x_1, x_2, x_3) to $(x_1,\ x_2,\ x_1 x_2 \oplus x_3)$. Figure 2.3 (a) shows accepted symbol with its function and Figure 2.3 (b) shows the quantum realization of the Toffoli gate [51], where V and V^+ are unitary matrices known as the Controlled-V and Controlled-V+ gate [7, 31]. Where $V = \frac{1+i}{2}\begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}$ and $V^+ = \frac{1-i}{2}\begin{bmatrix} 1 & i \\ i & 1 \end{bmatrix}$, and $V^2 = (V^+)^2 = NOT$ and V^+ represents the adjoint of V, so $V \times V^+ = V^+ \times V = I$ [7, 31, 48].



(a)

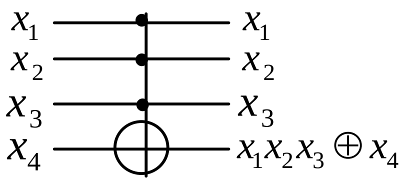


(b)

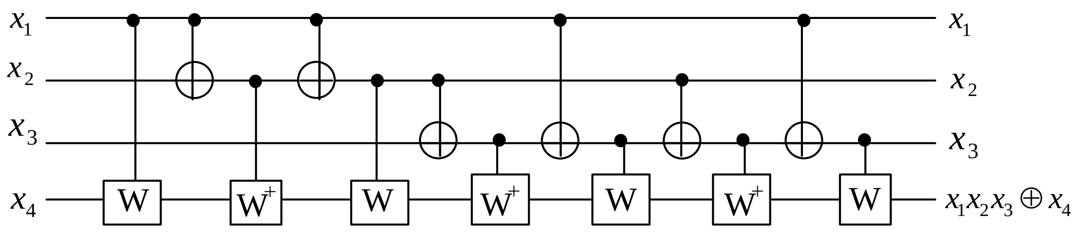
Figure 2.3. (a) Toffoli gate symbol (b) Quantum realization of the Toffoli gate

2.3.4 The Toffoli4 Gate

The Toffoli4 gate [53] is a four bit reversible gate, it has a quantum cost of 13 [52]. The gate maps a Boolean pattern (x_1, x_2, x_3, x_4) to $(x_1,\ x_2,\ x_3,\ x_1x_2x_3 \oplus x_4)$. Figure 2.4 (a) shows the accepted symbol for Toffoli4 and Figure 2.4 (b) shows the quantum realization disclosed within, where W and W^+ are unitary matrices known as the Controlled-W and Controlled-W+ gate [31, 50]. Where $W = \frac{1}{2}\begin{bmatrix} 1+\sqrt{i} & 1-\sqrt{i} \\ 1-\sqrt{i} & 1+\sqrt{i} \end{bmatrix}$ and $W^+ = \frac{1}{2}\begin{bmatrix} 1-i\sqrt{i} & 1+i\sqrt{i} \\ 1+i\sqrt{i} & 1-i\sqrt{i} \end{bmatrix}$, and such that $W^4 = (W^+)^4 = NOT$ and $W \times W^+ = W^+ \times W = I$ [31, 50].



(a)

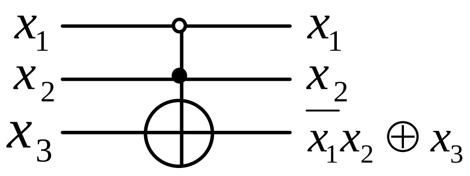


(b)

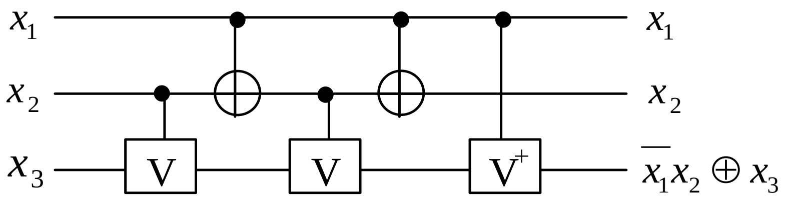
Figure 2.4. (a) Toffoli4 gate symbol (b) Quantum realization of the Toffoli4 gate

2.3.5 The Mixed-Polarity Toffoli Gate

The mixed-polarity Toffoli gate is a three bit reversible gate. For mixed-polarity Toffoli gate with one negative bit, it has a quantum cost of 5 [32, 51]. For mixed-polarity Toffoli gate with two negative bits, it has a quantum cost of 6 [54]. The gate maps the Boolean pattern (x_1, x_2, x_3) to $(x_1, x_2,\ (a_1 \oplus x_1)(a_2 \oplus x_2) \oplus x_3)$, where a_1, a_2 are coefficients that tells which line is a negative polarity control line. If $a_x = 1$, this means that line x is a negative polarity control line. Depending on which input bit to be a negative bit, the mixed-polarity Toffoli gate will result in three combinations. Figure 2.5 (a), Figure 2.6 (a) and Figure 2.7 (a) shows the accepted symbol of these gates with their corresponding function. Figure 2.5 (b), Figure 2.6 (b) and Figure 2.7 (b) shows the quantum realization disclosed [32, 51, 54]. From the Barenco-type realizations seen in the mixed-polarity Toffoli gates and the Toffoli gate, the positive and negative control signal are formed through different in connection of the V or V^+ elementary gate realizing it [31]. Since pin order of the gate is not considered for theoretical based simulation, the two mixed-polarity Toffoli gates in Figure 2.5 and Figure 2.6 are considered as the same gate.

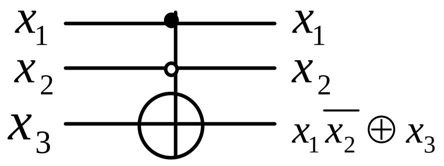


(a)

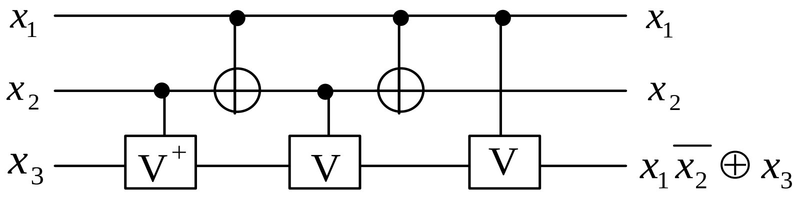


(b)

Figure 2.5. (a) Mixed-polarity Toffoli gate symbol with first bit as negative bit (b) Quantum realization of the mixed-polarity Toffoli gate with first bit as negative bit

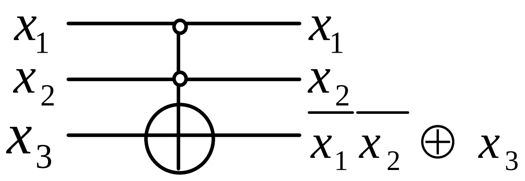


(a)

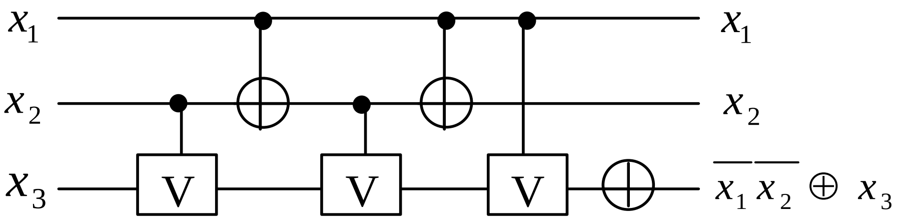


(b)

Figure 2.6. (a) Mixed-polarity Toffoli gate symbol with second bit as negative bit (b) Quantum realization of the mixed-polarity Toffoli gate with second bit as negative bit



(a)

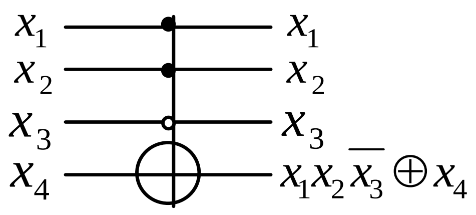


(b)

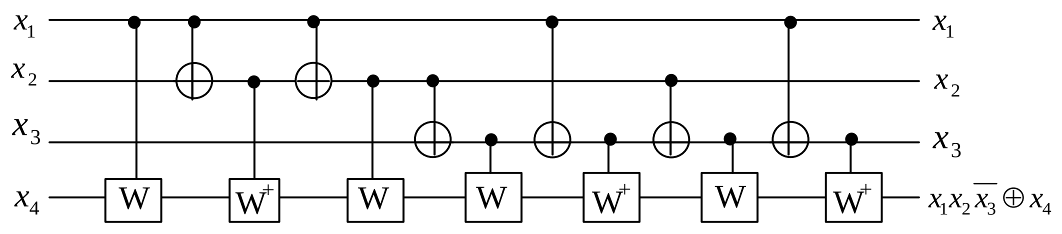
Figure 2.7. (a) Mixed-polarity Toffoli gate symbol with two negative bits (b) Quantum realization of the mixed-polarity Toffoli gate with two negative bits

2.3.6 The Mixed-Polarity Toffoli4 Gate

The mixed-polarity Toffoli4 gate is a four bit reversible gate. For mixed-polarity Toffoli4 gate with one or two negative bits, it has a quantum cost of 13. And as for mixed-polarity Toffoli4 gate with three negative bits, it has a quantum cost of 14. The gate maps the Boolean pattern (x_1, x_2, x_3, x_4) to $(x_1, x_2,\ (a_1 \oplus x_1)(a_2 \oplus x_2)(a_2 \oplus x_3) \oplus x_4)$, where a_1, a_2, a_3 are coefficients which tell that which line is a negative polarity control line. If $a_x = 1$, this means that line x is a negative polarity control line. Depending on which input bit to be a negative bit, the mixed-polarity Toffoli4 gate will result in three combinations. Figure 2.8 (a), Figure 2.9 (a) and Figure 2.10 (a) show the accepted symbol for these gates. Figure 2.8 (b), Figure 2.9 (b) and Figure 2.10 (b) show their quantum realization disclose within, where A is introduced as a new unitary matrix which is known as the Controlled-A gate. Such that $A = \frac{1}{2}\begin{bmatrix} 1 - i\sqrt{i} & 1 + i\sqrt{i} \\ 1 + i\sqrt{i} & 1 - i\sqrt{i} \end{bmatrix}$, which represents $A = W^3 \times NOT$.

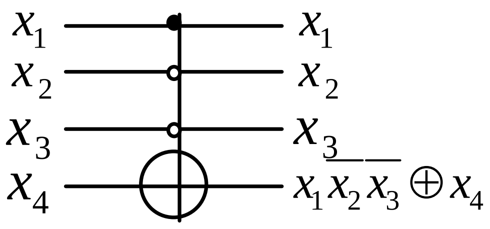


(a)

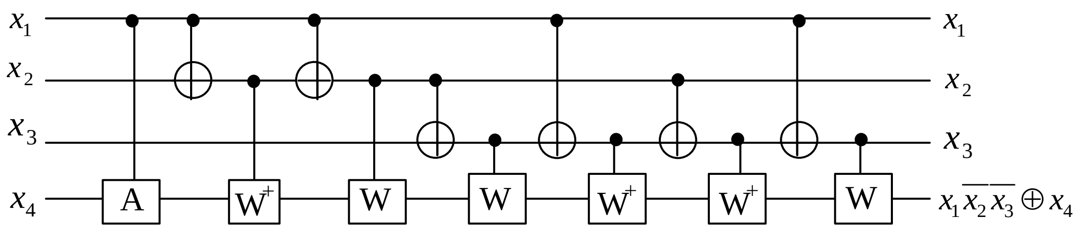


(b)

Figure 2.8. (a) Mixed-polarity Toffoli4 gate symbol with one negative bit (b) Quantum realization of the mixed-polarity Toffoli4 with one negative bit



(a)



(b)

Figure 2.9. (a) Mixed-polarity Toffoli4 gate symbol with two negative bits (b) Quantum realization of the mixed-polarity Toffoli4 with two negative bits

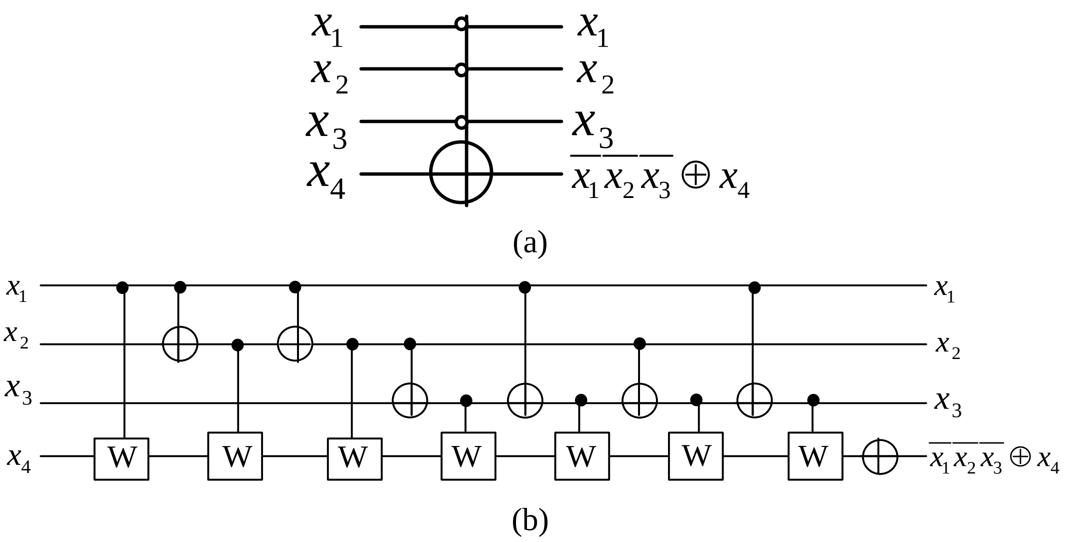


Figure 2.10. (a) Mixed-polarity Toffoli4 gate symbol with three negative bits (b) Quantum realization of the mixed-polarity Toffoli4 with three negative bits

2.3.7 The SWAP Gate

The SWAP gate is a two bit reversible gate, it has a quantum cost of 3 [55]. The gate maps a Boolean pattern (x_1, x_2) to (x_2, x_1). Figure 2.11 (a) shows the accepted symbol of the gate with its function and Figure 2.11 (b) shows the quantum realization disclosed [55]. From its quantum realization connection as shown in Figure 2.11 (b), the SWAP gate is constructed from 3 CNOT gate.

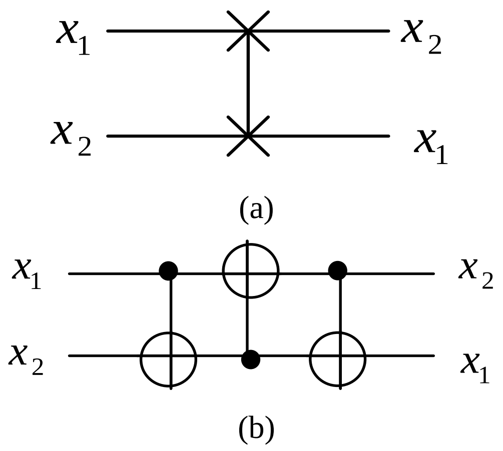
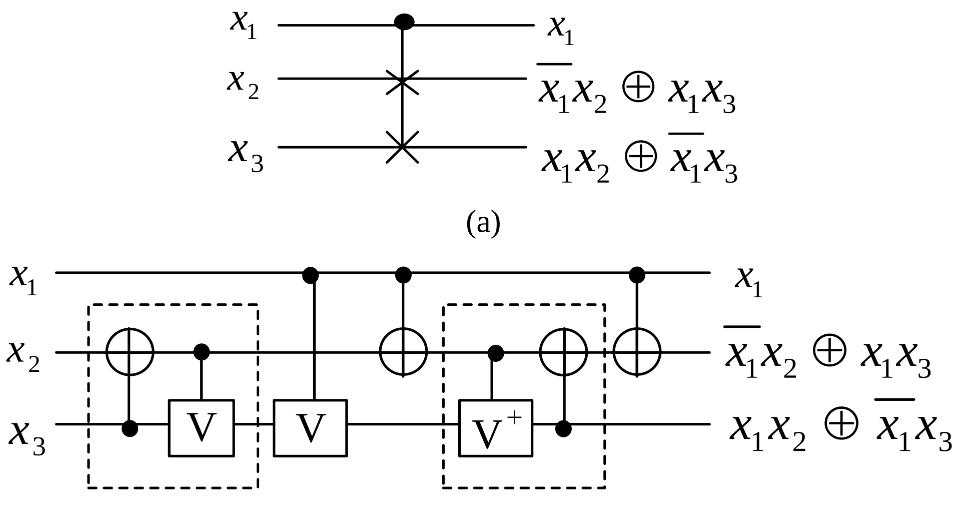


Figure 2.11. (a) SWAP gate symbol (b) Quantum realization of the SWAP gate

2.3.8 The Fredkin Gate

The Fredkin gate [46] is a three bit reversible gate, it has a quantum cost of 5 [56]. The gate maps a Boolean pattern (x_1, x_2, x_3) to $(x_1,\ \overline{x_1}x_2 \oplus x_1x_3,\ x_1x_2 \oplus \overline{x_1}x_3)$. Figure 2.12 (a) shows the accepted symbol of the gate with its function and Figure 2.12 (b) shows the quantum realization disclosed [56-58]. From Figure 2.12 (b), when a CNOT gate is connected head-to-toe with a Controlled-V or a Controlled-V+ gate, the quantum cost can be reduced to 1 [59, 60]. The Fredkin gate is known to be an universal gate which means that any logical or arithmetic operation can be solely constructed entirely using Fredkin gate [61, 62].


x_1
x_2
x_3
x_1
$\overline{x_1}x_2 \oplus x_1x_3$
$x_1x_2 \oplus \overline{x_1}x_3$
(a)
V
V
V^+

(b)

Figure 2.12. (a) Fredkin gate symbol (b) Quantum realization of the Fredkin gate

2.3.9 The Peres Gate

When two control lines of a Toffoli gate are connected to a CNOT gate as seen in Figure 2.13, the CNOT gate constructed within the Toffoli gate and the external CNOT gate can be cancelled out following the reduction rules explained in [52]. The resulting logic circuit is known as the Peres gate [63]. The Peres gate is a three bit reversible gate, it has a quantum cost of 4 [48]. The gate maps a Boolean pattern (x_1, x_2, x_3) to $(x_1,\ x_1 \oplus x_2,\ x_1x_2 \oplus x_3)$. Figure 2.14 (a) shows accepted symbol with its function and Figure 2.14 (b) shows the quantum realization disclosed [64].

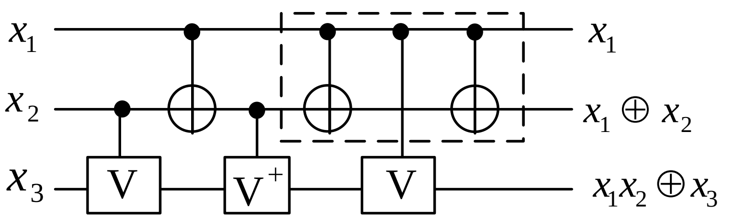
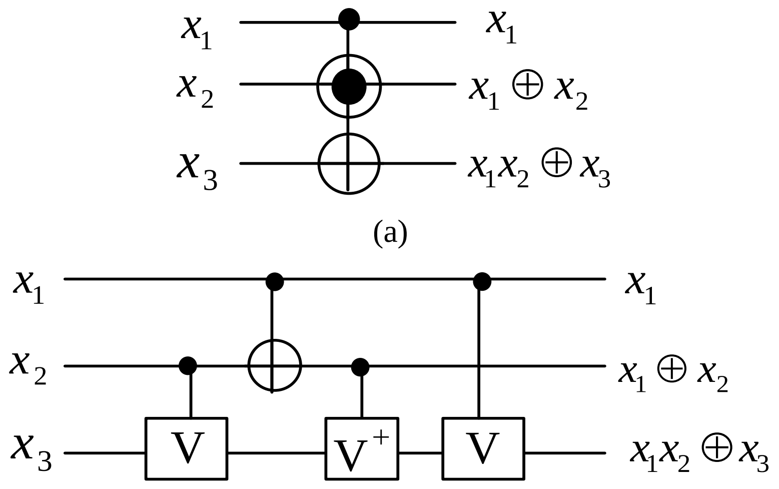


Figure 2.13. Toffoli gate with CNOT gate connected to both its control line



(b)
Figure 2.14. (a) Peres gate symbol (b) Quantum realization of the Peres gate

2.3.10 The Inverse-Peres Gate

Unlike the CNOT and Toffoli gate, the Peres gate is not a self-reversible gate. This means that the inverse connection of the Peres gate will give different Boolean equation and hence result is a different gate. The inverse connection of the Peres gate is called the inverse-Peres gate or the TR gate in [65, 66]. It is a three bits reversible gate and has a quantum cost of 4. The gate maps a Boolean pattern (x_1, x_2, x_3) to $(x_1,\ x_1 \oplus x_2,\ x_1\overline{x_2} \oplus x_3)$. Figure 2.15 (a) shows the accepted symbol with its function and Figure 2.15 (b) shows the quantum realization disclosed [64].

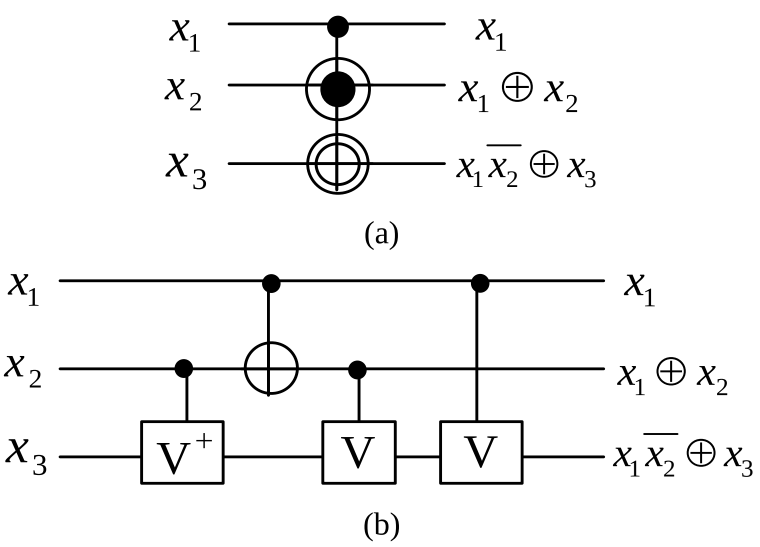


Figure 2.15. (a) TR gate symbol (b) Quantum realization of the TR gate

2.4 Delay of Reversible Gates

The time required to execute a reversible gate is known as the propagation delay. As research on reversible logic remains on theoretical work, calculation of actual delay in time is not possible. Hence a difference metric for measuring delay is considered. In this dissertation, logical depth is used as delay measurement and is denoted by the symbol Δ [44, 67]. Similar to quantum cost calculation, the propagation delay of each elementary quantum gate is considered as Δ 1 [44, 67]. Propagation delay of a reversible circuit can be measured in parallel. When two or more quantum gates are connected in parallel to one another, they are executed simultaneously; hence the propagation delay is only affected by the quantum gate that contributes to the highest delay. Table 2-4 shows the propagation delay of the reversible gate discussed in Section 2.3.

Table 2-4. Propagation delay of reversible gate

Reversible Gate	Delay Δ
NOT, CNOT	1
Toffoli	5
Toffoli4	13
Mixed-polarity Toffoli (1 negative bit)	5
Mixed-polarity Toffoli (2 negative bit)	6
Mixed-polarity Toffoli4 (1 and 2 negative bit)	13
Mixed-polarity Toffoli4 (3 negative bit)	14
SWAP	3
Fredkin	5
Peres, Inverse-Peres	4

2.5 Representation Model

Reversible functions can be described in several ways such as truth table, matrix, binary decision diagram and positive polarity Reed-Muller expansion. Each of these representations is discussed in details in the sub-sections below:

2.5.1 Truth Table

Truth table representation model is the simplest and straight forward method to describe a reversible function [68]. For a reversible function of n variable described using truth table representation, it will be written in a table of n columns and 2^n rows. Figure 2.16 and Table 2-5 show an example of a reversible circuit and its truth table representation.

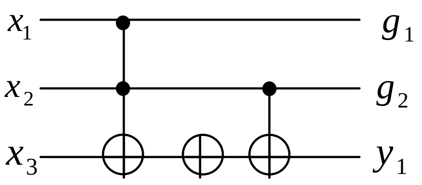


Figure 2.16. Reversible circuit

Table 2-5. Truth table representation

	Input x_1, x_2, x_3	Output g_1, g_2, y_1
0	0 0 0	0 0 1
1	0 0 1	0 0 0
2	0 1 0	0 1 0
3	0 1 1	0 1 1
4	1 0 0	1 0 1
5	1 0 1	1 0 0
6	1 1 0	1 1 1
7	1 1 1	1 1 0

2.5.2 Permutation Matrix

Permutation matrix based representation reflects the quantum state evolution of a reversible function [12, 69]. It represents the permutation function of a reversible function in a 0-1 matrix with only one 1 appear in each column. If a reversible function has n variables, it will be represented by a matrix of 2^n rows and columns. The (i, j)-th element of each column of the matrix is set to 1 when the input bit string corresponding to the i-th row is mapped to the output bit string corresponding to the j-th column [57]. Example below shows a matrix representation of a CNOT gate with it truth table defined in Table 2-6.

Table 2-6. CNOT gate truth table

	Input x_1, x_2	Output y_1, y_2
0	0 0	0 0
1	0 1	0 1
2	1 0	1 1
3	1 1	1 0

The output of the CNOT gate is written as its permutation form

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Writing the outputs into a matrix, the permutation matrix of the CNOT gate is expressed as

$$\text{CNOT} = \begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}$$

2.5.3 Binary Decision Diagram

Any Boolean function can be visually represented by different type of Decision Diagrams (DD) [70-72]. A Binary Decision Diagram (BDD) is a directed acyclic graph where a Shannon decomposition ($f = \overline{x_i} f_{x_i=0} + x_i f_{x_i=1}$) is carried out in each non-terminal node [43, 73]. Figure 2.17 shows an example of a BDD for the Boolean function $x_3 + x_1 x_2$.

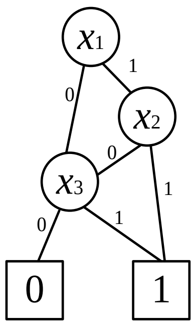


Figure 2.17. BDD for Boolean function $x_3 + x_1x_2$

2.5.4 Reed Muller Expansion

Any Boolean function can be described as EXOR sum-of-product (ESOP) expansion [74]. Since most commonly used reversible gates reflect their operation in Reed Muller logic form, this makes Reed Muller expansion suits reversible logic expression better [75]. The positive-polarity Reed-Muller (PPRM) expansion is those un-complemented ESOP expansion which can be written as a polynomial of the form $a_0 \oplus a_1x_1 \oplus \ldots \oplus a_nx_n \oplus a_{1,2}x_1x_2 \oplus a_{1,3}x_1x_3 \oplus \ldots \oplus a_{n-1,n}x_{n-1}x_n \oplus \ldots \oplus a_{1,2,\ldots,n}x_1x_2 \ldots x_n$ where $a_0, a_1, \ldots, a_n, a_{1,2}, a_{1,3}, \ldots, a_{1,2,\ldots,n}$ refer to the Boolean coefficients [76]. The size of a PPRM expansion is 2^n where n denotes the number of variables. To express a Boolean function into the PPRM expansion, the Boolean function is first expressed in their ESOP expansion. Then the un-complement rule $\overline{x_n} = 1 \oplus x_n$ is applied to all complemented term, where $\overline{x_n}$ refers to the complemented term. Example below shows a logic function expressed in its ESOP expansion and how it is converted into its PPRM expansion [75].

$$\begin{aligned} y &= \bar{a}b \oplus \bar{b}\bar{c} \\ &= (1 \oplus a)b \oplus (1 \oplus b)(1 \oplus c) \\ &= (b \oplus ab) \oplus (1 \oplus b \oplus c \oplus bc) \\ &= 1 \oplus c \oplus ab \oplus bc \end{aligned}$$

CHAPTER 3 LITERATURE REVIEW

In this chapter, existing literature works related to the research work are being discussed. In Section 3.1, review on the commonly used reversible logic synthesis algorithms proposed by various authors over the period has been discussed. In Section 3.2, review on the existing design of reversible full adder circuit has been discussed. In Section 3.3, review on the existing research work on reversible BCD adder has been discussed.

3.1 Related Work on Synthesis of Reversible Logic

In this section, overviews of the widely proposed reversible logic synthesis algorithm are discussed. The synthesis algorithms are being categorized into several subsections according to their types and functions.

3.1.1 Transformation Based Synthesis Algorithm

Transformation based synthesis algorithm is one of the earliest developed reversible logic synthesis algorithm. The synthesis algorithm was first introduced by Iwama et al. [25] in which the authors proposed several rules to transform Boolean function into a set of Toffoli based circuits. The proposed work has shown that any circuit can be brought to a canonical form through utilizing their proposed rules. Unfortunately, the proposed work does not provide any practical example thus the proposed work are more for theoretical interest. The algorithm was later extended by Miller et al. [26] by adding transformation rules on a set of predefined patterns called templates. The algorithm works by transforming Boolean function into a set of Toffoli based circuits in the truth table format using the defined transformation rules. After the resulting Toffoli based circuit has been obtained, several simple template matching rules are applied to further simplify the circuit. The synthesis results were considerably good during the era. An extension version to the existing synthesis algorithm given in [25],[26] has been made by Dueck et al. [28] and Maslov et al. [29], respectively, which has added SWAP gate and Fredkin gate into their transformation database. The algorithm given in [26] was later improved by Maslov et al. [77] by introducing bidirectional algorithm that allows the algorithm to iterate through the truth table looking for difference between the inputs and outputs rather than the traditional method which searches only on the output side. Later Malsov et al. [78] used a mixture of different techniques which includes the MMD (Miller-Maslov-Dueck) developed in [77], template matching, Reed-Muller spectra and resynthesis to synthesize reversible function in term of gate count or quantum cost. The synthesis algorithm drops the traditional truth table representation method use by earlier algorithms and uses the Reed-Muller spectra form which direct reflects the reversible gate’s operation from the NCT gate library. Using the Reed-Muller spectra form, it gives better gate substitution as compared to the earlier method and thus improves in the synthesis results. The flow of the algorithm can be broken into three steps. In the first step, the MMD is applied which selects reversible gates for transformation until all identity term is determined. In the second step, template matching is applied which simplify each of the synthesized networks using templates, choosing smaller network. In the last step, resynthesis is applied where a sequence of gates is chosen from the network as an independent problem and computation are preformed to reduce the quantum cost.

3.1.2 Rule Based Synthesis Algorithm

Arabzadeh et al. [33] proposed a rule based optimization approach that can be used after transformation based synthesis algorithm such as that given in [26, 77] to transform NCT based gate circuit into mixed-polarity Toffoli gate based circuit. The algorithm works by adding rules to reduce redundant NOT gate on the synthesized circuit. The algorithm can be broken into two steps: during the first step, multiple NOT gates are applied all across a given reversible circuit and converting Toffoli based gates into mixed-polarity Toffoli gate, all redundant NOT gates are being removed at this step. At the second step, a Karnaugh map-based optimization is applied to optimize sub-circuits with common-target gates.

Datta et al. [79], the authors proposed a post-synthesis optimization rules that uses template matching to transform NCT based reversible circuits into a cascade of mixed-polarity based circuit. Template matching rules are applied to cascade several Toffoli based gates together to match the function of a mixed-polarity based gate and replace them accordingly. Their current template rules are only capable of matching mixed-polarity Toffoli gate up to 4 variables.

3.1.3 Cycle Based Synthesis Algorithm

Cycle based synthesis algorithm are known for separating the entire permutation into several cycles and synthesize them separately. The first cycle based synthesis algorithm is introduced by Shende et al. [27] in which the algorithm decompose any given permutation into a set of disjoint cycles and synthesize individual cycle separately. This method is efficient when dealing with permutations that do not have a regular patterns and reversible function that leave many input combinations unchanged. Yang et al. [80] introduced a similar method to [27] except that the algorithm choose to drop the uses of CNOT gate and uses only NOT and Toffoli gate when synthesizing qubit function that is greater than 3 variables. The algorithm has proposed several permutations formula that can be easily applied on targeted function area and transform them to obtain the synthesized gates. And Prasad et al. [81] improve the existing algorithm in [27] by applying NOT and CNOT gates instead of Toffoli gate in many situations. The gate count and quantum cost results improves dramatically through applying the following changes.

Sasanian et al. [82] extended the algorithm given in [27] by removing the decomposition of larger cycles into a set of 2-cycles which results in using more reversible gates. In the proposed method, larger cycles are decomposed into a set of single 3-cycles, pairs of 3-cycles, and pairs of 2-cycles and synthesize the resulted cycle directly. This direct implementation of cycles greatly reduce unwanted reversible gates in the synthesis algorithm compared to the existing algorithm given in [27].

In Saeedi et al. [83], a *k*-cycle-based synthesis method is developed that uses a set of seven building blocks to directly synthesize a given permutation to reduce both quantum cost and average run-time. Given a large cycle inputs, the synthesis algorithm uses the decompose rules to extract the seven building blocks from the input specification before synthesizing them. The seven building blocks include a pair of 2-cycles, a single 3-cycles, a pair of 3-cycles, a single 5-cycles, a pair of 5-cycles, a single 2-cycles (4-cycles) followed by a single 4-cycles (2-cycles) and a pair of 5-cycles. This algorithm shows capability to synthesize reversible function of large variable (up to variable size of 20 given a time limit of 12 hour per function) and produces better quantum cost result for reversible benchmark function for variable size 8 and above.

In Saeedi et al. [84], the proposed algorithm uses a set of building blocks and a library to synthesize a given specification. Each specification is considered as a permutation with several cycles where each cycle is synthesized by some reversible gates. If a given cycle is found in the library, it is synthesized directly. Otherwise, a decomposition algorithm detaches the building blocks from the given cycle and explores all possible minimal and inequivalent factorizations where the number of disjoint cycles is maximized. Experimental results have shown that the proposed approach reduces the quantum cost and runtime.

3.1.4 Binary Decision Diagrams Based Synthesis

Binary Decision Diagrams (BDDs) based synthesis algorithm was first introduced by Kerntopf [85]. For any given Boolean function, a Decision Diagram is constructed through examine all possible gates. Gates that minimize the Decision Diagram complexity are selected and further analyzed by repeating the same process. The process stops when all paths are analyzed and the minimal node path is selected and transformed into reversible circuits. Wille et al. [86] introduced a different algorithm which starts by directly constructing a BDD. Each of the BDD nodes is substituted by a cascade of reversible gates. Since BDDs may contain shared nodes, hence it will result in using fan-outs which is not allowed in reversible logic. To overcome it, additional ancilla input and CNOT gate are added. Although the algorithm leads to a good reduction in quantum cost, but the usage of a large amount of ancilla input is rather impractical. The algorithm is later improved by Wille et al. [3], by introducing a post-processing optimization method that reduces the garbage line by merging some garbage output lines with appropriate ancilla input lines. Although improvement has been made, improvement is not obvious and the problem remains.

Krishana et al. [87] introduced a technique that uses isomorphic sub-graph matching which helps map sub-graphs of the BDD to structures with known reversible circuit templates. Using this approach it greatly reduces the number of ancilla inputs made by earlier BDD based synthesis algorithm [86].

Ancilla problem for BDD based synthesis algorithm was finally solved by a recent proposed synthesis algorithm by Soeken et al. [88]. The concept of the proposed algorithm in [88] is based on Young subgroups using symbolic function representations than can efficiently be implemented with BDDs. As a result the synthesis algorithm generates no ancilla inputs and able to cope with large reversible functions.

3.1.5 Search Based Synthesis Algorithm

Gupta et al. [89] proposed a search based synthesis algorithm, which reversible logic s synthesized in term of gate count using positive polarity Reed-Muller (PPRM) expansion. The algorithm works by traversing the input reversible function into a search tree and determine all possible solution using the gate matching factor. For a given amount of time, the algorithm searches all possible solution and stores only the best solution.

The algorithm was later improved by Saeedi et al. [90], which introduced several new substitutions rules to the original ones. Their experimental results have shown that their method can lead to a better synthesis circuit with respect to the total circuit cost as well as the probability of leading to a synthesized circuit better than the existing algorithm given in [89].

Donald et al. [30] further extended the algorithm of [89] to handle SWAP gate, Fredkin gate, Peres gate and reverse Peres gate in a similar search based framework. Capability to synthesize in term of quantum cost was also added to the algorithm.

The synthesis method of the algorithm remains unchanged compared to the earlier work given in [89].

3.1.6 Graph Based Synthesis Algorithm

Graph based synthesis algorithm is proposed by Yexin et al. [91] in which reversible logic is synthesized in term of gate count using NCT library gates. The algorithm targets on faster synthesis time by reducing the function complexity during each synthesis steps. Although synthesis time is shorten, the algorithm is still able to capable of finding possible solution for all input reversible function. The drawback of the algorithm is that it lacks of the ability to determine the best gate to substitute in each step and hence often uses more unnecessary gates to realize a function.

3.1.7 Optimal Based Synthesis Algorithm

In Golubitsky et al. [92], the authors presented an optimal based reversible synthesis algorithm which is capable of finding optimal solution through matching the input reversible function to a set of presets in their database. Using the database, the gate count of the optimal circuit can be easily found in a short amount of time through lookup in their canonical representative form. Golubitsky et al. [54] made an extension to further improve the existing database in [92] which removes all inverse functions and added several new functions to the existing databases. This improves the performance of the algorithm and allows the algorithm to synthesize reversible functions or higher variables. When dealing with larger reversible function, the algorithm may break the function into several sub-functions and matches them individually using their database and combine them afterwards.

Szprowski et al. [34] further extends the algorithm presented in [92] through adding method to further synthesize in term of quantum cost after being synthesize using the algorithm given in [92]. The proposed algorithm is able to generate optimal circuits up to 4 variables based reversible functions. Although gate count is at minimal but this does not mean that these circuits are exact minimal in terms of quantum cost.

This is because there might be longer cascades with lower quantum cost value. In Golubitsky et al. [93], the authors further improve the algorithm in [34] by using only classical decompositions techniques without using additional lines. By using this technique, it provides substantial saving of quantum cost. Szyprowski et al. [36] further extended the work given in [93] by adding mixed-polarity based Toffoli gate into the synthesis library.

Li et al. [94] proposed an algorithm that is able to find exact minimum solution to most 4-bit reversible function using NCTP library (NOT, CNOT, Toffoli, Toffoli4, Peres and inverse Peres gates). When synthesizing a given function, the proposed algorithm first use a hash table to determine a combination with the minimization of gate count using NOT, CNOT, Toffoli and Toffoli4 gates. Then the proposed algorithm merged and split the hash tables to generate a longer hash table using several of their permutation rules. Lastly, the proposed algorithm synthesizes the resulting circuits in term of quantum cost through transforming the selected path into Peres and Inverse Peres gates. The proposed synthesis algorithm is designed in a memory efficient way in which it uses the shortest amount of coding for their permutations rules and uses topological compression and flexible data structures for better memory savings. With comparison to the synthesis algorithm proposed in [34], it archived better synthesis time and produces better quantum cost results.

3.1.8 Summary

To summarize the proposed algorithms in the literature, all the key features of each synthesis algorithm are listed out in Table 3-1. The first column “Synthesis Method” shows the synthesis methods proposed in the literatures. The second column “Features” shows the important features which are considered in the different approaches. Third column “Limitations” shows the limitation of each algorithm. Fourth column “Library” shows the synthesis library function which is used in the proposed algorithms and the last column “Metric” indicates the synthesis parameter focus on.

Table 3-1. Comparison table of all discussed synthesis methods

Synthesis Method	Features	Limitations	Library	Metric
Iwama, *et al.* (2002) [25]	• Transformation based synthesis	• Large amount of ancilla inputs and garbage output • Circuit dependency	NCT	GC
Miller, *et al.* (2003) [26]	• Transformation based synthesis • No ancilla input	• Limited scalability • Circuit dependency	NCT	GC
Dueck, *et al.* (2003) [28]	• Transformation based synthesis	• Large amount of ancilla inputs and garbage output • Circuit dependency	NCTSF	GC
Maslov, *et al.* (2005) [29]	• Transformation based synthesis • No ancilla input	• Limited scalability • Circuit dependency	NCTSF	GC
Maslov, *et al.* (2005) [77]	• Transformation based synthesis • No ancilla input	• Limited scalability	NCT	GC
Maslov, *et al.* (2007) [78]	• Transformation based synthesis • Able to cope with large function	• Limited scalability	NCT	GC
Arabzadeh, *et al.* (2010) [33]	• Rule based synthesis • No ancilla input	• Circuit dependency	m-NCT	QC
Datta, *et al.* (2013) [79]	• Rule based synthesis • No ancilla input	• Limited scalability • Circuit dependency	m-NCT	GC, QC
Shende, *et al.* (2003) [27]	• Cycle based synthesis • No ancilla input	• Circuit dependency • Limited scalability	NCT	GC
Yang, *et al.* (2006) [80]	• Cycle based synthesis • No ancilla input	• Circuit dependency • Limited scalability	NCT	GC
Prasad, *et al.* (2006) [81]	• Cycle based synthesis • No ancilla input	• Circuit dependency • Limited scalability	NCT	GC
Sasanian, *et al.* (2009) [82]	• Cycle based synthesis • No ancilla input	• Limited scalability	NCT	GC
Saeedi, *et al.* (2010) [83]	• Cycle based synthesis • No ancilla input • Able to cope with large function	• Circuit dependency	NCT	QC
Saeedi, *et al.* (2010) [84]	• Cycle based synthesis • No ancilla input • Able to cope with large function	• Circuit dependency	NCT	QC
Kerntopf (2004) [85]	• BDD based synthesis	• Limited scalability	NCTSF	QC
Wille and Drechsler (2009) [86]	• BDD based synthesis • Able to cope with large function	• Ancilla input • Garbage output	NCT	QC
Wille, *et al.* (2010) [3]	• BDD based synthesis • Able to cope with large function	• Ancilla input • Garbage output • Circuit dependency	NCT	Ancilla input
Krishna and Chattopadhyay (2014) [87]	• BDD based synthesis • Able to cope with function up to 9 variables • Much efficient in handling	• Ancilla input • Garbage output • Poor scalability	NCT	Ancilla input

	ancilla input			
Soeken, *et al.* (2016) [88]	• BDD based synthesis • Able to cope with large function • Ancilla free	• Circuit dependency • Requires large memory space	NCT	QC, Ancilla input
Gupta, *et al*,(2006) [89]	• Search based synthesis	• Limited scalability	NCT	GC
Saeedi, *et al.* (2007) [90]	• Search based synthesis	• Limited scalability	NCT	QC
Donald and Jha (2008) [30]	• Search based synthesis	• Limited scalability • Slow synthesis time	NCTSFP	GC, QC
Yexin, *et al.* (2009) [91]	• Graph based synthesis • Fast synthesis time	• Poor scalability	NCT	GC
Golubitsky, *et al.* (2010) [92]	• Optimal based synthesis • Fast synthesis time	• Function dependency • Limited scalability • Requires large amount of storage	NCT	GC
Golubitsky, *et al.* (2012) [54]	• Optimal based synthesis • Fast synthesis time • Able to cope with large function	• Function dependency • Limited scalability • Requires large amount of storage	NCT	GC
Szyprowski, et al. (2011) [34]	• Optimal based synthesis • Fast synthesis time	• Function dependency • Limited scalability	NCT	GC
Szyprowski, et al. (2011) [93]	• Optimal based synthesis • Fast synthesis time	• Function dependency • Limited scalability	NCT	GC, QC
Szyprowski, et al. (2013) [36]	• Optimal based synthesis • Fast synthesis time	• Function dependency • Limited scalability	m-NCT	GC, QC
Li, *et al.* (2014) [94]	• Optimal based synthesis • Fast synthesis time	• Function dependency • Only limited to 4 variable based reversible function	NCT	QC

3.2 Related Work on Reversible Full Adder Circuit

In many computational arithmetic units, adder is the fundamental building blocks which form the essential components of a computing system [64]. After reversible logic was introduced to digital design, reversible logic implementation on full adder circuit has been proposed by various authors over the years [26, 62, 95-99].

During the early research on reversible logic, the varieties of reversible gate are very limited. All logic functions realizations are only limited to Toffoli based gate family. The earliest reversible logic based implementation on full adder circuit is presented in Khlopotine et al. [95] and Miller et al. [26], which uses 2 Toffoli gates and 2 CNOT gates for the implementation. Using these gates to construct the reversible full adder circuit, the realization uses 1 ancilla input, 1 garbage output and has a total quantum cost of 12 and a delay of $\Delta 12$.

After Fredkin gate has been introduced to reversible logic design, usage of this gate for reversible full adder has been proposed in [62, 96]. In Bruce et al. [62], the reversible full adder circuit is constructed using 4 Fredkin gates and uses 2 ancilla inputs, 2 garbage outputs and has a total quantum cost of 20 and a delay of $\Delta 20$. In James et al. [96], the reversible full adder circuit is designed to have lower delay with the tradeoff of degrading other parameters. The overall circuit requires 5 Fredkin gates, 4 ancilla inputs, 3 garbage outputs and has a total quantum cost of 25 and a delay of $\Delta 15$.

After Peres gate has been introduced to reversible logic design, usage of this gate for reversible full adder has been proposed by Islam et al. [97]. The overall realization uses only two Peres gates which were proven to be the most efficient design. Overall the full adder circuit design uses 1 ancilla input, 1 garbage output and has a total quantum cost of 8 and a delay of $\Delta 8$.

Mohammadi et al. [98] applied genetic algorithm to design a reversible full adder circuit. The proposed reversible full adder circuit is constructed using only primitive gates and later, it is named as modified-TSG (MTSG) gate by Nayeem et al. [100]. Overall the full adder circuit design uses 1 ancilla input, 1 garbage output and has a total quantum cost of 6 and a delay of $\Delta 5$.

In Malsov et al. [99] applied template match rule and level compactor on the earliest reversible full adder circuit design which were constructed using 2 Toffoli gates and 2 CNOT gates. It decomposed the gates into their primitive gate form and applied template match rule and level compactor to minimize the intermediate gates and delay. The resultant reversible full adder circuit design is currently the most efficient design in terms of quantum cost and delay. Overall the proposed full adder circuit design uses 1 ancilla input, 1 garbage output and has a total quantum cost of 6 and a delay of $\Delta 4$.

Besides that, there were many other reversible full adder circuits proposed over the years, however most of the proposed works are not promising as they are unable to present the quantum realization (primitive gate form) of the proposed reversible gates. Since quantum realization is not presented, there is no verification on whether the proposed design is unique and not just result of cascading multiple reversible gates together. Hence, without quantum realization, quantum cost (which is the most important parameter for all reversible logic design), the proposed reversible designs were not able to present in this section.

3.3 Related Work on Reversible BCD Adder

In computers, numbers are stored straight in binary format due to the ease of the hardware design because representation in binary form is always advantageous over decimal format [101]. But due to the characteristic of most floating-point number, they cannot be represented exactly in binary form and thus often be represented using approximate values [69, 102]. However the financial, commercial and internet-based applications are not able to tolerate such error of the conversion, thus computing in decimal format is gaining significant popularity and efforts [69, 102].

Hardware support for decimal arithmetic performs slower than binary arithmetic [103]. However fast decimal floating-point arithmetic is necessarily importance in financial and internet based applications. Therefore, faster circuits for Binary Coded Decimal (BCD) have great impact, as it is likely to be incorporated in more complex circuits like future mathematical processors.

Recent years, several reversible logic implementations of reversible BCD adder have been reported in [64, 96, 98, 103-106]. Quantum cost has always been one of the most important metric to be considered in all reversible logic design. Designing reversible BCD adder with low quantum cost has always been challenging in which other important parameters are not to be considered such as ancilla input, garbage output and delay. Balancing between these parameters has always been a challenging task to researchers.

Babu et al. [104] proposed reversible BCD adder circuit is constructed using two 4-bit reversible full adders, one combinational circuit and several CNOT gates to increase the fan out. The half adder circuit is constructed using a Peres gate and the full adder circuit is constructed by cascading a New gate [107] and one Peres gate. The combinational circuit is constructed using three Toffoli gates. The overall design uses 17 ancilla inputs, 22 garbage outputs and it produces 143 quantum cost. Until present, using Peres gate as a half adder circuit contributes to the least amount of quantum cost.

In Mohammadi et al. [98], genetic algorithms were used and don’t care concept was considered to design a reversible BCD adder in terms of quantum cost, garbage output and ancilla input. The proposed BCD adder [98] was constructed using two 4-bit full adders and a detection circuit. In the overall design, 16 ancilla inputs, 20 garbage outputs and produced 70 quantum cost were used. The significant about the BCD adder design was that the full adder circuit is designed through the genetic algorithm. The full adder has a quantum cost of 6 and a delay of Δ5. The proposed full adder circuit was later named as modified-TSG (MTSG) gate by Nayeem et al. [100].

Biswas et al. [103] proposed reversible BCD adder was constructed using a 4-bit full adder, an overflow detection circuit and a correction logic circuit. MTSG gate was used as their full adder circuit. The proposed design is mainly focused on reducing the garbage outputs, delay and quantum cost. The overall design used 7 ancilla inputs, 11 garbage outputs and produced 50 quantum cost.

In Thomsen et al. [105], mixed-polarity based Toffoli gate is used in the reversible BCD adder design. The adder component consists of a 4-bit binary half adder, an extension overflow circuit, a conditional 1-adder circuit and a conditional 6-adder circuit. High bit gates are used in all construction which has the great advantage of reducing circuit complexity, and as a result, the overall design uses very less reversible gate. However the drawback of using high bit gate is that it requires great amount of quantum cost to realize the gate function which made the design impractical. Besides, newer reversible logic designs are no longer focus on gate count as it has been taken over by quantum cost. The overall design uses 4 ancilla inputs, 7 garbage outputs and produced 200 quantum cost.

A quick addition of decimals (QAD) suitable for multi-digit BCD addition has been proposed by James et al. [96] using only Fredkin gate. The proposed reversible BCD adder design follows the concept of a carry look-ahead BCD adder in which it has the advantages of reducing the effect of propagation delay. The proposed work [96] uses only Fredkin gate throughout the entire work, however fan-out issue was not been taken into consideration. It can be noticed from the existing proposed work that several CNOT gates have been added and thus making their proposed adder fully reversible. After adding several CNOT gates, the overall design uses 35 ancilla inputs, 39 garbage outputs and produced 190 quantum cost.

In James et al. [106], a reversible BCD adder design and a reversible carry look-ahead BCD adder are proposed. In the proposed BCD adder design, only Toffoli based gate is used throughout the design. The components for the proposed BCD adder design consist of a 4-bit binary adder, an 8-correcion circuit, a k-bit generation circuit and a special adder circuit. The overall design uses 28 gates, 8 ancilla inputs, 12 garbage outputs and produced 80 quantum cost. The components for the carry select BCD adder design consist of a 4-bit binary adder, a 6-correcion circuit, a 2-to-1 multiplexer and a special adder. The overall design used 12 ancilla inputs, 16 garbage outputs and produced 102 quantum cost.

Thapliyal et al. [64] presented two reversible BCD adder designs. The first design consisted of a 4-bit binary adder, a detection circuit and a correction circuit. The overall design used 2 ancilla inputs, 6 garbage outputs and produced 88 quantum cost. The second design consisted of a 4-bit binary adder and a 5-bit binary to BCD converter circuit. The overall design used 1 ancilla input, 5 garbage outputs and produced 70 quantum cost. Both of the proposed designs were primarily optimized for ancilla input and garbage output. Other parameters such as quantum cost and delay were also considered.

Besides that, there were many other proposed reversible BCD adders, however most of the proposed works are not promising as they are unable to present the quantum cost (quantum realization) of the reversible gates used especially when the authors included their proposed reversible gates. Since quantum cost of is essential for all reversible logic design. Without it, there is no verification on whether the proposed work can be functionally verified as reversible. Hence, those proposed reversible designs were not presented in this section.

CHAPTER 4 SEARCH-BASED REVERSIBLE LOGIC SYNTHESIS USING MIXED-POLARITY TOFFOLI GATE

As reversible logic begins to gain significant research attention towards the growing of emerging technologies, reversible logic would most probably replace current existing logic systems due to the promising heatless ability of reversible logic. Since properties of reversible logic differ greatly than traditional logic gate, synthesis algorithms used for synthesizing traditional logic cannot be used for reversible logic. Therefore, a new set of synthesis algorithm is required for synthesizing Boolean functions into reversible logic. In this chapter, a search-based reversible logic synthesis algorithm is proposed. In the proposed algorithm, mixed-polarity Toffoli gate is used. In the chapter, an overview is given in Section 4.1, conversion of Boolean function into PPRM expansion is discussed in Section 4.2. The details of the proposed synthesis algorithm are discussed in Section 4.3. Simulation results are presented in Section 4.4 and conclusion is drawn in Section 4.5.

4.1 Overview

Over the last decade, synthesis of reversible logic has been intensively studied [43]. Research attention has been focused on the synthesis of circuits built using the NCT library gates [43]. Over the years, NCT library gates have been intensively studied and explored. Extension of the existing Toffoli gate has also been made to increase its variable size to allow more possible synthesis outcome. Recently, researcher has started to include polarity control capability into Toffoli gate's operation [31, 32]. Toffoli gate with polarity control is known as the mixed-polarity Toffoli gate. Usage of this gate in designing reversible logic synthesis algorithm has been reported in a number of research papers [33-36, 99, 108, 109]. Earlier proposed mixed-polarity Toffoli gates do not attract much research interest due to its poor quantum realization which are directly constructed from Toffoli gate with NOT gates [32]. Usage of this gate in reversible logic synthesis algorithm may enhance the algorithm synthesis time, but does not provide any improvement in terms of gate count and quantum cost. The research interest on this gate sparks again after its new quantum realization has been implemented in [32] and [39]. With the new quantum realization, usage of this gate provides better scaling because it is able to realize much complex reversible function using less quantum cost. Although there are various research works using this gate being proposed, usage of this gate for search-based reversible logic synthesis has not yet to be reported in any literature. Therefore in this research work, the usage of NCT gate library with mixed-polarity Toffoli gate for search based reversible logic synthesis algorithm has been proposed.

Search based synthesis algorithms are proven to be capable of exploiting reversible function better than other existing synthesis algorithm [43]. The current existing search based synthesis algorithms Gupta et al. [89] and Donald et al. [30] have shown that for any given reversible function, the proposed algorithms are able to determine all possible reversible circuit realization of the function. Besides, search based synthesis algorithm does not generate any extra lines like BDD based synthesis algorithms, thus keeping the synthesized function in its simplest form with least amount of ancilla inputs and garbage outputs. Moreover, search based synthesis algorithm does not require mixture of different techniques like transformation based synthesis algorithm, thus keeping the algorithm simple for implementation. Besides, synthesis result generated can be stored into the database of optimal based synthesis algorithm for further use.

In this chapter, a search-based reversible logic synthesis algorithm using mixed-polarity Toffoli gate is proposed. To the best of knowledge, this is the first attempt in literature to use mixed-polarity Toffoli gate for search-based reversible logic synthesis algorithm. The proposed algorithm synthesizes reversible function in their positive-polarity Reed Muller (PPRM) expansion and selects suitable gates for substitution through finding matching terms in the expansion. The algorithm targets on synthesizing reversible logic in term of gate count and quantum cost. The algorithm does not generate extra ancilla input thus keeping garbage output at minimal.

4.2 Converting Logic Function into PPRM Expansion

Before explaining the methodology of the proposed synthesis algorithm, an important issue is addressed on converting the irreversible logic function into PPRM expansion which can be used as input to the proposed synthesis algorithm.

Since most functions encountered are non-reversible, the first step is to convert them into reversible function. To do that, extra lines are added to create a unique one-to-one mapping between the inputs and the outputs. Once the reversible function has been determined, the next step is to convert it into ESOP expansion. In this work, this job is done using the program, EXORCISM-4 [110]. EXORCISM-4 uses efficient heuristics and look-ahead strategies to quickly find the ESOP expansion of a Boolean function. Once the ESOP expansion is obtained, the next stage is to convert them into PPRM expansion. This is done by applying rules $\overline{x_n} = 1 \oplus x_n$ to all complemented variables. Then reduction is done by cancelling out even number of identical variable terms $x_n \oplus x_n = 0$. Thus, the PPRM expansion is obtained.

In this work, the PPRM expansion of a logic function is obtained in one of these following ways:

1. Irreversible based functions are converted manually to reversible function through adding extra necessary lines. EXORSIM-4 is then used for converting them to ESOP expansion and then the expansion is later converted into PPRM expansion. These steps are done in a C language program and the resulting expansions are stored as PPRM format (.pprm).
2. Most reversible benchmarking functions obtained from [37], [38] and [36] are in '.blif' or '.pla' format. It is then converted into PPRM format using EXORCISM-4 and a written program.

4.3 Synthesis Algorithm

In this section, the flow of the proposed synthesis algorithm is described. For completeness of the dissertation, all the terms are explained in details. Examples are provided for better understanding.

4.3.1 Gate Substitutions

Given a reversible function expressed in its PPRM expansion, the proposed algorithm firstly explores the expansion and finds all possible matching reversible gates. For each selected reversible gate, a new path is formed and the gate is substituted into the original function to achieve reduction in the PPRM expansion. To determine which reversible gate to be selected will have a higher chance of leading to a better solution, a certain requirements are set. A reversible gate is selected only when the requirements are met. The requirements and substitutions of each reversible gate are as follows:

A NCT library gate will be selected, if and only if the PPRM expansion term contains any single variable term x_i and a *factor*, where *factor* is a term that can be any single variable to several multiplied variables terms that does not consist of x_i. For case NOT gate, the *factor* term is constant 1. A substitution of $x_i \oplus factor$ will be performed on expansion containing variable term x_i. The selected gates are limited by the number of variables in the function such that the largest gate size cannot be greater than the function variable term. For example, when synthesizing a three variable based function, the selected largest gate will be the Toffoli gate and for four variable based functions, the largest gate will be the Toffoli4.

A m-NCT library gate with a $n-2$ negative line (for a n lines variable gate) will be selected if and only if the PPRM expansion term contains any single variable term x_i and a *factor*, where *factor* is a term of any other two variables term to several multiplied variable term that does not consist of x_i. Gate substitution will be performed on expansion containing variable term x_i, the negative control variable will be selected from the *factor* term.

A m-NCT library gate with $n-1$ negative line (for a n lines variable gate) will be selected if and only if the PPRM expansion term contains a constant 1, any single variable term x_i and a *factor*, where *factor* is a term of any other two variables term to several multiplied variable term that does not consist of x_i. Gate substitution will be performed on expansion containing variable term x_i, the negative control variable will be selected from the *factor* term.

4.3.2 Hamming Distance

During reduction of reversible function from one path to another, the algorithm will find all possible reversible gates that simplify the original function. A single PPRM expansion may result in having several substitutions of different gates. Each of these substations is stored into a separate path. To determine which path will tend to lead to a better solution faster, Hamming Distance (HD) [26] approach is employed. The approach finds the sum of the variables difference between the inputs and outputs term. Based on experimental findings, path that contains the lowest non-zero HD has the highest probability of leading to a solution faster. Therefore, path that contains the lowest non-zero HD will be selected as the next transformation path.

When a solution is determined, the HD of a corresponding path reaches zero. Information of the path will be cached (gate connection, gate count and quantum cost). The algorithm then continues to explore the next lowest non-zero HD path for possible better solution. The algorithm will eventually stop until all the paths are explored. To improve synthesis time and memory, the program is set to terminate a path before even a solution is found. This is done by comparing the current gate count and quantum cost accumulated on the path with the current best solution. If it does not produce a better solution, then the path is terminated.

4.3.3 Algorithm

Figure 4.1 shows the flow chat for the main synthesis function of the proposed synthesis algorithm. The input to the function is a reversible function written in its PPRM expansion $f(x_1, x_2, \dots, x_n)$. Upon receiving the input PPRM expansion, the algorithm performs some initialization and then set the initial gate count and quantum cost to 0. The HD of the expansion is calculated during this stage. The algorithm then explores the PPRM expansion terms and determines all possible matching reversible gates for further reduction. When a reversible gate is determined, substitution will be done and a new path number will be assigned. Corresponding gate count, quantum cost and HD up to the current path are also updated. Next, the algorithm picks the next synthesize path by choosing path that contains the lowest non-zero HD. The algorithm iterate itself until a possible solution is found, where the HD of the corresponding path reaches zero. All information of the path such as reversible gates used, circuit connection from beginning to the end, accumulated gate count and quantum cost are stored as best solution. A copy of the accumulated gate count and quantum cost of the path are stored in register *best_gate_count* and *best_quantum_cost* for further comparison. The algorithm then continues exploring the next lowest non-zero HD path. Whenever a gate substitution is made and a new path is formed, the algorithm checks the solution generated up to the current path. A path is terminated and its information is discarded if it does not produce a better solution than the current best ones. The solution is compared in the following ways: When synthesizing in term of gate count, a path will be terminated if $gate\_count(path) > best\_gate\_count - 1$ or $gate\_count(path) == best\_gate\_count - 1$ with $quantum\_cost(path) \geq best\_quantum\_cost - 1$. As for synthesizing in term of quantum cost, a path will be terminated if $quantum\_cost(path) > best\_quantum\_cost - 1$ or $quantum\_cost(path) == best\_quantum\_cost - 1$ with $gate\_count(path) \geq best\_gate\_count - 1$. For cases where the explored path generates a better solution at HD equal to zero, the current best solution will be replaced. The algorithm will eventually come to a stop once all paths have been explored or the timer set by the user had been reached.

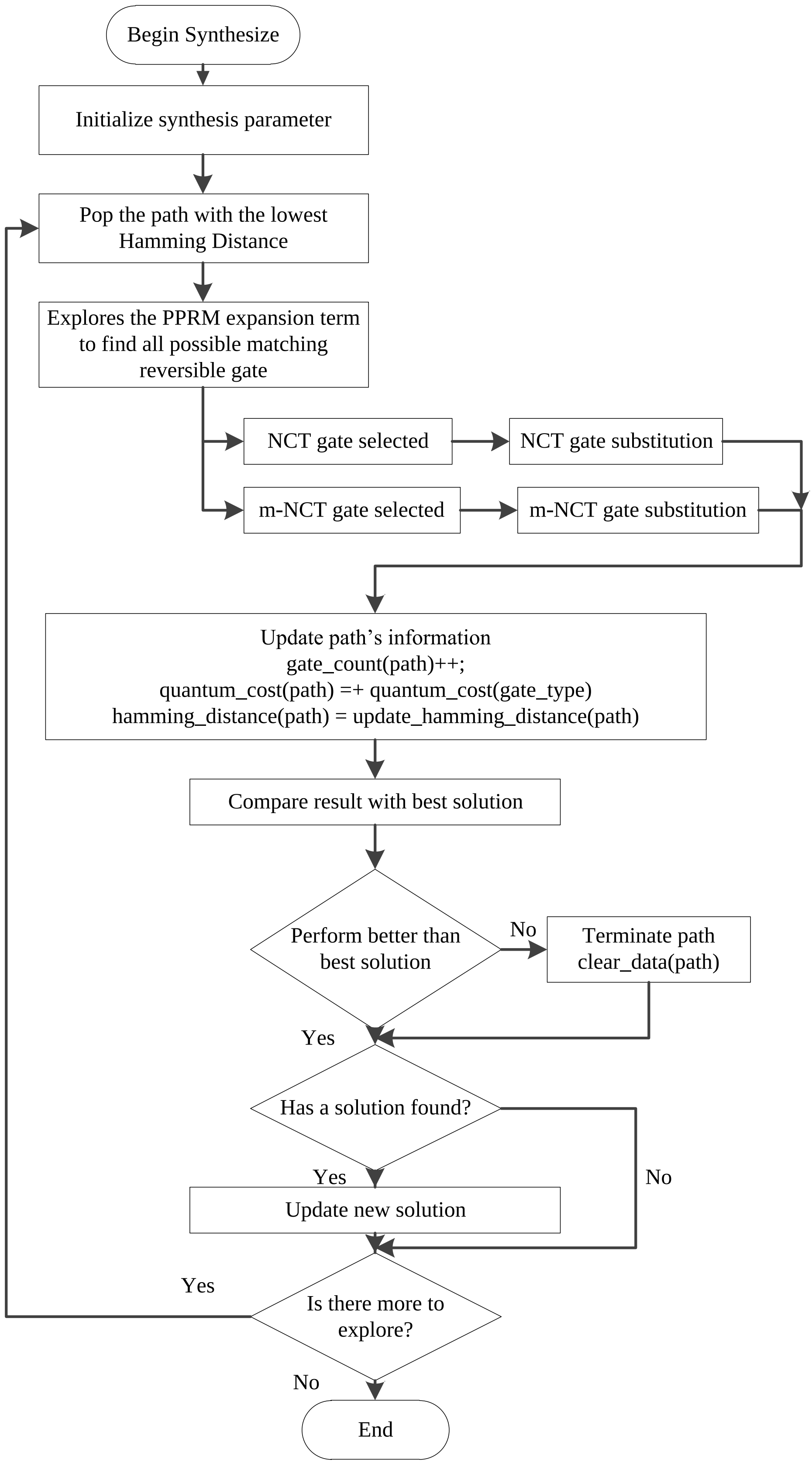

Begin Synthesize
Initialize synthesis parameter
Pop the path with the lowest Hamming Distance
Explores the PPRM expansion term to find all possible matching reversible gate
NCT gate selected
NCT gate substitution
m-NCT gate selected
m-NCT gate substitution
Update path's information
gate_count(path)++;
quantum_cost(path) =+ quantum_cost(gate_type)
hamming_distance(path) = update_hamming_distance(path)
Compare result with best solution
Perform better than best solution
No
Terminate path
clear_data(path)
Yes
Has a solution found?
Yes
No
Update new solution
Yes
Is there more to explore?
No
End

Figure 4.1. Flow chart of function Synthesize()

4.3.4 Example

For a better understanding of the proposed algorithm, it can be demonstrated by an example on synthesizing a reversible function with each step explains in details and how the best solution is achieved. In this example, the proposed algorithm is set to synthesize benchmark functions in term of gate count. For illustration purpose, a few reversible gates are selected when the algorithm explores a path and determine all possible matching gates. The benchmark function taken for this example is namely '3_17' and has such specification [7, 1, 4, 3, 0, 2, 6, 5] [37]. Having it written in PPRM expansion, it gives $a' = 1 \oplus a \oplus c \oplus ab \oplus ac$, $b' = 1 \oplus a \oplus b \oplus c$, $c' = 1 \oplus a \oplus b \oplus ab \oplus bc$. The program stores the expression in a tabular form, as shown in Table 4-1. The HD for this path is 13.

Table 4-1. Benchmark function 3_17 written in PPRM expansion

	Function
Coefficient	abc
1	111
a	111
b	011
ab	101
c	110
ac	100
bc	001
abc	000
HD	13
Gate applied	-
GC(QC)	0(0)

The algorithm analyzes the PPRM expansion terms and determines all possible matching reversible gates. Substitution is made for each selected gates and a new functions are formed and stored as sub paths. Unique path number is assigned and information such as the gate connections, gate count and quantum cost are updated. Table 4-2 shows the results of four sub paths with their respective gate substituted. At row 'gate applied', the symbol '1' means gate control line; 'x' means gate target line; '–' means no connection and 'o' means negative control line (only for m-NCT library gate).

Table 4-2. After first gate substitution

	Function			
Coefficient	abc	abc	abc	abc
1	111	000	100	001
a	111	111	010	011
b	011	110	011	010
ab	101	101	101	101
c	101	010	110	110
ac	100	100	100	100
bc	100	001	001	001
abc	000	000	000	000
HD	11	9	11	11
Gate applied	TOF x–1	TOF x– –	TOF –x–	TOF – –x
GC(QC)	1(1)	1(1)	1(1)	1(1)

From the transformed paths in Table 4-2, the algorithm calculates the HD for each of selected gates. As none of the path has a zero HD, this means that a valid solution has yet to be found. Next, the algorithm picks the lowest non-zero HD path as its next synthesize path. As shown in Table 4-2, the shaded colour path is the selected path as it is having the lowest non-zero HD. Gates are selected and substitution is made on this path and the result is as shown in Table 4-3. The gate count, quantum cost and HD for the new path are also updated.

Table 4-3. After second gate substitution

	Function			
Coefficient	abc	abc	abc	abc
1	000	000	000	000
a	111	111	100	110
b	110	110	110	111
ab	101	101	101	101
c	010	001	010	100
ac	100	100	101	001
bc	111	100	001	001
abc	000	000	000	000
HD	11	7	8	9
Gate applied	TOF x11	TOF x–1	TOF –x1	TOF –x1
GC(QC)	2(6)	2(3)	2(3)	2(3)

From Table 4-3, the lowest non-zero HD is 7. Further reduction is done on the path and results are shown in Table 4-4.

Table 4-4. After third gate substitution

	Function			
Coefficient	abc	abc	abc	abc
1	000	000	000	000
a	111	111	100	010
b	110	100	110	110
ab	101	101	101	000
c	001	001	001	001
ac	100	100	000	100
bc	010	110	100	100
abc	000	000	000	000
HD	7	9	4	5
Gate applied	TOF x11	TOF x1o	TOF 1x–	TOF 1ox
GC(QC)	3(8)	3(8)	3(4)	3(8)

From Table 4-4, the lowest non-zero HD is 4. Further reduction is done on the path and resulting in Table 4-5. It can be noticed that the output line *b* has already reach its identical term, so further reduction on this path will no longer use coefficient *b* as a control line.

Table 4-5. After fourth gate substitution

	Function		
Coefficient	abc	abc	abc
1	000	000	000
a	100	101	100
b	110	110	110
ab	101	100	000
c	001	000	001
ac	000	000	000
bc	101	100	100
abc	000	000	000
HD	5	6	2
Gate applied	TOF x11	TOF x1o	TOF 11x
GC(QC)	4(9)	4(9)	4(9)

From Table 4-5, the lowest non-zero HD is 2. Further reduction is done on the path and results are shown in Table 4-6.

Table 4-6. After fifth gate substitution

	Function		
Coefficient	abc	abc	abc
1	000	000	000
a	100	100	100
b	110	010	010
ab	000	000	000
c	001	001	001
ac	000	000	000

bc	000	100	000
abc	000	000	000
HD	1	1	0
Gate applied	TOF x11	TOF x1–	TOF x1o
GC(QC)	5(14)	5(10)	5(14)

From Table 4-6, the HD of the shaded path has reached zero, this means that a valid solution has been found. The information of the path is cached. As for the other siblings paths that share the same parent path have not reached zero HD, all their information are discarded as exploring on these paths will not yield to a better solution. The reason is that these paths still require further reduction to find a solution, in other word it requires more gate count and will not yield to a better solution. Hence, the algorithm discards these paths and continues to explore the next lowest non-zero HD path to find other better solution.

4.4 Synthesis Result

In this section, the result of the proposed synthesis algorithm is presented. All experiments were conducted on a Dell Precision T1600 featuring a Intel Xeon CPU E31280 processor at 3.5GHz, 512kB cache, 16GB RAM, and running on Windows 7 Professional edition. The synthesis algorithm has been implemented using C language running on Ubuntu 10 through Oracle VM VirtualBox.

4.4.1 Three-Variable Based Reversible Functions

To determine the efficient for heuristic based reversible synthesis algorithm, one practise is to examine the synthesis result for all three variable based reversible functions. In this section, the proposed algorithm is set to synthesize all three variable based reversible functions, which are a total of 40,320 functions. Synthesizing in term of gate count and quantum cost for both NCT and m-NCT library gates have been done. Synthesis results are taken for further comparison with other algorithms proposed by different authors. Only the best algorithms classify according to their algorithmic paradigms (search-based, cycle-based, transformation-based and BDD-based) are selected. Table 4-7 shows the results of applying the proposed algorithms to synthesize all three variable based reversible functions in term of gate count with comparison to others. The synthesized circuits are categorized according to its gate count size as listed in Table 4-7. The column "Number of Gates" shows the number of gate count the synthesized circuits are categorized into. The data in the other columns indicate the number of such circuits that were synthesized as indicated by their citations. The ideal case is to have the least amount of gate count for any synthesized reversible function. The results generated by the proposed synthesis algorithm are listed in column "Proposed Work". In the corresponding table, result from the proposed synthesis algorithm is compared with other earlier reported synthesis algorithms and the optimal NCT result reported by Shende et al. [27]. It can be seen that the result produced by the proposed algorithm using NCT gate library are close to the optimal solution and are better than several earlier synthesis algorithms reported by [30, 77, 91]. It also produces better solution than the current best search-based synthesis algorithm reported by Donald et al. [30]. Moreover, with the m-NCT gate library included, the gate count becomes significantly lower. Overall, the algorithm produces a lower average gate count and does not produce any circuits that use more than 8 gates.

Table 4-7. Gate count for all three variable based reversible functions

Number of Gates	Proposed Work NCT	Proposed Work m-NCT	Donald et al. [30] NCT	Maslov et al. [77] NCT	Maslov et al. [78] NCT	Yexin et al. [91] NCT	Optimal Solution Shende et al. [27] NCT
0	1	1	1	1	1	1	1
1	12	21	12	12	12	12	12
2	102	225	102	102	102	90	102
3	625	1527	625	567	625	476	625
4	2702	6058	2642	2125	2780	1833	2780
5	7932	14139	7479	5448	8819	4996	8921
6	14384	14995	13596	9086	16953	9126	17049
7	12201	3273	12476	9965	10367	10630	10253
8	2339	81	3351	7274	659	7820	577
9	22	0	36	3837	2	3788	0
10	0	0	0	1444	0	1265	0
11	0	0	0	391	0	258	0
12	0	0	0	62	0	25	0
13	0	0	0	6	0	0	0

Besides synthesizing in term of gate count, the proposed algorithm is also capable of synthesizing reversible function in term of quantum cost. Table 4-8 shows the quantum cost result of the proposed algorithm for all three variable reversible functions. The synthesized circuits are categorized according to its quantum cost size as listed in Table 4-8. The data in the other columns indicate the number of reversible functions that fall in the category of the given quantum cost size. As quantum cost table cannot be found in any of other authors' proposed work, a comparison test cannot be performed. However produced synthesis results are guaranteed to be close to optimal solution. This is because the proposed algorithm checks and verifies the generated quantum cost of each subsection and takes only the path which generates the least quantum cost. Besides that, with the usage of the m-NCT gate library, it helps further improve the quantum cost results.

Table 4-8. Gate count and quantum cost comparison

Quantum Cost Size	Proposed Work NCT	Proposed Work m-NCT
0	1	1
1	9	9
2	51	51
3	187	187
4	387	387
5	426	432
6	305	353
7	350	560
8	1305	1812
9	2952	3458
10	3418	2938
11	1416	1001
12	946	1964
13	3543	5728
14	7278	7851
15	6095	2798
16	1017	856
17	950	2601
18	3319	5048
19	4884	2221
20	1461	64
21	20	0

One of the best features of the proposed algorithm is that it can be set to find minimal quantum cost when synthesize in term of gate count. This ensures that the quantum cost is optimal for the corresponding gate count. The same applies for finding minimal gate count when set to synthesize in term of quantum cost. Table 4-9 shows the comparison of the proposed algorithm with the feature turn on and off when synthesizing for gate count using NCT gate library. Clearly, it can be seen that the added feature provides a better result. However the only drawback for using this feature is that it requires much longer synthesis time.

Table 4-9. Quantum cost result for all three variable based reversible functions

Gate Count / Quantum Cost Size	Synthesize in term of GC with QC minimization		Synthesize in term of GC without QC minimization	
	GC	QC	GC	QC
0	1	1	1	1
1	9	9	9	9
2	51	51	51	51
3	187	187	187	187
4	387	387	387	387
5	426	426	426	426
6	305	305	305	305
7	350	350	350	347
8	1305	1305	1305	1267
9	2952	2946	2952	2753
10	3418	3388	3418	2981
11	1416	1377	1416	1088
12	946	937	946	915
13	3543	3543	3543	3483
14	7278	7246	7278	6847
15	6095	5945	6095	5176
16	1017	949	1017	754
17	950	980	950	1173
18	3319	3358	3319	3765
19	4884	4893	4884	5177
20	1461	1463	1461	1478
21	20	62	20	122
22	0	138	0	537
23	0	74	0	846
24	0	0	0	233
25	0	0	0	0
26	0	0	0	0
27	0	0	0	5
28	0	0	0	7

4.4.2 Benchmark Functions

After experiment on all three variable based functions, the next step is to use the proposed algorithm to synthesize practical reversible benchmark functions. All benchmark functions are obtained from [37], [38] and [36]. Table 4-10, Table 4-11, Table 4-12 and

Table **4-13** show the synthesis result of the proposed algorithm synthesizing the benchmark functions in term of gate count and quantum cost. The tables are categorized in term of the benchmark function size. All displayed results are the best selected result from the proposed synthesis algorithm having the lowest gate count or quantum cost.

The tables are organized in the following way. The row that connected across the whole table contains the name of the benchmark and the size of the function. The next rows which are constructed with several columns contain the experiment result of the corresponding benchmark. The first column shows the gate library used to operate the simulation. The next two columns show the gate count and quantum cost result. The next column contains the circuit connection of the benchmarks determined from the algorithm. The notation for circuit connection follows the following format tx $b_1, .., b_n, b_{n+1}$, where t denotes Toffoli based gates, x denotes the Toffoli gate size and $b_1, b_2, \ldots, b_n, b_{n+1}$ are coefficients of control lines and b_{n+1} is the target bit. If a coefficient is a negative control line, a stroke will be given. Example: Toffoli4 with lines a, c, d as control lines, d is a negative control line and b as target line, the written notation is t4 a, c, d', b.

Table 4-10. Synthesis result of reversible benchmark functions of size 3

Gate Library	GC	QC	Circuit Connection
3_17 (size = 3)			
NCT	6	12	t1 a; t2 a,c; t2 c,b; t3 b,c,a; t3 a,b,c; t1 a
m-NCT	5	13	t2 a,c; t1 c; t3 b',c,a; t2 c,b; t3 a',b,c
ham3 (size = 3)			
NCT	5	9	t3 b,c,a; t2 b,c; t2 c,b; t2 a,b; t2 b,c
nth_prime3_inc (size = 3)			
NCT	4	6	t2 c,b; t3 a,b,c; t2 a,b; t2 b,a

Table 4-11. Synthesis result of reversible benchmark functions of size 4

Gate Library	GC	QC	Circuit Connection
4_49 (size = 4)			
NCT	13	59	t1 a; t3 a,c,d; t2 a,d; t2 d,b; t4 a,b,d,c; t4 b,c,d,a; t3 a,b,d; t2 c,b; t3 b,c,a; t3 a,d,c; t1 a; t3 b,d,a; t3 a,c,b
NCT	14	36	t1 a; t2 c,b; t2 d,b; t2 a,c; t2 a,d; t3 c,d,a; t3 a,b,c; t3 a,c,d; t3 b,d,a; t3 a,d,b; t3 b,c,a; t2 c,b; t2 a,b; t1 a
m-NCT	9	30	t2 c,a; t3 a’,b’,d; t2 d,a; t3 c,d,b; t3 a’,d,c; t3 b,c,a; t3 a’,b,d; t2 d,b; t2 d,c

4b15g_1 (size = 4)			
NCT	15	45	t2 a,c; t2 c,d; t2 b,a; t2 a,b; t3 c,d,a; t2 c,b; t3 a,d,c; t1 a; t2 c,d; t4 a,b,c,d; t3 a,b,c; t2 b,a; t3 a,d,b; t3 b,c,a; t2 b,d
NCT	18	44	t3 a,c,b; t2 d,b; t1 d; t2 a,b; t2 b,a; t2 b,c; t2 a,d; t3 b,d,a; t3 a,c,b; t2 c,a; t2 b,a; t2 d,a; t2 a,c; t2 a,d; t3 c,d,a; t4 a,b,c,d; t1 c; t2 b,c
m-NCT	13	49	t3 a',d,b; t2 b,a; t3 a,c,b; t2 d,c; t4 b,c,d,a; t2 a,c; t3 b',c,a; t3 a,c,d; t2 d,c; t2 b,d; t3 c',d,a; t3 a,c',b; t1 a
m-NCT	16	42	t3 a,c,b; t2 a,c; t2 c,d; t1 b; t2 d,c; t2 c,a; t3 a',b,c; t2 d,a; t2 b,d; t3 c,d,a; t2 b,c; t2 a,b; t2 a,d; t3 b,d',a; t2 c,b; t4 a,b,c,d
4b15g_2 (size = 4)			
NCT	15	53	t3 a,d,c; t4 a,c,d,b; t2 a,d; t2 b,a; t2 a,b; t2 c,b; t1 a; t3 c,d,a; t2 d,c; t4 a,c,d,b; t2 b,c; t3 a,b,d; t2 d,a; t2 c,d; t3 a,d,b
NCT	18	38	t2 d,b; t1 b; t2 a,c; t2 a,d; t3 c,d,a; t2 a,d; t2 b,c; t3 b,c,a; t2 c,d; t3 c,d,a; t3 a,d,b; t2 a,c; t2 d,a; t2 a,d; t3 b,c,d; t3 a,d,c; t2 b,d; t2 a,b;
m-NCT	14	42	t3 b,c,a; t3 b,d,c; t2 a,d; t2 a,b; t2 a,c; t3 c,d',a; t2 a,c; t3 b,c',a; t3 a,c',b; t3 a,d,c; t1 a; t2 d,b; t3 b',c,d; t2 b,d
m-NCT	17	35	t2 d,b; t1 b; t2 a,c; t3 a,c,d; t2 d,a; t2 c,d; t3 a,c',d; t2 a,c; t2 d,a; t3 a,b,d; t2 b,a; t3 a,d,b; t2 d,a; t3 b,c,d; t2 a,b; t2 b,c; t3 b',d,c
4b15g_3 (size = 4)			
NCT	16	44	t2 a,b; t3 a,c,b; t1 a; t2 b,c; t3 c,d,a; t2 a,b; t3 a,d,c; t3 a,b,d; t3 b,c,a; t1 c; t2 a,b; t3 c,d,b; t2 c,d; t3 a,b,c; t3 a,d,b; t2 d,a
m-NCT	14	58	t3 a,b,d; t1 a; t2 c,a; t2 a,b; t3 b,d,a; t2 d,c; t3 a',c,d; t4 a,b,d',c; t3 a,c',b; t2 b,a; t3 c,d,b; t4 a,b',d,c; t1 a; t2 b,c
m-NCT	15	43	t3 a,c',b; t1 a; t2 b,c; t3 c,d,a; t2 a,b; t3 a,d,c; t3 a,b,d; t3 b,c,a; t1 c; t2 a,b; t3 c,d,b; t2 c,d; t3 a,b,c; t3 a,d,b; t2 d,a
4b15g_4 (size = 4)			
NCT	15	45	t2 c,b; t3 b,c,a; t1 a; t2 d,c; t3 b,c,d; t2 a,b; t2 a,d; t2 a,c; t1 a; t4 a,c,d,b; t2 d,a; t3 b,c,d; t3 a,b,c; t3 a,d,b; t2 d,c
m-NCT	14	48	t2 c,d; t3 b',d,c; t2 a,b; t3 b,d,a; t2 c,d; t2 c,a; t3 a',b,d; t2 d,b; t1 b; t3 a,b,c; t2 b,a; t3 a,c,b; t4 a,b',d,c; t3 b,c,a
m-NCT	16	42	t3 b,c,d; t2 b,c; t2 a,b; t2 d,c; t2 d,a; t4 a,b',d,c; t3 c,d,b; t1 c; t2 b,c; t2 a,d; t3 b',c,d; t2 c,a; t3 a,b,c; t2 d,b; t2 d,c; t1 d
4b15g_5 (size = 4)			
NCT	15	37	t3 c,d,b; t2 d,a; t1 a; t3 a,c,d; t2 b,c; t2 d,c; t3 b,c,a; t2 a,b; t3 a,b,c; t2 c,a; t3 c,d,b; t3 a,b,d; t1 a; t2 d,a; t1 c
m-NCT	15	43	t2 c,d; t3 a,d,b; t2 b,d; t2 c,a; t3 a',b,c; t3 b,d,c; t2 a,c; t2 c,b; t1 b; t2 b,a; t3 a',d,b; t3 a,c,d; t3 b,d,a; t3 b,c',d; t2 a,d
aj-e11 (size = 4)			
NCT	11	29	t2 a,c; t2 b,d; t2 b,c; t3 a,d,c; t2 a,d; t2 c,b; t4 b,c,d,a; t3 a,c,b; t2 b,d; t2 d,c; t1 d
m-NCT	10	28	t2 a,c; t2 b,d; t2 b,c; t3 a,d,c; t2 a,d; t4 b',c,d,a; t3 a',c,b; t2 b,d; t2 d,c; t1 d
decod24 (size = 4), embedding with constants 00			
NCT	6	16	t3 c,d,a; t2 c,d; t3 c,d,b; t3 a,c,d; t2 d,c; t1 d
m-NCT	5	17	t3 c,d',b; t3 c,d,a; t3 a',c,d; t2 d,c; t1 d
decod24 (size = 4), embedding with constants 01			
NCT	7	29	t4 b,c,d,a; t3 c,d,b; t3 b,c,d; t3 c,d,b; t2 d,c; t1 b; t1 d
m-NCT	7	31	t3 c,d',b; t4 b,c,d,a; t3 b',c,d; t3 c,d',b; t2 d,c; t1 b; t1 d
decod24 (size = 4), embedding with constants 10			
NCT	6	18	t3 c,d,a; t3 a,c,d; t3 a,c,b; t2 d,c; t1 a; t1 d
m-NCT	7	19	t3 c,d,a; t3 a,c,b; t3 a',c,d; t2 c,b; t2 d,c; t1 a; t1 d
decod24 (size = 4), embedding with constants 11			
NCT	7	35	t4 b,c,d,a; t4 a,b,c,d; t3 a,c,b; t2 d,c; t1 b; t1 a; t1 d

hwb4 (size = 4)			
NCT	11	23	t2 c,d; t2 d,b; t3 a,c,d; t2 b,c; t2 a,b; t3 b,d,a; t3 a,c,d; t2 c,b; t2 b,a; t2 d,b; t2 a,c
m-NCT	11	27	t3 a,c,d; t3 a,c',b; t2 d,a; t2 c,d; t3 b,d',c; t2 a,d; t2 b,a; t3 a,c,b; t2 d,c; t2 c,a; t2 b,d
imark (size = 4)			
NCT	7	19	t3 c,d,a; t3 a,b,d; t2 d,a; t2 b,c; t2 d,c; t3 a,c,b; t1 c
majority3 (size = 4)			
NCT	4	14	t2 c,a; t3 a,c,b; t3 a,b,c; t3 b,c,a
m-NCT	4	16	t3 a',c,b; t2 c,a; t3 a,b,c; t3 b,c,a
mini_alu (size = 4)			
NCT	6	46	t4 b,c,d,a; t3 a,d,c; t3 a,d,b; t4 b,c,d,a; t3 b,c,d; t3 a,d,c
NCT	8	16	t3 b,c,a; t2 b,c; t3 a,d,b; t2 d,a; t2 b,c; t3 b,c,d; t2 d,a; t3 a,d,c
mperk (size = 4)			
NCT	12	20	t2 d,c; t2 a,d; t2 b,a; t2 a,b; t3 b,c,d; t2 d,b; t3 b,c,a; t2 d,a; t2 c,b; t2 b,c; t1 b; t2 b,a
m-NCT	9	17	t2 d,c; t3 a',c',d; t2 b,a; t1 c; t2 d,a; t2 a,b; t3 b',c,a; t2 c,d; t2 b,c
nth_prime4_inc (size = 4)			
NCT	13	51	t4 a,b,d,c; t3 a,c,b; t2 d,c; t2 d,b; t3 b,c,d; t3 b,d,c; t2 c,b; t3 a,b,c; t2 a,b; t2 b,a; t4 b,c,d,a; t2 d,b; t2 d,c
NCT	14	38	t3 a,b,c; t2 a,b; t3 b,c,d; t3 a,d,b; t2 b,a; t2 c,b; t2 d,c; t2 c,d; t3 b,c,d; t3 a,c,b; t2 d,a; t3 a,d,c; t3 b,c,a; t2 d,c
m-NCT	10	54	t3 a',c,b; t2 d,c; t3 b',c,d; t3 a',b,c; t3 c',d,b; t2 b,c; t2 a,b; t3 b,d',a; t4 b',c,d,a; t4 a,b,d,c
m-NCT	13	37	t3 a,b,c; t2 a,b; t3 b,c,a; t3 a,d,b; t2 b,a; t2 c,b; t2 d,c; t3 b',c,d; t3 a,c,b; t2 d,a; t3 a,d,c; t3 b,c,a; t2 d,c
oc5 (size = 4)			
NCT	12	42	t2 a,c; t2 c,a; t2 a,c; t3 b,d,a; t3 a,d,b; t3 b,c,a; t1 c; t2 d,b; t3 a,b,d; t3 b,c,d; t2 c,b; t4 a,b,d,c
NCT	13	39	t2 d,b; t2 c,a; t2 a,c; t1 c; t3 a,d,b; t2 c,a; t3 a,b,d; t2 b,a; t3 a,d,b; t1 a; t3 b,c,a; t2 c,b; t4 a,b,d,c
m-NCT	11	67	t3 c,d,b; t2 c,a; t3 a',b',c; t2 b,c; t3 b',c',a; t3 a',d,b; t4 b',c,d,a; t3 b,c,d; t4 a,b,d',c; t2 c,b; t4 a,c,d,b
m-NCT	14	40	t3 b,d,c; t2 c,b; t2 c,d; t3 c,d,b; t2 c,d; t3 a,b',c; t2 c,a; t2 d,b; t3 b,c',d; t2 a,c; t2 c,b; t4 a,b',d,c; t1 b; t1 c
oc6 (size = 4)			
NCT	13	41	t3 c,d,a; t2 d,c; t2 c,b; t2 c,d; t3 a,b,c; t3 b,c,a; t3 a,d,c; t1 a; t4 a,c,d,b; t2 b,a; t3 a,b,d; t2 c,a; t2 a,d
m-NCT	12	38	t2 c,a; t2 d,c; t3 b,c,d; t3 a,d,b; t4 a,b,d,c; t3 b,c,a; t2 c,b; t2 b,d; t1 a; t3 a',d,c; t2 a,d; t2 b,a
oc7 (size = 4)			
NCT	13	37	t3 a,b,c; t2 b,d; t1 b; t2 c,a; t3 b,d,c; t3 a,d,b; t2 a,d; t4 a,b,c,d; t1 a; t2 b,a; t2 c,d; t3 c,d,b; t2 b,c
m-NCT	13	49	t3 b,d,c; t1 b; t3 b,d,a; t3 c,d',b; t4 a,c,d,b; t3 a,c,d; t2 b,a; t3 a,b',c; t2 d,c; t1 a; t2 a,d; t3 c,d',a; t2 b,c
m-NCT	14	47	t3 a',c,b; t2 b,d; t2 a,d; t1 b; t4 a,b,d,c; t2 d,a; t3 a,b,d; t3 a,c',b; t2 c,a; t3 a,d',c; t1 a; t2 b,a; t2 c,b; t3 a',b,c; t2 d,b
oc8 (size = 4)			
NCT	13	49	t2 a,d; t2 a,b; t2 d,a; t3 b,c,d; t3 a,c,b; t4 a,b,d,c; t3 b,d,a; t3 a,d,b; t1 d; t3 a,b,d; t1 b; t2 d,a; t3 b,c,d
m-NCT	11	45	t2 c,b; t4 a',b,d,c; t3 c',d,b; t3 a',c',d; t3 a,b,d; t2 d,a; t2 c,d; t3 a',b,d; t3 a',d',b; t2 c,a; t1 b
m-NCT	15	43	t3 c,d,b; t1 a; t2 a,d; t2 b,d; t2 a,c; t3 a,d,b; t1 b; t2 b,c; t4 a,b,d,c; t2 b,a; t3 c,d',b; t1 b; t1 d; t3 b,c,d; t2 a,c

primes4 (size = 4)			
NCT	11	29	t3 c,d,b; t2 b,c; t2 c,a; t2 d,c; t2 c,b; t3 b,c,d; t1 b; t3 a,c,b; t2 d,a; t3 b,c,a; t3 b,d,c
NCT	12	26	t3 c,d,b; t2 b,c; t1 b; t2 d,c; t3 b,c,d; t2 b,a; t2 d,a; t3 a,c,b; t2 c,a; t2 b,a; t2 d,a; t3 b,d,c
m-NCT	9	41	t2 d,c; t2 c,a; t3 b',c,d; t2 b,c; t4 a,b',d,c; t3 a,c,b; t4 a',c,d,b; t2 b,a; t1 b
m-NCT	11	27	t3 c,d',b; t2 b,d; t2 d,c; t2 b,a; t3 a,d,b; t2 b,a; t2 c,b; t2 c,d; t3 b,d,c; t3 b,c',d; t1 b
rd32 (size = 4)			
NCT	4	12	t3 a,b,d; t2 a,b; t3 b,c,d; t2 b,c
NCT	5	9	t2 b,d; t2 b,c; t2 a,b; t3 b,c,d; t2 a,c

Table 4-12. Synthesis result of reversible benchmark functions of size 5

Gate Library	GC	QC	Circuit Connection
4mod5 (size = 5) with ancilla input 0			
NCT	5	9	t1 a; t2 a,c; t2 c,b; t2 d,b; t3 b,c,e
m-NCT	4	12	t1 d; t2 b,d; t3 c,d,e; t3 a',d,e
4mod5 (size = 5) with ancilla input 1			
NCT	5	7	t2 a,c; t2 b,d; t3 c,d,e; t2 c,d; t2 d,e
m-NCT	5	9	t2 a,c; t2 b,d; t3 c,d',e; t2 c,d; t2 c,d
alu (size = 5), with 1st line is function output			
NCT	6	14	t2 c,b; t2 d,e; t3 b,d,c; t2 c,a; t1 c; t3 a,e,c
m-NCT	7	15	t2 c,b; t2 a,c; t3 b,d,c; t2 d,e; t3 c,e',a; t1 e; t1 a
4mod5 (size = 5) with ancilla input 0			
NCT	5	9	t1 a; t2 a,c; t2 c,b; t2 d,b; t3 b,c,e
m-NCT	4	12	t1 d; t2 b,d; t3 c,d,e; t3 a',d,e
majority5 (size = 5)			
NCT	18	118	t3 d,e,a; t4 a,c,e,d; t4 a,b,c,d; t4 a,c,e,b; t2 e,d; t2 d,c; t3 a,d,b; t3 c,d,a; t3 b,d,c; t3 c,d,b; t4 a,b,c,d; t2 e,c; t2 d,b; t3 c,d,e; t4 a,c,e,b; t2 e,d; t4 a,b,d,e; t3 c,e,d
m-NCT	14	96	t3 d,e,a; t2 e,d; t4 a,d',e,b; t3 a',d,b; t4 a,b,c,d; t3 c',d,a; t3 b,d,c; t4 a,b,c,d; t2 e,c; t3 d,c,e; t4 a',d,e,b; t2 e,d; t4 a,b,d,e; t3 c,e,d
nth_prime5_inc (size = 5)			
NCT	28	166	t4 b,d,e,a; t3 a,c,d; t2 e,d; t2 d,e; t4 a,c,d,e; t3 d,e,a; t3 b,e,a; t4 a,d,e,b; t2 c,a; t2 e,c; t4 b,d,e,c; t4 b,c,d,e; t3 b,c,a; t3 b,c,d; t2 a,b; t4 a,b,d,c; t2 b,a; t3 b,c,a; t3 c,d,b; t3 a,d,b; t2 d,a; t4 a,d,e,c; t4 b,c,e,d; t2 a,c; t2 d,c; t3 c,e,a; t3 a,b,c; t3 b,e,d
m-NCT	27	165	t4 b,d,e,a; t3 a,c,d; t2 e,d; t2 d,e; t4 a,c,d,e; t3 d,e,a; t3 b,e,a; t4 a,d,e,b; t2 c,a; t2 e,c; t4 b,d,e,c; t4 b,c,d,e; t3 b,c,a; t3 b,c,d; t2 a,b; t4 a,b,d,c; t3 b,c',a; t3 c,d,b; t3 a,d,b; t2 d,a; t4 a,d,e,c; t4 b,c,e,d; t2 a,c; t2 d,c; t3 c,e,a; t3 a,b,c; t3 b,e,d
xor5 (size = 5)			
NCT	4	4	t2 a,e; t2 b,e; t2 c,e; t2 d,e

Table 4-13. Synthesis result of reversible benchmark functions of size 6 or greater

Gate Library	GC	QC	Circuit Connection
graycode6 (size = 6)			
NCT	5	5	t2 b,a; t2 c,b; t2 d,c; t2 e,d; t2 f,e
mod5adder (size = 6)			
NCT	35	255	t3 c,e,b; t3 d,f,b; t4 a,b,d,c; t3 a,f,c; t3 a,f,b; t3 b,f,a; t3 d,f,a; t4 a,b,f,c; t3 c,f,b; t4 a,b,e,c; t4 a,d,e,c; t4 c,e,f,b; t3 b,e,c; t3 c,d,a; t4 a,e,f,c; t3 d,f,c; t4 b,c,e,a; t2 e,b; t3 e,f,b; t3 d,f,b; t4 a,b,d,c; t4 b,c,d,a; t2 e,a; t3 b,e,a; t3 c,d,b; t3 d,f,a; t3 b,f,a; t4 a,b,d,c; t4 c,e,f,b; t3 a,d,b; t3 e,f,a; t4 a,c,d,b; t2 d,a; t2 f,c; t3 e,f,c
m-NCT	16	112	t3 b',f,c; t3 b,e',c; t4 a,b,e,c; t4 a,b',f,c; t4 a',c,e,b; t4 b,c,e,a; t2 b,c; t4 a,c,e,b; t3 a',d,b; t3 c,d,a; t2 e,b; t4 a,b,d,c; t3 c',f,a; t2 d,a; t2 d,b; t3 a,f,b
ham7 (size = 7)			
NCT	22	74	t4 e,f,g,c; t2 g,d; t2 c,a; t2 g,c; t2 b,a; t3 f,g,b; t3 e,f,g; t2 g,f; t4 e,f,g,d; t4 e,f,g,b; t2 e,g; t2 g,e; t3 f,g,e; t2 c,f; t2 a,c; t2 d,c; t3 e,g,f; t2 b,e; t2 c,b; t2 d,g; t2 b,d; t2 a,b
m-NCT	21	73	t3 e,g,d; t2 c,a; t2 d,c; t4 e,f',g,b; t2 b,a; t4 e,f',g,d; t3 e,g,a; t3 e',g,d; t3 e,f',g; t3 f,g,b; t3 f',g,e; t2 e,f; t3 f,g,e; t2 d,f; t2 c,f; t2 a,c; t2 b,e; t2 c,b; t2 d,g; t2 b,d; t2 a,b

4.5 Chapter Summary

In conclusion, a synthesis algorithm that uses the NCT library gate with mixed-polarity control is presented. The proposed algorithm synthesizes any reversible function using its PPRM expansions and applies the HD approach to select the transformation path. For the selected path, a variety of reversible gates are selected through finding possible matching reversible gate. The proposed algorithm does not generate extra ancilla input, so it is able to keep the garbage output at minimum. The proposed algorithm has been applied to synthesize all three variable reversible functions and has shown to obtain good result. Result of using the proposed algorithm to synthesize reversible benchmark functions has also been provided. From the experimental results, it has been shown that with the m-NCT gate library added, the results are greatly improved.

CHAPTER 5 THE PROPOSED REVERSIBLE GATE AND ITS APPLICATION AS FULL ADDER CIRCUIT

Full adder circuit is an essential fundamental building block in most computational arithmetic units. After reversible logic is introduced to digital design, reversible logic implementation on full adder circuit has been reported in several works. In this chapter, a novel reversible gate acting as a reversible full adder circuit is presented. An overview is given in Section 5.1; the Boolean function of the proposed reversible gate is derived in Section 5.2. The design methodology of the proposed gate is discussed in Section 5.3. In Section 5.4, functional verification of the proposed gate is discussed. Comparison of the proposed gate with the existing literature work is discussed in Section 5.5 and conclusion is given in Section 5.6.

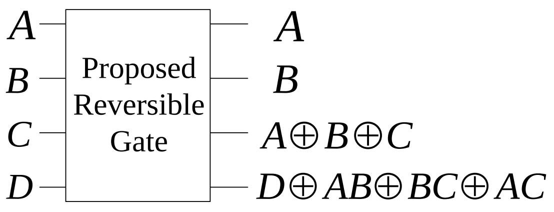
5.1 Overview

With the introduced of reversible logic to design digital arithmetic logic unit, reversible logic based implementation of full adder circuit has been proposed by various authors such as [26, 62, 95-99]. Over the time, better synthesis tool are proposed and much efficient reversible gate are introduced to achieve better synthesize results [43]. Reversible based implementation on full adder circuit has also been improved over the time. Design parameters such as ancilla input, garbage output, quantum cost and delay had been widely addressed and exploited. Although various research works has been done, however none of the existing work is capable in achieving 0 garbage output. In this work, a novel reversible logic implementation on full adder circuit that achieves 0 garbage output is presented. Other important parameters such as ancilla input, quantum cost and delay are kept in their lowest possible value.

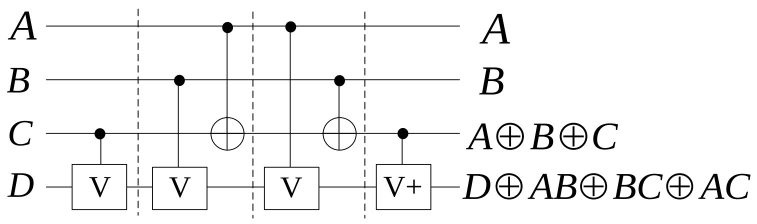
5.2 Proposed Reversible Gate

The proposed reversible full adder gate is a four variable reversible gate. It has the inputs to outputs mapping of ($P = A$, $Q = B$, $R = A \oplus B \oplus C$, $S = D \oplus AB \oplus BC \oplus AC$).

Figure 5.1 (a) shows the gate symbol of the proposed gate and Figure 5.1 (b) shows the quantum realization disclosed. Subsequently, Table 5-1 shows truth table of the proposed gate. From Figure 5.1 (b), the proposed gate is constructed using 3 Controlled-V+ gates, 1 Controlled-V gate and 2 CNOT gates which produces a total quantum cost of 4. The logic depth of the proposed gate is 4 as seen in Figure 5.1 (b) which resulting in a propagation delay of Δ4. The objective of this proposed gate is to serve as a reversible full adder circuit with 0 garbage output. With the gate input C feed to a constant ancilla value 0, the gate is able to implement the full adder Boolean function $A \oplus B \oplus C$ and $AB \oplus BC \oplus AC$. Besides that, the proposed gate can also be used to serve other reversible logic design and synthesis purposes.



(a)



(b)

Figure 5.1. (a) The proposed revesible gate (b) Quantum realization of the proposed revesible gate

Table 5-1. Truth table of the proposed reversible gate

Input A, B, C, D	Output P, Q, R, S
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 0
0 0 1 1	0 0 1 1
0 1 0 0	0 1 1 0
0 1 0 1	0 1 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 1

1 0 1 1	1 0 0 0
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

5.3 Design Methodology

To begin, the Boolean expressions of a typical full adder circuit is analyzed in Equation (5.1) and Equation (5.2) and its corresponding truth table is shown in Table 5-2. In both equations, A and B denote the two binary numbers to be summed, C_{in} is the carry input, S is the binary sum and C_{out} is the carry output.

$$S = A \oplus B \oplus C_{in} \tag{5.1}$$

$$C_{out} = AB \oplus (A \oplus B)C_{in} \tag{5.2}$$

Table 5-2. Full adder function

Input A, B, C_{in}	Output S, C_{out}
0 0 0	0 0
0 0 1	1 0
0 1 0	1 0
0 1 1	0 1
1 0 0	1 0
1 0 1	0 1
1 1 0	0 1
1 1 1	1 1

From the truth table as shown in Table 5-2, the full adder function has three inputs but it has only two outputs, thus the expression is not reversible. For the expression to be reversible, one extra output is added to balance between the input vectors and the output vectors. However for the input arguments 001, 010 and 100, they are mapped to the same output argument 10. Even with an extra garbage output added, two of the input arguments will still be mapped to the same output and hence the function is not reversible. To overcome that, at least one ancilla input and an extra output are required. To keep garbage output minimal, two of the inputs are directly mapped to their output and use an ancilla input of constant value 0 to create a unique one-to-one mapping between the inputs and outputs. The resulting truth table of the proposed reversible full adder is shown in Table 5-3. Since the output P and Q are mapped directly from their respective input, it does not contribute any garbage output [46, 47]. Overall, the reversible function of the proposed full adder uses only 1 ancilla input and has 0 garbage output.

Table 5-3. Truth table of the proposed reversible full adder

Input A, B, C, D	Output P, Q, R, S
0 0 0 0	0 0 0 0
0 0 1 0	0 0 1 0
0 1 0 0	0 1 1 0
0 1 1 0	0 1 0 1
1 0 0 0	1 0 1 0
1 0 1 0	1 0 0 1
1 1 0 0	1 1 0 1
1 1 1 0	1 1 1 1

For easier representation, the inputs and outputs are assigned as (A, B, C, D) and (P, Q, R, S). The logical expression of the final reversible full adder can be expressed as follow:

$$P = A \tag{5.3}$$

$$Q = B \tag{5.4}$$

$$R = A \oplus B \oplus C \tag{5.5}$$

$$S = D \oplus AB \oplus BC \oplus AC \tag{5.6}$$

To realize the reversible function into reversible circuit, the developed reversible logic synthesis algorithm in Chapter 4 is used. The settings for the algorithm are set to optimize in terms of gate count and quantum cost using NCT gate library. The resulting reversible circuit generated by the synthesis algorithm is shown in Figure 5.2. The overall circuit uses 1 ancilla input, 0 garbage output and has a total quantum cost of 8 and a delay of ∆ 8.

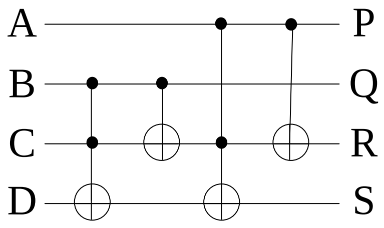
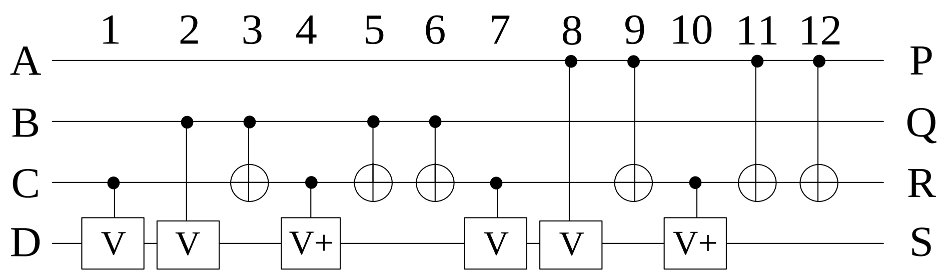
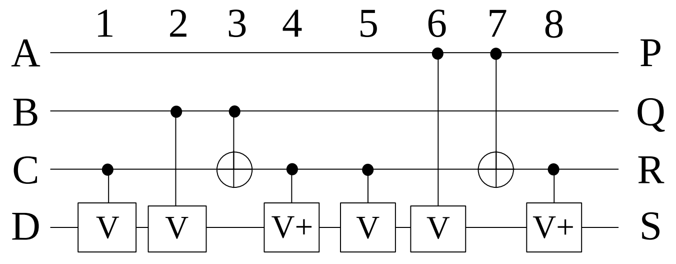


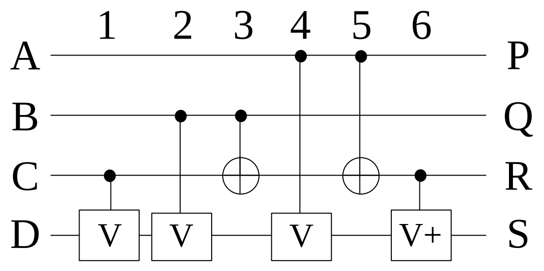
Figure 5.2. Realized reversible full adder



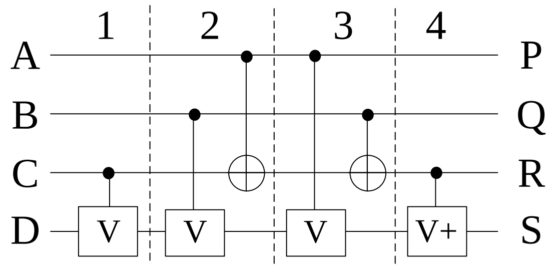
(a)



(b)



(c)

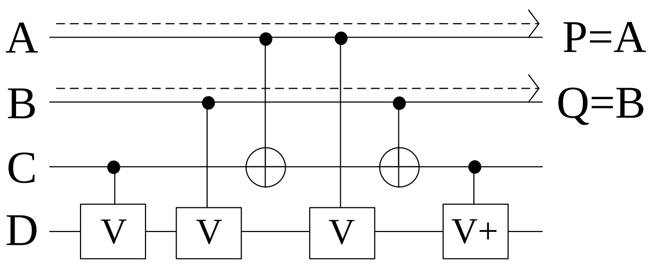


(d)

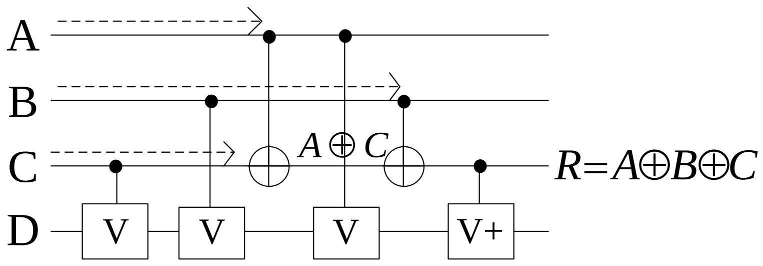
Figure 5.3. Optimization of the reversible full adder circuit in primitive gate form

To optimize the circuit to its minimal form, all the gates in Figure 5.2 are first decomposed into their primitive gate form. The resulting circuit is shown in Figure 5.3 (a). Observing from Figure 5.3 (a), gates 5, 6 and gates 11, 12 are repeated terms and can be deleted because their function offset one another. After reduction, the circuit can be drawn as shown in Figure 5.3 (b). It can be noticed from Figure 5.3 (b) that gates 4 and 5 can be removed as their operation are inverse to one another by following the rules $V^{+} \cdot V = I$ [31]. After reduction, the circuit can be drawn as shown in Figure 5.3 (c). At this stage, the overall circuit uses 1 ancilla input, 0 garbage output and has a total quantum cost of 8 and a delay of Δ 8. Next, the full adder circuit is further optimized in terms of delay by applying the moving rules described in [99]. The moving rules are applied by grouping all parallel connected gates together and the delay measurement for the grouped section are determined by the highest delay gate. The resulting circuit after applying the moving rules is as shown in Figure 5.3 (d). Based on Figure 5.3 (d), the number of levels in the compacted circuit is 4 and thus the delay of the circuit is Δ 4. The overall circuit generated at this stage uses 1 ancilla input, 0 garbage output and has a total quantum cost of 6 and a delay of Δ 4. With no possible reduction available, the final circuit is achieved.

5.4 Quantum Realization Verification of the Reversible Chua Gate



(a)



(b)

Figure 5.4. Verification of output P, Q, R

To ensure that the final circuit follows the specification as described in Equation (5.3) – Equation (5.6), Boolean verification on the final circuit obtained in Figure 5.3 (d) is performed. Verification of output P and Q are direct since the input A, B are connected directly to the output as shown in Figure 5.4 (a). For the output R, the output is the XOR of input A, B and C as shown in Figure 5.4 (b). For the output S, the path consists of the Controlled-V and Controlled-V+ gates, so direct computation of the output is not possible. To verify the output S, the verification has to be done in its truth table format. With the input A, B, C varying from 000 to 111, the truth table of the function is derived in Table 5-4.

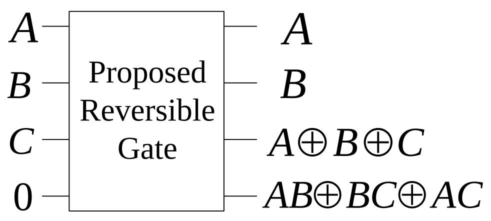
Table 5-4. Truth table of output S

Input A, B, C	Output S
0 0 0	D
0 0 1	$D \cdot V \cdot V^+ = D$
0 1 0	$D \cdot V \cdot V^+ = D$
0 1 1	$D \cdot V \cdot V = \overline{D}$
1 0 0	$D \cdot V \cdot V^+ = D$
1 0 1	$D \cdot V \cdot V = \overline{D}$
1 1 0	$D \cdot V \cdot V = \overline{D}$
1 1 1	$D \cdot V \cdot V \cdot V \cdot V^+ = \overline{D}$

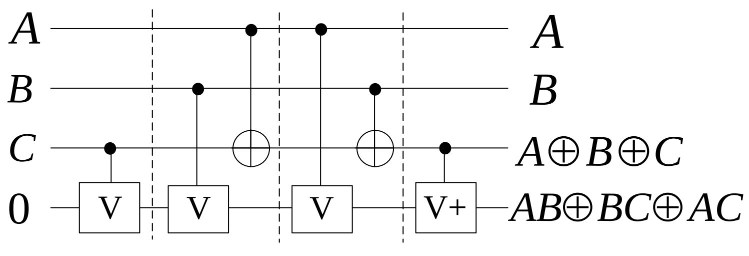
By comparing the derived output result with the expected result in Table 5-3, the quantum realization of the proposed reversible gate is verified.

5.5 Comparison of Reversible Full Adder Circuit

The proposed gate can be acted as a reversible full adder circuit with setting the input D to 0 as seen in Figure 5.5 (a) and Figure 5.5 (b).



(a)



(b)

Figure 5.5. (a) The propsed reversible gate as reversible full adder circuit (b) Quantum realization of the propsed reversible gate as reversible full adder circuit

Table 5-5 shows the comparison of reversible full adder circuits. By comparing the ancilla input, the best ancilla input is 1 and has been proven to be optimal for a reversible based implementation of a reversible full adder circuit. Quantum cost and delay of the proposed adder are the same as the existing work proposed by Maslov et al. [99]. However, the proposed adder has an advantage in terms of garbage output. Since, the proposed adder uses no garbage output and other parameters are also kept at minimal, thus it is shown to be more efficient compared to all the current existing works in literature.

Table 5-5. Comparison of different reversible full adder circuit designs

Reversible Full Adder Circuit	Ancilla Input	Garbage Output	Quantum Cost	Delay ∆
Proposed Work	1	0	6	4
Khlopotine et al. [95] and Miller et al. [26]	1	1	12	12
Bruce et al. [62]	2	2	20	20
James et al. [96]	4	3	25	15
Islam et al. [97]	1	1	8	8
Mohammadi et al. [98] and Nayeem et al. [100]	1	1	6	5
Maslov et al. [99]	1	1	6	4

5.6 Chapter Summary

In conclusion, a novel design of a reversible gate which is specially designed for the use as a reversible full adder circuit is presented. Using the proposed gate as a reversible full adder circuit, it performances are better than all the existing designs in literature in terms of garbage output and other parameters such as ancilla input, quantum cost and delay are in pair with the current best existing ones. The proposed reversible gate will be beneficial for any reversible logic arithmetic based design that requires the use of a reversible full adder.

CHAPTER 6 DESIGN OF MINIMAL DELAY REVERSIBLE BCD ADDER

With the increasing demand of decimal arithmetic and reversible logic, reversible based implementation of reversible binary coded decimal (BCD) adder has received significant research attention. In this chapter, several designs of reversible implemented BCD adder is addressed in terms of propagation delay. An overview of the chapter is presented in Section 6.1. Section 6.2 discusses about the basic of a BCD adder and the proposed BCD adder designs. Design methodologies of the proposed reversible BCD adders are addressed in Section 6.3 and Section 6.4. Evaluation of the proposed BCD adders with existing literature work is discussed in Section 6.5. Simulation and verification of the proposed designs are discussed in Section 6.6 and a conclusion is drawn in Section 6.7.

6.1 Overview

With the increasing demand of decimal arithmetic, reversible based implementation of reversible binary coded decimal (BCD) adder has received significant research attention. Over the last decades, numerous research work on reversible BCD adders had been reported, for example, [64, 96, 98, 103-106] to reduce quantum cost, garbage output or ancilla input are considered for optimization. Balancing between these parameters is a challenging task for researchers and still remains as an open problem. Till now, optimization for propagation delay to design faster reversible BCD adder has rarely been studied except in [96], [106] and [64]. As fast decimal arithmetic approach is gaining popularity in the computing community, propagation delay should also be included in reversible BCD adder design.

In this chapter, a reversible BCD adder design primarily optimized for propagation delay is presented. As the optimization in terms of propagation delay may degrade the result of other parameters such as ancilla input, garbage output and quantum cost, thus these parameters are also considered into optimization and are kept in acceptable range. Two BCD adder concepts are presented, one with carry input C_{in}

and the other without C_{in}. For each of them, two designs are made, one without using mixed-polarity Toffoli gate and another with mixed-polarity Toffoli gate.

6.2 BCD Adder

In this work, two BCD adder designs are presented, one with carry input C_{in} and the other without carry input C_{in}. Before going into the details of the two designs, basic of BCD adder is discussed.

6.2.1 Basic

A BCD adder is a circuit that sums two decimal numbers and produces the sum in decimal point. The two input decimal numbers are represented in a 4-bit binary number range from 0000-1001 representing decimal value of 0 to 9. Figure 6.1 shows an example of traditional BCD adder circuit. The two decimal digits (a_3, a_2, a_1, a_0) and (b_3, b_2, b_1, b_0) and the carry input C_{in} are firstly added in the top 4-bit binary adder which result is a binary sum (S_3, S_2, S_1, S_0) and its carry output C_3. The binary sum produced by the top 4-bit binary adder is compatible when the sum output is equal to or less than 1001 (decimal number 9) as the BCD number is the same as the binary number. But when the binary sum is greater than 1001, the top 4-bit binary adder represents the output in hexadecimal format and hence a correction is required. To correct the result, a binary sum of 0110 is added to convert the output into correct BCD number (D_3, D_2, D_1, D_0).

In circuit, the correction is done by adding a detection circuit and a 4-bit binary adder to the original 4-bit binary adder. The detection circuit is added to check if the binary sum is greater than 1001 and can be expressed in the following Boolean function $C_3 + S_3(S_2 + S_1)$. Since the detection circuit only flags when the binary sum is greater than 1001, its operation is the same as the BCD carry output C_{out}. Hence, the function can be written as $C_{out} = C_3 + S_3(S_2 + S_1)$. When $C_{out} = 1$, binary 0110 is added to the original binary sum which is done by the final 4-bit binary adder.

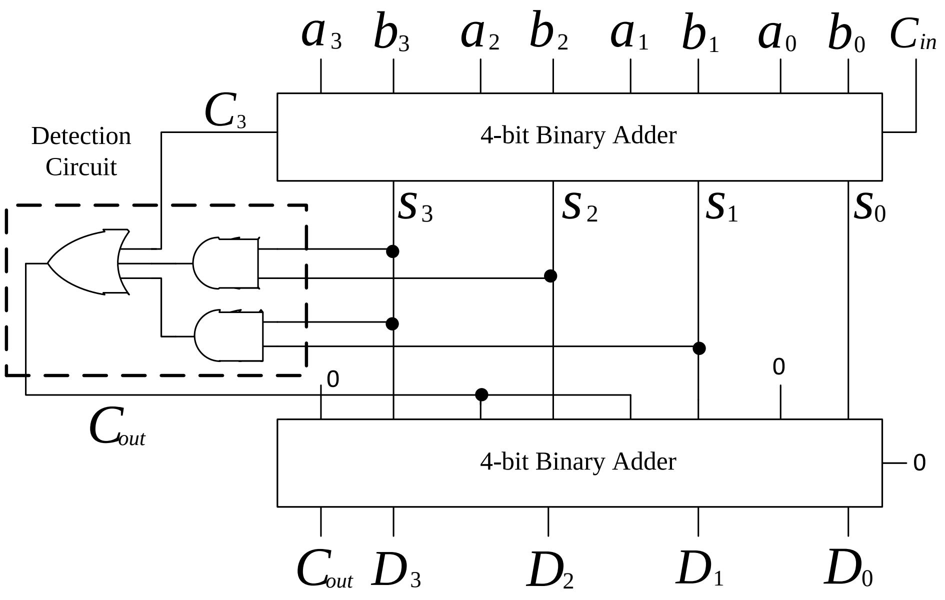


Figure 6.1. BCD Adder Circuit

6.2.2 Proposed BCD Adder Design with Carry Input

In BCD adder, the carry output C_{out} generated by the lowest digit of a BCD adder needs to propagate through all the intermediate adders until it reaches to the most significant adder digit. As the BCD adder digit increases up to n digit, the propagation delay also increases accordingly. To reduce delay and maximize speed, the irreversible BCD adder design illustrated in Figure 6.1 has been redesigned using the following ways:

1. Carry input C_{in} to the BCD adder is designed to be fed in after the 4-bit binary adder. This method allows the 4-bit binary adder of the n digit BCD adder to execute simultaneously.
2. The required carry output C_{out} of the BCD adder is designed to generate faster before even the BCD sum is completed. This allows faster feeding of the carry output C_{out} to the next BCD adder digit.

The redesigned BCD adder with the above modification made is as shown in Figure 6.2. Noticed that the carry input C_{in} and the carry output C_{out} of the proposed BCD adder has been shifted from the 4-bit binary adders to the correction circuit. Where the correction circuit is a modified version to the original detection circuit which added carry input C_{in} calculation and generates the carry output C_{out}. The Boolean function of the correction circuit is $C_{out} = C_3 + S_3(C_{in}S_0 + S_1 + S_2)$ for checking the condition. The 4-bit adder in the last stage has also been modified to exclude any calculation involving carry output C_{out} as it has already been generated by the correction circuit. The operation of the modified 4-bit adder is summarized in the pseudo code below.

```
If ($C_{out}$==1)
{
        If ($C_{in}$==1)
                {Sum binary 0111} else {Sum binary 0110}
}
Else
{
        If ($C_{in}$ ==1)
                {Sum binary 0001} else {Sum binary 0000}
}
```

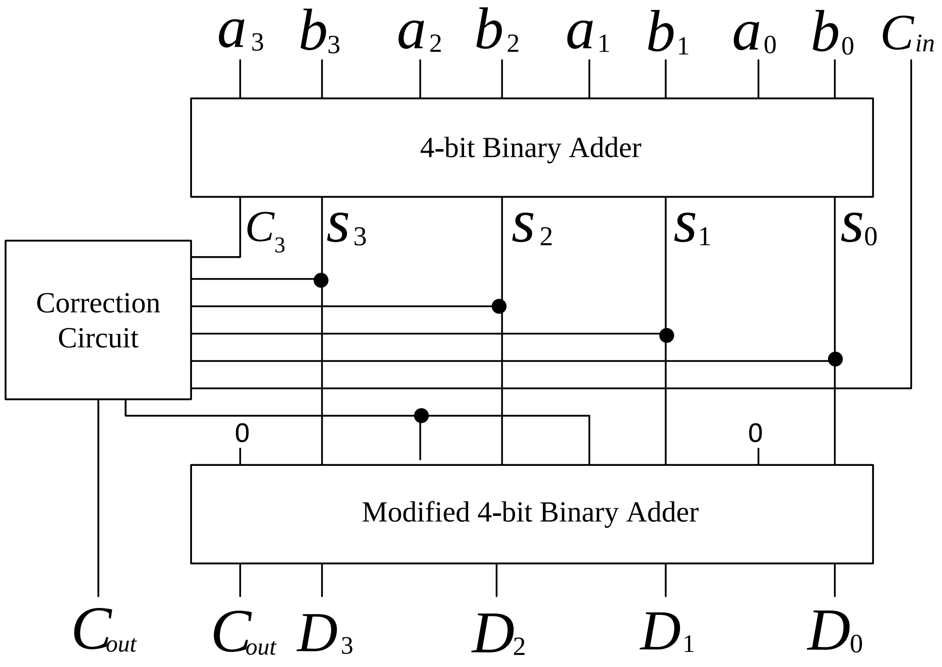

a_3 b_3 a_2 b_2 a_1 b_1 a_0 b_0 C_{in}
4-bit Binary Adder
C_3 S_3 S_2 S_1 S_0
Correction Circuit
0
0
Modified 4-bit Binary Adder
C_{out} C_{out} D_3 D_2 D_1 D_0

Figure 6.2. Proposed Reversible BCD Adder with Carry Input

The *n* digit BCD adder using this design is constructed with cascading of multiple 1 digit BCD adders together with the carry output C_{out} of lower BCD adder digit connecting to the carry input C_{in} of higher digit as illustrated in Figure 6.3. All the 4-

bit binary adders are connected in parallel to each other and hence can be executed simultaneously. So the propagation delay required for an n digit 4-bit binary adder is the same as one adder. As for the correction circuit, it cannot be executed simultaneously because their operation requires the carry output C_{out} from the previous digit to be generated. So the propagation delay required for the n digit is equal to the propagation delay sum of each individual correction circuit. As for the modified 4-bit binary adder, the propagation delay for the n digit is only depending on the modified 4-bit binary adder at the highest significant digit. This is because all the other modified 4-bit binary adders from lower digit have already been executed before reaching to the highest significant digit.

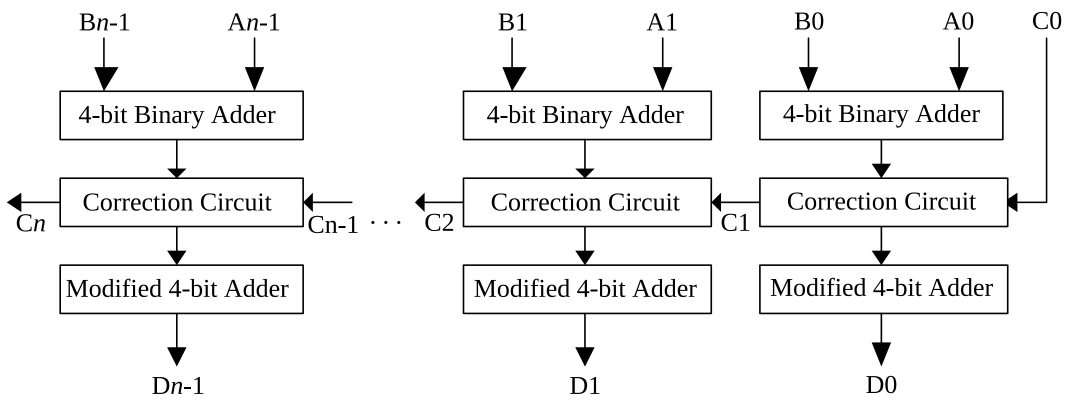


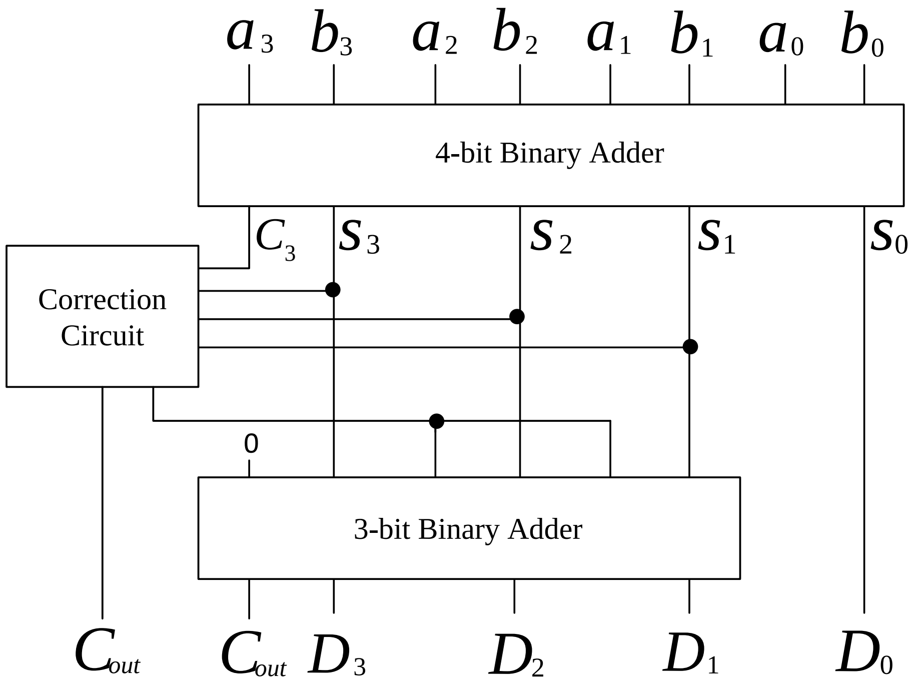
Figure 6.3. Proposed Reversible n Digit BCD Adder with Carry Input

6.2.3 Proposed BCD Adder Design with No Carry Input Required

In most decimal sum, the carry output C_{out} at any given stage would only be available after the carry output C_{out} of previous stage has been generated. At the least significant digit, the decimal sum does not have a carry input C_{in}. So, instead of using a normal adder circuit at the least significant digit, adder circuit with no carry input C_{in} is much preferred. To archive that, the reversible BCD adder design shown in Figure 6.2 has been redesigned to remove all calculation involving the carry input C_{in} and method to generate the required carry output C_{out} faster has been applied.

The proposed reversible BCD adder with no carry input C_{in} is shown in Figure 6.4. The Boolean function used for checking the condition of the correction circuit is $C_{out} = C_3 \oplus S_3(S_1 + S_2)$. Since the decimal result D_0 has already been generated by the top 4-bit binary adder, thus the 4-bit binary adder at the last stage can be replaced with a 3-bit binary adder. The advantage of the BCD adder design is that it has the ability to generate the required carry output C_{out} for the next BCD adder digit before even completing the BCD sum. This reduces the unnecessary delay caused to compute the whole BCD number. The operation of the modified 4-bit adder is summarized in the pseudo code below.

```
If ($C_{out}$==1)
{Sum binary 0111} else {Sum binary 0110}
```



a_3 b_3 a_2 b_2 a_1 b_1 a_0 b_0
4-bit Binary Adder
C_3 S_3 S_2 S_1 S_0
Correction Circuit
0
3-bit Binary Adder
C_{out} C_{out} D_3 D_2 D_1 D_0

Figure 6.4. Proposed Reversible BCD Adder with No Carry Input

In a n digit BCD adder system, BCD adder with no carry input C_{in} can only be used at the least significant bit as illustrated in Figure 6.5. Since its operation does not involve with the carry input C_{in} so it can produce a better solution as compared to the BCD adder with carry input C_{in}.

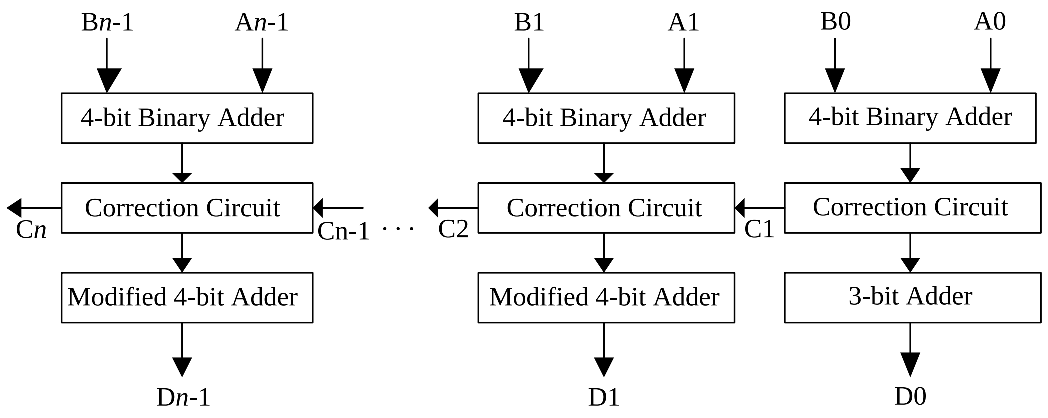


Figure 6.5. Proposed Reversible *n* Digit BCD Adder with No Carry Input

6.3 Design Methodology of Reversible BCD Adder without using Mixed-Polarity Toffoli Gate

In this section, two reversible BCD adder designs are presented, one with carry input C_{in} and another without carry input C_{in}.

6.3.1 Design 1 of Reversible BCD Adder with Carry Input

In order to get the design as shown in Figure 6.2, reversible circuit for the subcomponents must be designed. The complete circuit is constructed using the subcomponents. Details of the design are explained in the subsections below.

6.3.1.1 Proposed Reversible 4-Bit Binary Adder

The reversible gates used for constructing the reversible 4-bit binary adder are the Peres gate and the proposed reversible full adder circuit (discussed in Chapter 5 of this dissertation) is as shown in Figure 6.6. With the control line of the Peres gate and the proposed reversible full adder circuit feed to a constant 0 input, these gates act as a half adder and a full adder. By using these gates, the reversible 4-bit adder has a quantum cost of 22, uses 4 ancilla input and has 0 garbage output.

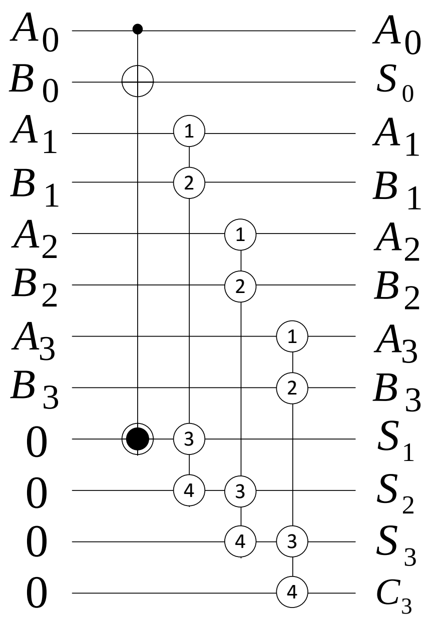


Figure 6.6. Proposed Reversible 4-Bit Binary Adder

6.3.1.2 Proposed Reversible Correction Circuit

The reversible gates used for constructing the reversible correction circuit are the Peres gate, TR gate and NOT gate and CNOT gate as shown in Figure 6.7. In the design, the first Peres gate generates output S_0S_3 and passes it to the second Peres gate. The second Peres gate takes the generated output together with the carry input C_{in} and a constant ‘1’ to generate output $\overline{C_{in}S_0S_3}$. On the other side, a NOT gate connecting to a TR gate generates the output $\overline{S_1}\overline{S_2}$ and passes it to the next TR gate. The TR gate takes the generated output together with S_3 and a constant ‘1’ to generate output $\overline{S_3} + \overline{S_1}\overline{S_2}$. Then output $\overline{C_{in}S_0S_3}$ and $\overline{S_3} + \overline{S_1}\overline{S_2}$ are passed to a Peres gate to obtain output $S_3(C_{in}S_0 + S_1 + S_2)$. Next, a CNOT gate with input C_3 is added to obtain $C_3 \oplus S_3(C_{in}S_0 + S_1 + S_2)$. Upon careful observation, it can be reduced to $C_3 + S_3(C_{in}S_0 + S_1 + S_2)$ because there is no condition where C_3 and $S_3(C_{in}S_0 + S_1 + S_2)$ are true at the same time. Hence, the carry output C_{out} is generated. Lastly, a CNOT gate is used to duplicate the carry output C_{out} in which one of it will be passed to the next BCD adder digit while the other will be used in the modified 4-bit binary adder. The binary output S_0, S_1, S_2, S_3 are also regenerated to be used further in the modified 4-bit adder. By using these gates, the reversible correction circuit has a quantum cost of 26, uses 6 ancilla input and has 5 garbage output.

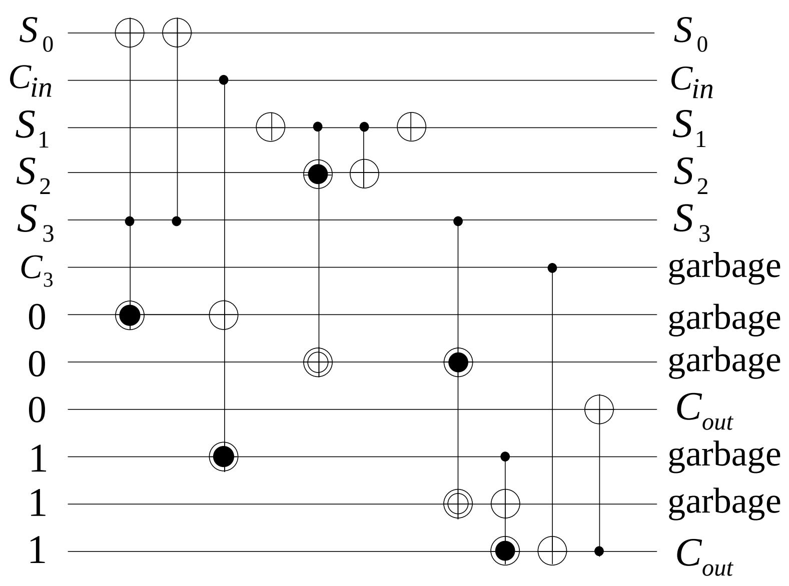


Figure 6.7. Proposed Reversible Correction Circuit

6.3.1.3 Proposed Reversible Modified 4-Bit Adder

The reversible gates used for constructing the reversible modified 4-bit binary adder are the Peres gate, CNOT gate and the proposed reversible full adder circuit. The proposed reversible modified 4-bit binary adder circuit is shown in Figure 6.8. The circuit used is similar to the one discussed in Section 6.3.1.1. Only the full adder used for generating output D_3 and C_{out} is replaced by a CNOT gate to generate only D_3. This carry output C_{out} is omitted because it has already been generated by the correction circuit. By using these gates, the reversible modified 4-bit adder has a quantum cost of 17, uses 3 ancilla inputs and has 4 garbage output.

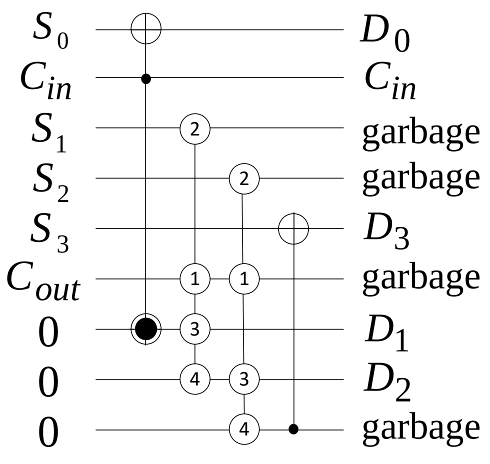


Figure 6.8. Proposed Reversible Modified 4-Bit Binary Adder

6.3.1.4 Overall Circuit

Integrate all the proposed reversible subcomponents together into Figure 6.2 to design the 1 digit reversible BCD adder with carry input C_{in} is as shown in Figure 6.9. For easier reference, it will be named as C-BCD1 further in this work. The overall design has a total quantum cost of 65, uses 13 ancilla input and has 9 garbage output.

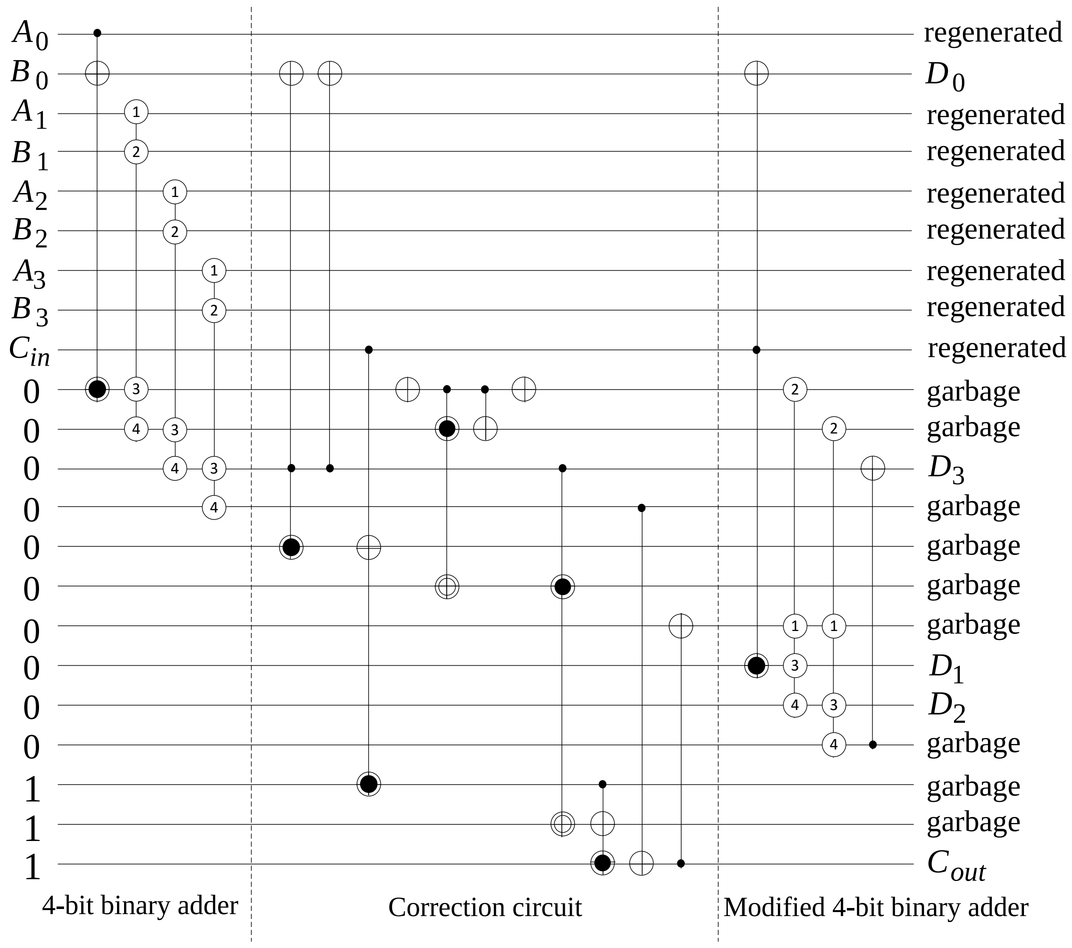


Figure 6.9. Proposed Reversible BCD Adder with Carry Input (C-BCD1)

6.3.1.5 Optimization In Terms of Propagation Delay

To keep propagation delay minimal, reversible gates constructing within the final circuit as shown in Figure 6.9 have been reallocated and the logic depth between the carry input C_{in} and carry output C_{out} has been reduced by following the moving rules described in [99]. Reversible gates that are parallel to one another are cascaded together to allow parallel execution and achieves reduction in the overall running time. Figure 6.10 shows the updated design for C-BCD1 after optimized in terms of propagation delay. The cascaded reversible gates are grouped in columns for easier propagation delay calculation. The propagation delay calculation for C-BCD1 is illustrated in the following steps:

1. Step 1 has 1 Peres gate. The propagation delay at this step is $\Delta 4$.
2. Step 2 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
3. Step 3 has 1 reversible full adder circuit and 1 NOT gate. The propagation delay at this step is $\Delta 4$ because the reversible full adder circuit is working parallel to the NOT gate.
4. Step 4 has 1 reversible full adder circuit and 1 TR gate. The propagation delay at this step is $\Delta 4$ because both of the gates are working parallel.
5. Step 5 has 1 CNOT gate and 1 Peres gate. The propagation delay at this step is $\Delta 4$ because both of the gates are working parallel.
6. Step 6 has 1 TR gate, 1 NOT gate and 1 Peres gate. The propagation delay at this step is $\Delta 4n$ as the gates are working parallel. The n here refers to the to the BCD digit. At this step, the carry input C_{in} is introduced. When the BCD adder is connected as a n digit BCD adder, all adders are run simultaneously together until it reaches the stage where the adder requires the C_{out} from the previous digit.
7. Step 7 has 2 CNOT gates. The propagation delay at this step is $\Delta 1n$ because both gates are working in parallel.
8. Step 8 has 2 Peres gates. The propagation delay at this step is $\Delta 4n$ because both gates are working in parallel.
9. Step 9 has 1 CNOT gate. The propagation delay at this step is $\Delta 1n$. The C_{out} is generated at this step.
10. Step 10 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
11. Step 11 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
12. Step 12 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$.

The total propagation delay produced is $\Delta(10n + 29)$, where n refers to the BCD digit.

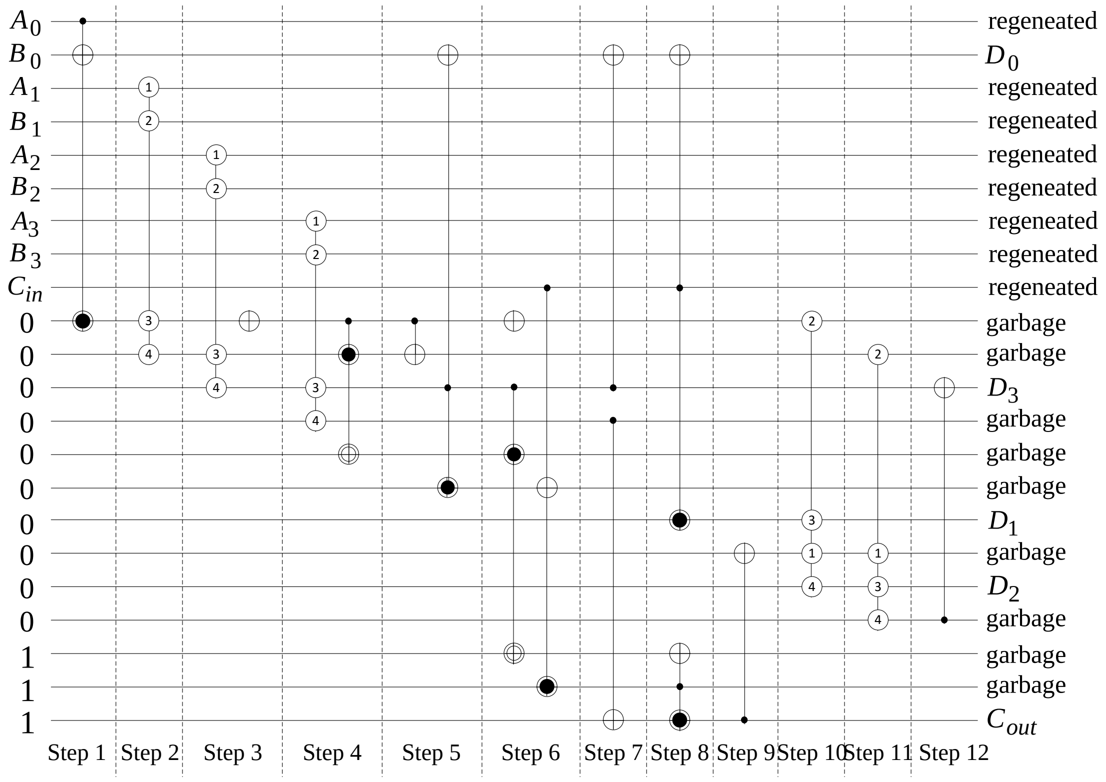


Figure 6.10. Propagation Delay Optimized for C-BCD1

6.3.2 Design 1: Reversible BCD Adder with No Carry Input Required

In order to get the design as shown in Figure 6.4, reversible circuit for the subcomponents must be designed. The complete circuit is constructed using the subcomponents. The reversible 4-bit binary adder used in this design is the same as the one proposed in Section 6.3.1.1. Details of the design are explained in the subsections below.

6.3.2.1 Proposed Reversible Correction Circuit

The reversible gates used for constructing the reversible correction circuit are the TR gate, NOT gate and CNOT gate as seen in Figure 6.11. In the design, with the help of the first NOT gate, the first TR gate generates output $\overline{S_1}\overline{S_2}$ and passes it to the second

TR gate. The second TR gate takes the generated output together with S_3 and a constant '0' to generate output $S_3(S_1 + S_2)$. Next, a CNOT gate with input C_3 is added to obtain $C_3 \oplus S_3(S_1 + S_2)$ which is the carry output C_{out} of the BCD adder. Lastly a CNOT gate is used to duplicate the carry output C_{out} in which one of it will be passed to the next BCD adder digit while the other will be used in the modified 4-bit binary adder. The binary output S_0, S_1, S_2, S_3 are also regenerated to be used further in the modified 4-bit adder. By using these gates, the reversible correction circuit has a quantum cost of 13, uses 3 ancilla input and has 2 garbage output. Compared to the proposed reversible correction circuit with carry input C_{in}, it has less quantum cost and uses less ancilla input.

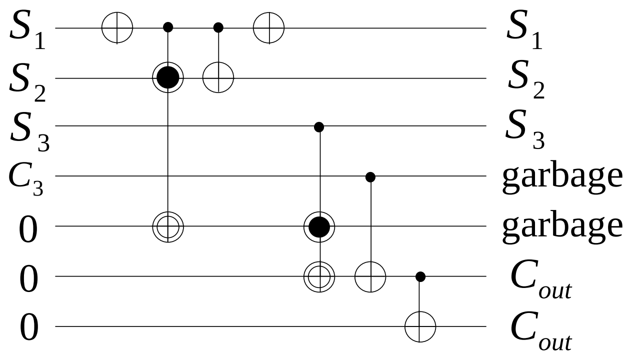


Figure 6.11. Proposed Reversible Correction Circuit

6.3.2.2 Proposed Reversible 3-Bit Binary Adder

The reversible gates used for constructing the reversible 3-bit binary adder are the Peres gate, CNOT gate and the proposed reversible full adder circuit as shown in Figure 6.12. The circuit is similar to the modified 4-bit binary adder proposed in Section 6.3.1.3 except it has one full adder short. By using these gates, the reversible 3-bit binary adder has a quantum cost of 11 and uses 2 ancilla inputs.

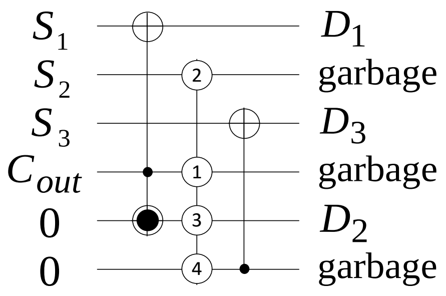


Figure 6.12. Proposed Reversible 3-Bit Binary Adder

6.3.2.3 Overall Circuit

Integrate all the proposed reversible modules together into Figure 6.4 to design the 1-digit reversible BCD adder with no carry input C_{in} is as shown in Figure 6.13. For easier reference, it will be named as NC-BCD1 further in this work. The overall design has a total quantum cost of 46, uses 9 ancilla inputs and has 5 garbage outputs.

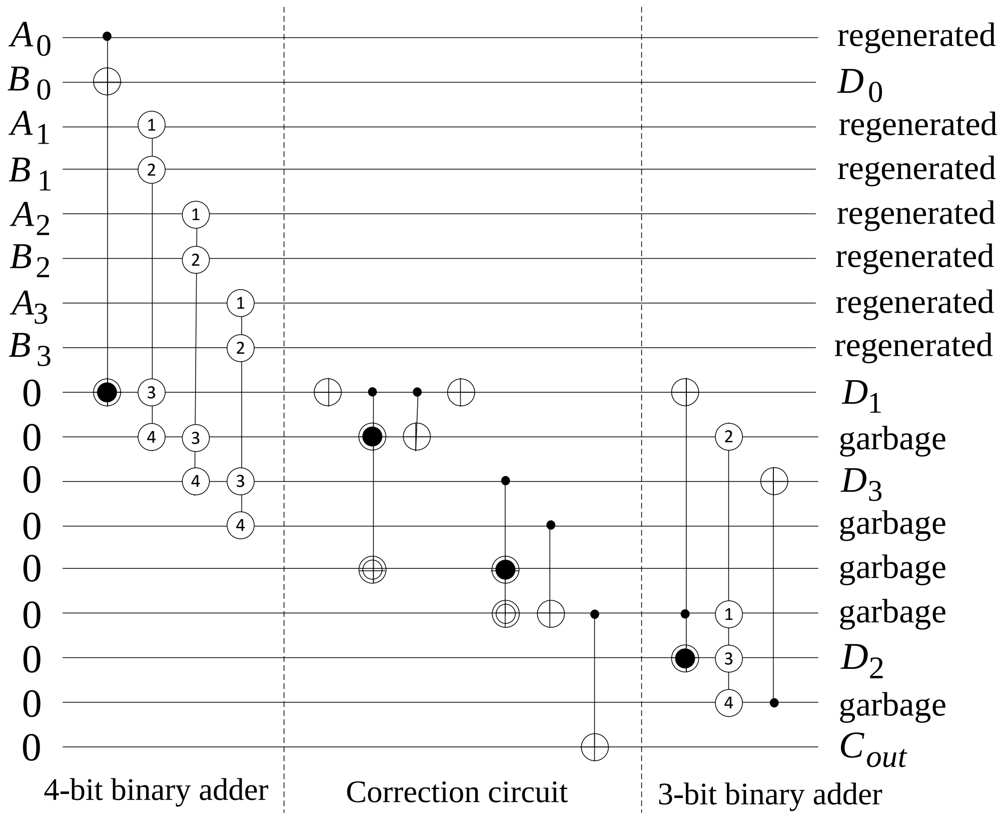


Figure 6.13. Proposed Reversible BCD Adder with No Carry Input Required (NC-BCD1)

6.3.2.4 Optimization in Terms of Propagation Delay

From the overall circuit shown in Figure 6.13, the optimized circuit in terms of propagation delay is as illustrated in Figure 6.14. The figure is divided into columns for easier propagation delay calculation. The propagation delay calculation for C-BCD2 is illustrated in the following steps:

1. Step 1 has 1 Peres gate. The propagation delay at this step is $\Delta 4$.
2. Step 2 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
3. Step 3 has 1 reversible full adder circuit and 1 NOT gate. The propagation delay at this step is $\Delta 4$ because the reversible full adder circuit is working parallel to the NOT gate.
4. Step 4 has 1 reversible full adder circuit and 1 TR gate. The propagation delay at this step is $\Delta 4$ because both gates are working parallel.
5. Step 5 has 1 CNOT gate and 1 TR gate. The propagation delay at this step is $\Delta 4$ because both of the gates are working parallel.
6. Step 6 has 1 NOT gate and 1 CNOT gate. The propagation delay at this step is $\Delta 1$ because both of the gates are working parallel.
7. Step 7 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$. The C_{out} is generated at this step.
8. Step 8 has 1 Peres gate. The propagation delay at this step is $\Delta 4$.
9. Step 9 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
10. Step 10 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$.

Thus the total propagation delay produced is $\Delta(22n + 9)$, where n refers to the BCD digit. Since there can only be one BCD adder with no carry input C_{in} required for a n digit BCD adder, so the n here will always be 1.

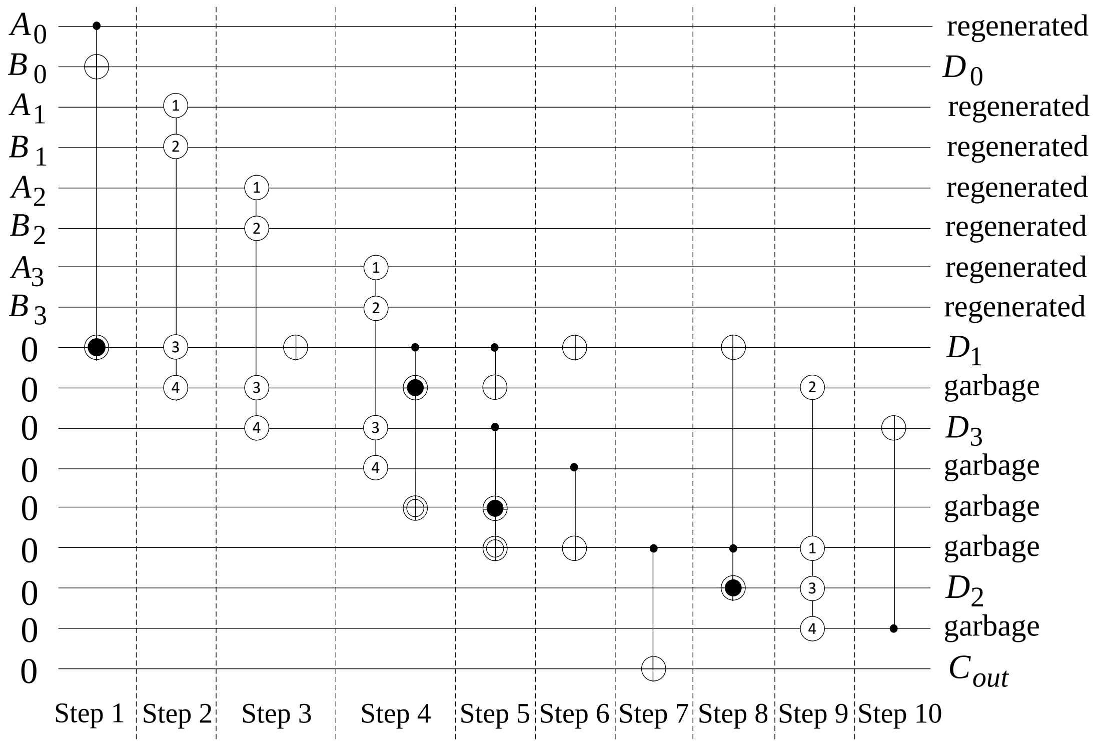


Figure 6.14. Propagation Delay Optimized for NC-BCD1

6.4 Design Methodology of Reversible BCD Adder with Mixed-Polarity Toffoli Gate

In this section, two reversible BCD adder designs with the use of mixed-polarity Toffoli gate are presented, one with carry input C_{in} and another without carry input C_{in}.

6.4.1 Design 2 of Reversible BCD Adder with Carry Input

In order to get the design seen in Figure 6.2 under Section 6.2.2, reversible circuit for the subcomponents must be designed. The complete circuit is constructed using the subcomponents. The reversible 4-bit binary adder and the reversible modified 4-bit adder design are the same as the previous one, so only the reversible correction circuit needs to be designed.

6.4.1.1 Proposed Reversible Correction Circuit

The reversible gates used for constructing the reversible correction circuit are the Peres gate, TR gate, mixed-polarity Toffoli gate and CNOT gate as seen in Figure 6.15. In the design, the first Peres gate generates output S_0S_3 and passes it to the second Peres gate. The second Peres gate takes the generated output together with the carry input C_{in} and a constant '1' to generate output $\overline{C_{in}S_0S_3}$. On the other side, a mixed-polarity Toffoli gate takes the input S_1, S_2 and generates the output $\overline{S_1}\,\overline{S_2}$ and passes it to a TR gate. The TR gate takes the generated output together with S_3 and a constant '1' to generate output $\overline{S_3} + \overline{S_1}\,\overline{S_2}$. Then output $\overline{C_{in}S_0S_3}$ and $\overline{S_3} + \overline{S_1}\,\overline{S_2}$ are passed to a Peres gate to obtain output $S_3(C_{in}S_0 + S_1 + S_2)$. Next, a CNOT gate with input C_3 is added to obtain $C_3 \oplus S_3(C_{in}S_0 + S_1 + S_2)$ which is the carry output C_{out} of the BCD adder. Lastly a CNOT gate is added to duplicate the carry output C_{out} in which one of it will be passed to the next BCD adder digit while the other will be used in the modified 4-bit binary adder. The binary output S_0, S_1, S_2, S_3 are also regenerated to be used further in the modified 4-bit adder. By using these gates, the reversible correction circuit has a quantum cost of 27, uses 7 ancilla inputs and has 7 garbage outputs.

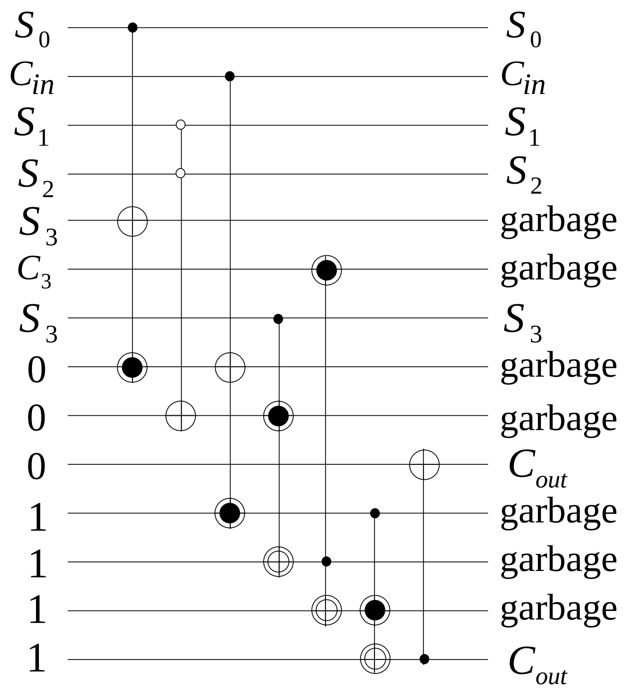


Figure 6.15. Proposed Reversible Correction Circuit

6.4.1.2 Overall Circuit

Integrate all the proposed reversible subcomponents together into Figure 6.2 to design the 1-digit reversible BCD adder with no carry input C_{in} is shown in Figure 6.16. For easier reference, it will be named as C-BCD2 further in this work. From the overall circuit, an extra CNOT gate is added to duplicate the signal S_3 because the reversible correction circuit requires two S_3. The overall design has a total quantum cost of 67, uses 15 ancilla inputs and has 11 garbage outputs.

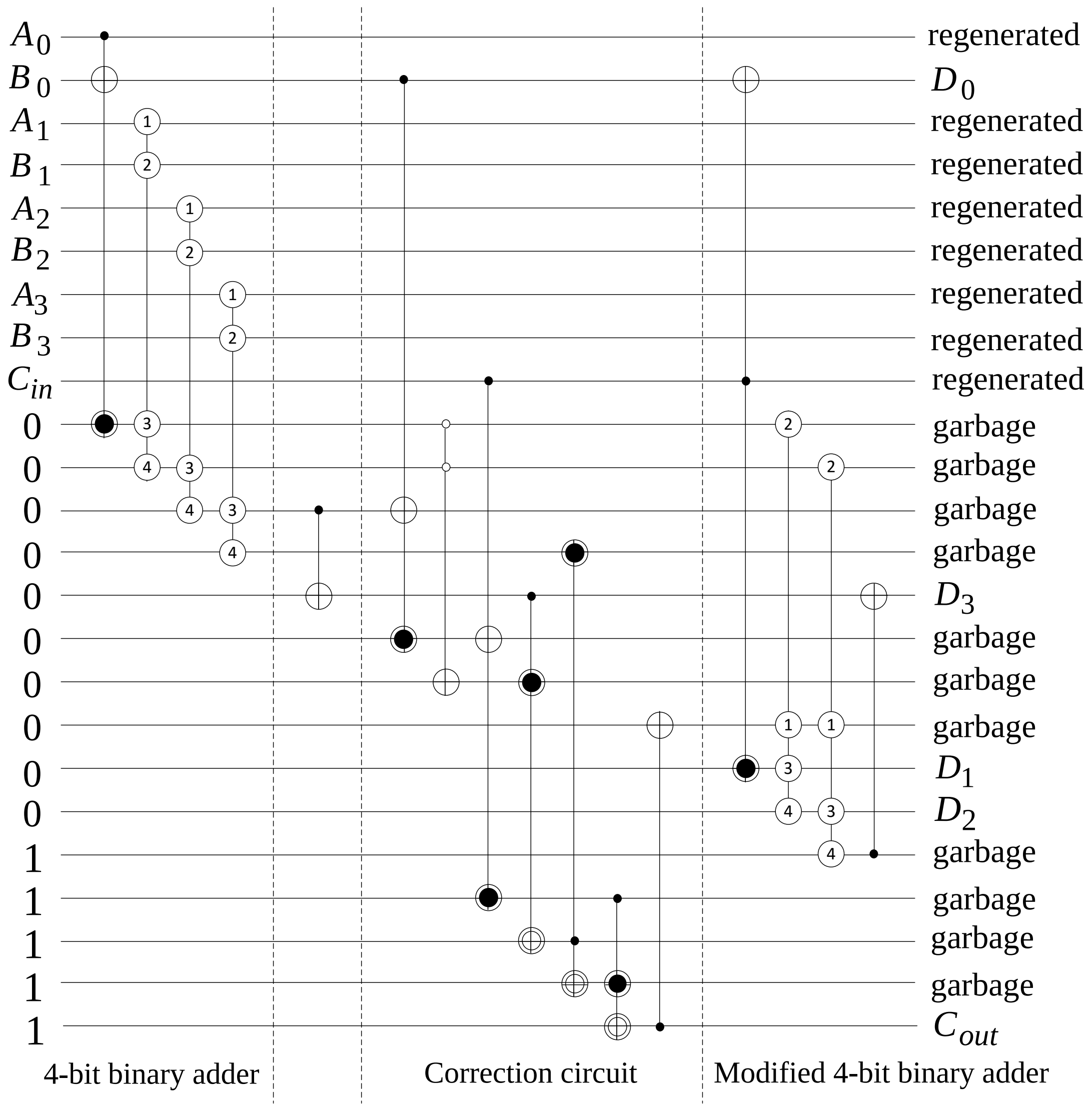


Figure 6.16. Proposed Reversible BCD Adder with Carry Input (C-BCD2)

6.4.1.3 Optimization In Terms of Propagation Delay

From the overall circuit shown in Figure 6.16, the optimized circuit in terms of propagation delay is illustrated in Figure 6.17. The figure is divided into columns for easier propagation delay calculation. The propagation delay calculation for C-BCD2 is illustrated in the following steps:

1. Step 1 has 1 Peres gate. The propagation delay at this step is $\Delta 4$.
2. Step 2 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
3. Step 3 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
4. Step 4 has 1 reversible full adder circuit, 1 mixed-polarity Toffoli gate and 1 CNOT gate. The propagation delay at this step is $\Delta 6$ because the mixed-polarity Toffoli gate is working parallel to the reversible full adder circuit and CNOT gate.
5. Step 5 has 1 TR gate and 1 Peres gate. The propagation delay at this step is $\Delta 4$ because both of the gates are working parallel.
6. Step 6 has 1 TR gate and 1 Peres gate. The propagation delay at this step is $\Delta 4n$. Where n refers to the BCD digit. At this step, carry input C_{in} is introduced. When the BCD adder is connected as a n digit BCD adder, all adders are run simultaneously together until it reaches the stage where the adder requires the C_{out} from the previous digit.
7. Step 7 has 1 Peres gate and 1 TR gate. The propagation delay at this step is $\Delta 4n$ because both gates are working in parallel.
8. Step 8 has 1 CNOT gate. The propagation delay at this step is $\Delta 1n$. The C_{out} is generated at this step.
9. Step 9 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
10. Step 10 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
11. Step 11 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$.

Thus the total propagation delay produced is $\Delta(9n + 31)$, where n refers to BCD digit.

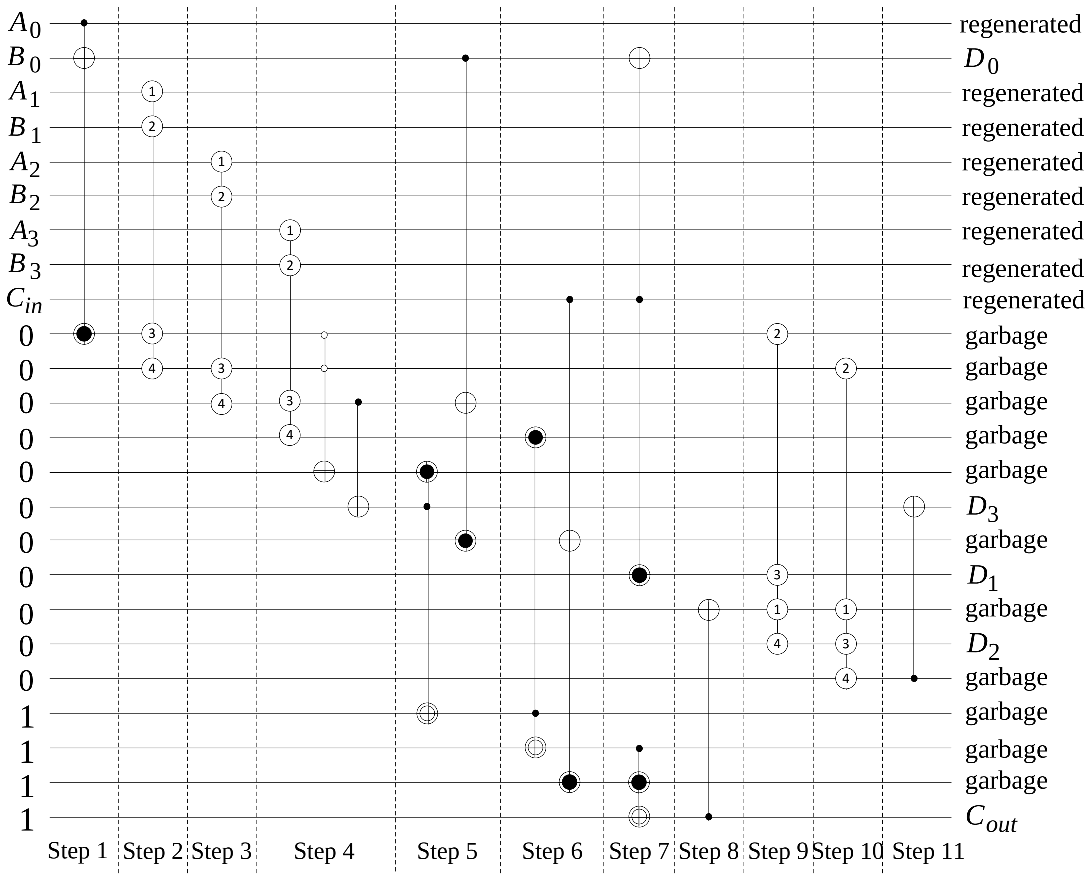


Figure 6.17. Propagation Delay Optimized for C-BCD2

6.4.2 Design 2 of Reversible BCD Adder with No Carry Input Required

In order to get the design seen in Figure 6.4 under Section 6.2.3, reversible circuit for the subcomponents must be designed. The complete circuit is constructed using the subcomponents. The reversible 4-bit binary adder and the reversible 3-bit adder design are the same as the previous one, so only the reversible correction circuit needs to be designed.

6.4.2.1 Proposed Reversible Correction Circuit

The reversible gates used for constructing the reversible correction circuit are the mixed-polarity Toffoli gate, TR gate and CNOT gate as seen in Figure 6.18. In the design, the mixed-polarity Toffoli gate takes S_1, S_2 as input and generates the output

$\overline{S_1}\overline{S_2}$. The TR gate takes the generated output together with S_3 and a constant '0' to generate output $S_3(S_1 + S_2)$. Next, a CNOT gate with input C_3 is added to obtain $C_3 \oplus S_3(S_1 + S_2)$ which is the carry output C_{out} of the BCD adder. Lastly a CNOT gate is used to duplicate the carry output C_{out} in which one of it will be passed to the next BCD adder digit while the other will be used in the modified 4-bit binary adder. By using these gates, the reversible correction circuit has a quantum cost of 12, uses 3 ancilla input and has 2 garbage output. Compared to the proposed reversible correction circuit with carry input C_{in}, it has less quantum cost and it uses less ancilla inputs.

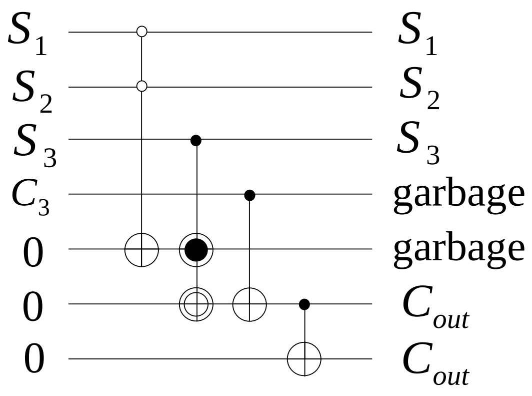


Figure 6.18. Proposed Reversible Correction Circuit

6.4.2.2 Overall Circuit

Integrate all the proposed reversible modules together into Figure 6.4 to design the 1-digit reversible BCD adder with no carry input C_{in} is shown in Figure 6.19. For easier reference, it will be named as NC-BCD1 further in this work. The overall design has a total quantum cost of 45, uses 9 ancilla inputs and has 5 garbage outputs.

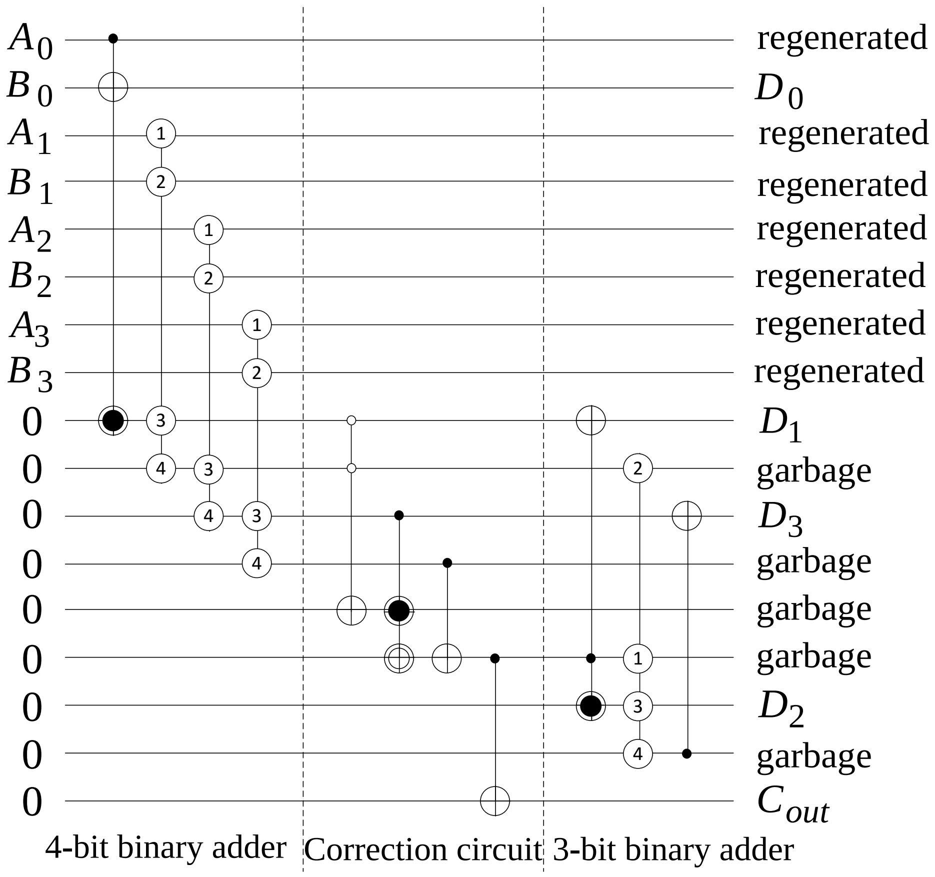


Figure 6.19. Proposed Reversible BCD Adder with No Carry Input Required (NC-BCD2)

6.4.2.3 Optimization In Terms of Propagation Delay

From the overall circuit as shown in Figure 6.19, the optimized circuit in terms of propagation delay is illustrated in Figure 6.20. The figure is divided into columns for easier propagation delay calculation. The propagation delay calculation for C-BCD2 is illustrated in the following steps:

1. Step 1 has 1 Peres gate. The propagation delay at this step is $\Delta 4$.
2. Step 2 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
3. Step 3 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
4. Step 4 has 1 reversible full adder circuit and 1 mixed-polarity Toffoli gate. The propagation delay at this step is $\Delta 6$ because the reversible full adder circuit is working parallel to the mixed-polarity Toffoli gate.
5. Step 5 has 1 TR gate. The propagation delay at this step is $\Delta 4$.

6. Step 6 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$.
7. Step 7 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$. The C_{out} is generated at this step.
8. Step 8 has 1 Peres gate. The propagation delay at this step is $\Delta 4$.
9. Step 9 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
10. Step 10 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$.

Thus the total propagation delay produced is $\Delta(24n + 9)$, where n refers to the BCD digit. Since there can only be one BCD adder with no carry input C_{in} required for a n digit BCD adder, so the n here will always be 1.

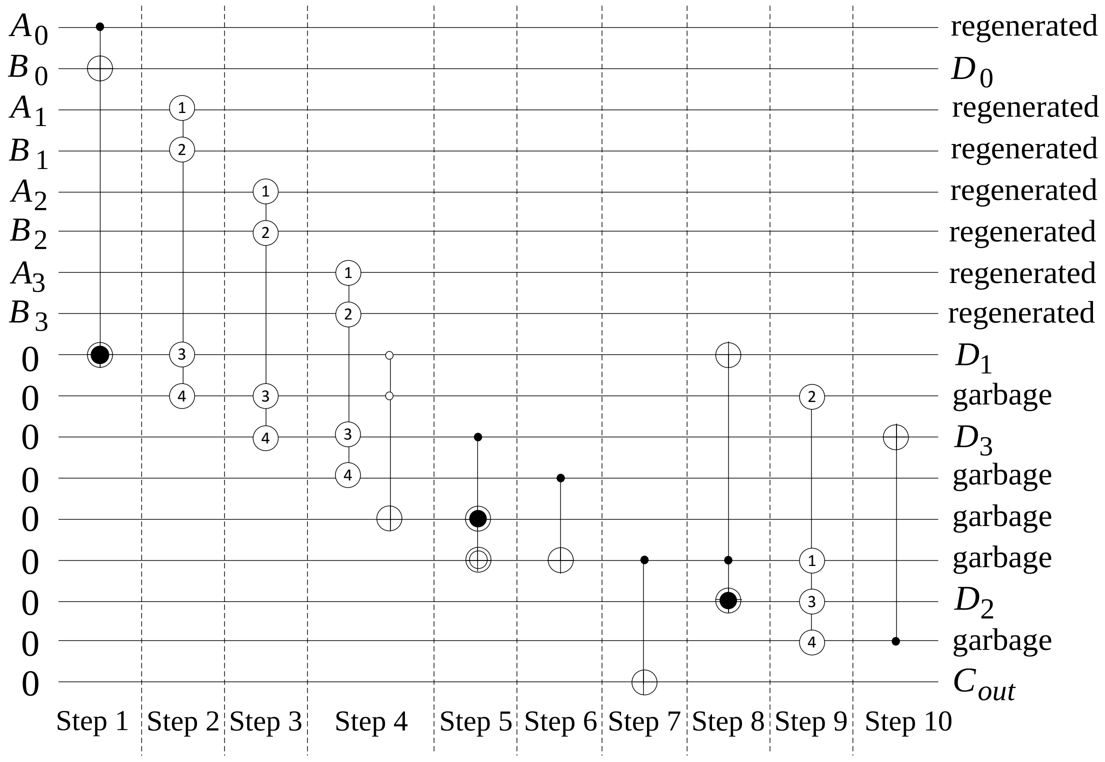


Figure 6.20. Propagation Delay Optimized for NC-BCD2

6.5 Evaluation of the Proposed Designs

The overall design for n digit reversible BCD adder with carry input C_{in} using C-BCD1 has a total ancilla input of $13n$ and $9n$ garbage outputs. It has a total quantum cost of $65n$ and propagation delay of $\Delta(10n + 29)$. Using C-BCD2, it has a total ancilla input of $15n$ and $11n$ garbage outputs. It has a total quantum cost of $67n$ and propagation delay of $\Delta(9n + 31)$. For n digit reversible BCD adder with no carry input C_{in} required, using C-BCD1 and NC-BCD1, it has a total ancilla input of 13n-4 and 9n-4 garbage outputs. It has a total quantum cost of 65n-19 and propagation delay of $\Delta(10n + 21)$. Using C-BCD2 and NC-BCD2, it has a total ancilla input of 15n-6 and 11n-6 garbage outputs. It has a total quantum cost of 67n-22 and propagation delay of $\Delta(9n + 24)$.

A comparison of the proposed designs with the existing designs is illustrated in Table 6-1. From the table, it can be observed that all the proposed designs are having less propagation delay as compared to all the existing ones. By comparing the two proposed designs with carry input C_{in}, design 2 is having less propagation delay; however the drawback is that it uses more ancilla inputs, garbage outputs and higher quantum cost. Among the existing designs, the designs presented by Biswas et al. [103], James et al. [106] Approach 1 and James et al. [106] Approach 2 have the least propagation delay. These designs are further taken into comparison with the proposed designs. Table 6-3, Table 6-4, Table 6-5 show the comparison for values of n ranging from $n = 4$ digits to $n = 1024$ digits. For the proposed design 1, it has an improvement ranging from 56.05% to 74.29% , 57.93% to 71.37% and 34.29% to 9.32% as compared to [103], [106] Approach 1 and [106] Approach 2. For the proposed design 2, it has an improvement ranging from 57.32% to 76.85% , 59.15% to 74.22% and 36.19% to 18.35% compared to [103], [106] Approach 1 and [106] Approach 2. For the proposed design 3, it has an improvement ranging from 61.15% to 74.31% , 62.8% to 71.39% and 41.9% to 9.4% compared to [103], [106] Approach 1 and [106] Approach 2. For the proposed design 4, it has an improvement ranging from 61.78% to 76.86% , 63.41% to 74.24% and 42.86% to 18.41% as compared to [103], [106] Approach 1 and [106] Approach 2.

From the comparison table with Biswas et al. [103] and James et al. [106] Approach 1, it can be seen that there exists a large propagation delay improvement. As the BCD digit increases, the improvement range increases significantly. As for the comparison with [106] Approach 2, the improvement range gradually drops as the digit increases. This is because [106] Approach 2 has a much higher propagation delay at its least significant digit. Taking it to compare with the proposed design 1, the propagation delay contributed respectively are $\Delta 72$ and $\Delta 39$. With each increase in the BCD digit, the propagation delay contributed increases by $\Delta 11$ for [106] Approach 2 and $\Delta 10$ for proposed design 1. Proposed design 1 has a $\Delta 33$ advantage at the least significant digit, however as the digit increases it only has a $\Delta 1$ benefit at each increase in the digit. So the improvement range drops as the improvement is not significant. Comparing with the other parameters, the proposed design 1 has a much lower quantum cost, however uses slightly more ancilla input and garbage output.

Overall, the proposed designs of reversible BCD adders are efficient in terms of propagation delay while maintaining the other parameters at an acceptable range compared to the existing designs in the literature.

Table 6-1. Comparison of *n* Digit Reversible BCD Adders

	Ancilla Inputs	Garbage Outputs	Quantum Cost	Delay Δ
Babu et al. [104]	$17n$	$18n$	$143n$	$68n+11$
James et al. [96]	$35n$	$35n$	$190n$	$36n+55$
Biswas et al. [103]	$7n$	$7n$	$50n$	$39n+1$
Mohamadi et al. [98]	$16n$	$16n$	$70n$	$52n$
Thomsen et al. [105] (with carry input)	$4n$	$3n$	$200n$	$65n+115$
Thomsen et al. [105] (without carry input)	$4n$	$3n+1$	$200n-113$	$\begin{cases} 67 & n=1 \\ 65n+117 & n>1 \end{cases}$
James et al. [106] (Approach 1)	$8n$	$8n$	$80n$	$35n+24$
James et al. [106] (Approach 2)	$12n$	$12n$	$102n$	$11n+61$
Thapliyal et al. [64] (Approach 1 with carry input)	$2n$	n	$88n$	$66n+1$
Thapliyal et al. [64] (Approach 1 without carry input)	$2n$	n	$88n-12$	$66n+3$
Thapliyal et al. [64] (Approach 2 with carry input)	n	0	$70n$	$52n+1$
Thapliyal et al. [64] (Approach 2 without carry input)	n	0	$70n-12$	$52n+2$
Proposed design 1 (with carry input)	$13n$	$9n$	$65n$	$10n+29$
Proposed design 2 (with carry input)	$15n$	$11n$	$67n$	$9n+31$
Proposed design 3 (without carry input)	$13n-4$	$9n-4$	$65n-19$	$\Delta(10n+21)$
Proposed design 4 (without carry input)	$15n-6$	$11n-6$	$67n-22$	$\Delta(9n+24)$

Proposed design 1 is designed using only C-BCD1
Proposed design 2 is designed using only C-BCD2
Proposed design 3 is designed using only C-BCD1 and NC-BCD1
Proposed design 4 is designed using only C-BCD2 and NC-BCD2

Table 6-2. Propagation Delay Comparison of *n* Digits Reversible BCD Adders using Proposed Design 1

Dig	Ref 1	Ref 2	Ref 3	PD1	Imp wrt Ref 1	Imp wrt Ref 2	Imp wrt Ref 3
4	157	164	105	69	56.05	57.93	34.29
8	313	304	149	109	65.18	64.14	26.85
16	625	584	237	189	69.76	67.64	20.25
32	1249	1144	413	349	72.06	69.49	15.5
64	2497	2264	765	669	73.21	70.45	12.55
128	4993	4504	1469	1309	73.78	70.94	10.89
256	9985	8984	2877	2589	74.07	71.18	10.01
512	19969	17944	5693	5149	74.22	71.31	9.56
1024	39937	35864	11325	10269	74.29	71.37	9.32

Dig: Number of digits
Ref 1: [103]
Ref 2: [106] (Approach 1)
Ref 3: [106] (Approach 2)
PD1: Proposed design 1 using only C-BCD1
Imp: Improvement of proposed work with respect to

Table 6-3. Propagation Delay Comparison of *n* Digits Reversible BCD Adders using Proposed Design 2

Dig	Ref 1	Ref 2	Ref 3	PD2	Imp wrt Ref 1	Imp wrt Ref 2	Imp wrt Ref 3
4	157	164	105	67	57.32	59.15	36.19
8	313	304	149	103	67.09	66.12	30.87
16	625	584	237	175	72	70.03	26.16
32	1249	1144	413	319	74.46	72.12	22.76
64	2497	2264	765	607	75.69	73.19	20.65
128	4993	4504	1469	1183	76.31	73.73	19.47
256	9985	8984	2877	2335	76.61	74.01	18.84
512	19969	17944	5693	4639	76.77	74.15	18.51
1024	39937	35864	11325	9247	76.85	74.22	18.35

Dig: Number of digits
Ref 1: [103]
Ref 2: [106] (Approach 1)
Ref 3: [106] (Approach 2)
PD2: Proposed design 2 using only C-BCD2
Imp: Improvement of proposed work with respect to

Table 6-4. Propagation Delay Comparison of *n* Digits Reversible BCD Adders using Proposed Design 3

Dig	Ref 1	Ref 2	Ref 3	PD3	Imp wrt Ref 1	Imp wrt Ref 2	Imp wrt Ref 3
4	157	164	105	61	61.15	62.8	41.9
8	313	304	149	101	67.73	66.78	32.21
16	625	584	237	181	71.04	69.01	23.63
32	1249	1144	413	341	72.7	70.19	17.43
64	2497	2264	765	661	73.53	70.8	13.59
128	4993	4504	1469	1301	73.94	71.11	11.44
256	9985	8984	2877	2581	74.15	71.27	10.29
512	19969	17944	5693	5141	74.26	71.35	9.7
1024	39937	35864	11325	10261	74.31	71.39	9.4

Dig: Number of digits
Ref 1: [103]
Ref 2: [106] (Approach 1)
Ref 3: [106] (Approach 2)
PD3: Proposed design 3 using only C-BCD1 and NC-BCD1
Imp: Improvement of proposed work with respect to

Table 6-5. Propagation Delay Comparison of n Digits Reversible BCD Adders using Proposed Design 4

Dig	Ref 1	Ref 2	Ref 3	PD4	Imp wrt Ref 1	Imp wrt Ref 2	Imp wrt Ref 3
4	157	164	105	60	61.78	63.41	42.86
8	313	304	149	96	69.33	68.42	35.57
16	625	584	237	168	73.12	71.23	29.11
32	1249	1144	413	312	75.02	72.73	24.46
64	2497	2264	765	600	75.97	73.5	21.57
128	4993	4504	1469	1176	76.45	73.89	19.95
256	9985	8984	2877	2328	76.69	74.09	19.08
512	19969	17944	5693	4632	76.8	74.19	18.64
1024	39937	35864	11325	9240	76.86	74.24	18.41

Dig: Number of digits
Ref 1: [103]
Ref 2: [106] (Approach 1)
Ref 3: [106] (Approach 2)
PD4: Proposed design 4 using only C-BCD2 and NC-BCD2
Imp: Improvement of proposed work with respect to

6.6 Simulation and Verification

To verify all the proposed reversible BCD adders follows exactly the required function, all the proposed reversible BCD adders including all their reversible circuit components and reversible gates are being functionally verified through Very High Speed Integrated Circuit Hardware Description Language (VHDL) simulation. Firstly, all reversible gates used are created into the library through VHDL and are later used throughout the whole design process. Next, each of the proposed reversible BCD adder and its subcomponents circuits are coded individually using VHDL with the reversible gates created at the library earlier. Finally, a test bench waveform is generated to verify the performance of each proposed designs that each component follows the theoretical behavior as discussed in Section 6.2.2 and Section 6.2.3. The simulation flow of this process is illustrated in Figure 6.21 and the test bench waveform verification is shown in Figure 6.22. Figure 6.22 shows a snap shot on how the test bench waveform is conducted. The section $a(0-3)$ and $b(0-3)$ represent the two decimal to be summed while d(0-3) and decimal_cout show the decimal output and carry output C_{out} from the BCD adder. Verification is done by inputting all inputs with logic 0 and slowly increases to its most significant value. The corresponding outputs are being observed and verified to check, if it functionally matches with the theoretical framework. All reversible circuits test bench waveforms are simulated and verified using Xilinx ISE 10.1.

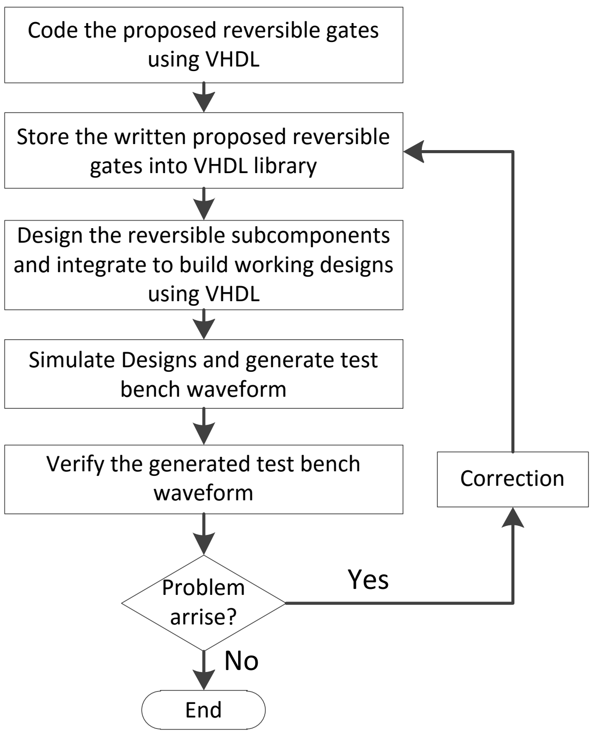

Code the proposed reversible gates using VHDL
Store the written proposed reversible gates into VHDL library
Design the reversible subcomponents and integrate to build working designs using VHDL
Simulate Designs and generate test bench waveform
Verify the generated test bench waveform
Correction
Problem arrise?
Yes
No
End

Figure 6.21. Simulation flow of the proposed reversible BCD adder designs using VHDL

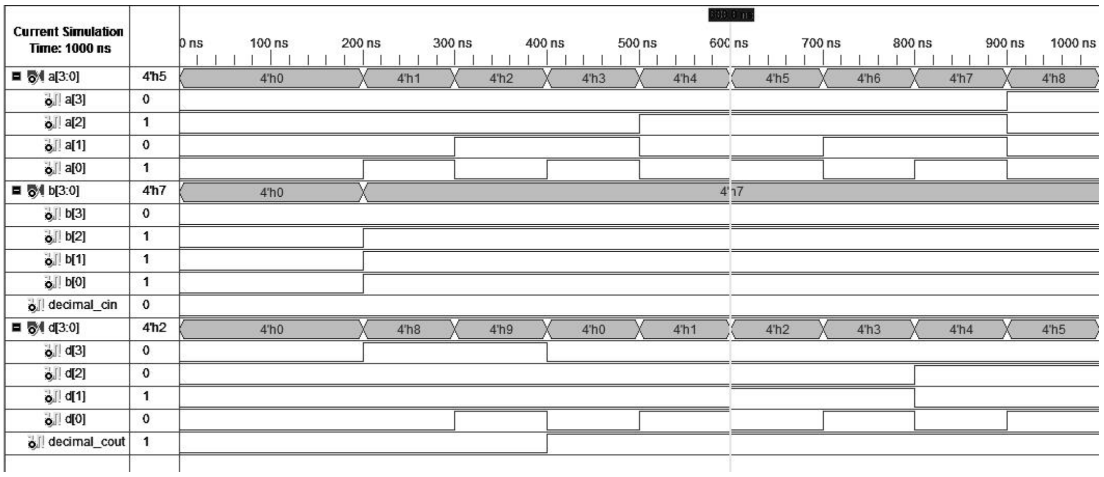


Figure 6.22. Test bench waveform verification

6.7 Chapter Summary

In conclusion, this work presented four reversible BCD adder designs primarily designed for optimizing in terms of propagation delay. All the reversible gates in the proposed adders have been reallocated with cascading of all parallel execute gates together to ensure that optimal propagation delay is achieved. Apart from that, the other parameters such as ancilla input, garbage output and quantum cost are maintained at an acceptable range. The proposed reversible BCD adder designs will be beneficial in the reversible computing applications requiring fast BCD arithmetic unit.

CHAPTER 7 DESIGN OF LOW QUANTUM COST REVERSIBLE BCD ADDER

In this chapter, two reversible BCD adder designs are proposed to optimize in terms of quantum cost. Section 7.1 provides an overview on the chapter and Section 7.2 discuss about the two proposed BCD adder concepts. Section 7.3 and Section 7.4 show the reversible logic implementation of the proposed designs. Optimization of the proposed reversible BCD adders in term of propagation delay is described in Section 7.5. Evaluation of the proposed reversible BCD adders with the existing literature work is discussed in Section 7.6. Simulation and verification of the proposed designs are discussed in Section 7.7 and a conclusion is drawn in Section 7.8.

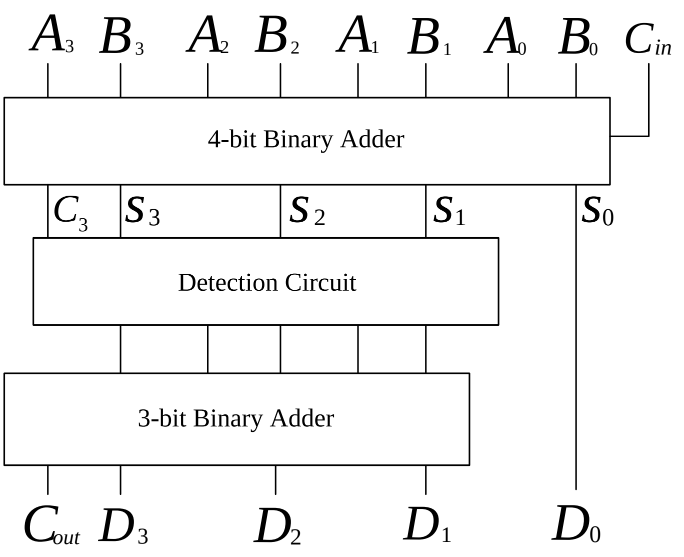
7.1 Overview

With the increasing demand of decimal arithmetic, reversible based implementation of reversible binary coded decimal (BCD) adder has received significant research attention. In Chapter 6, a reversible logic implementation of BCD adder design with primarily optimized in terms of propagation delay is introduced. Parallel execution technique is used to achieve fast BCD sum.

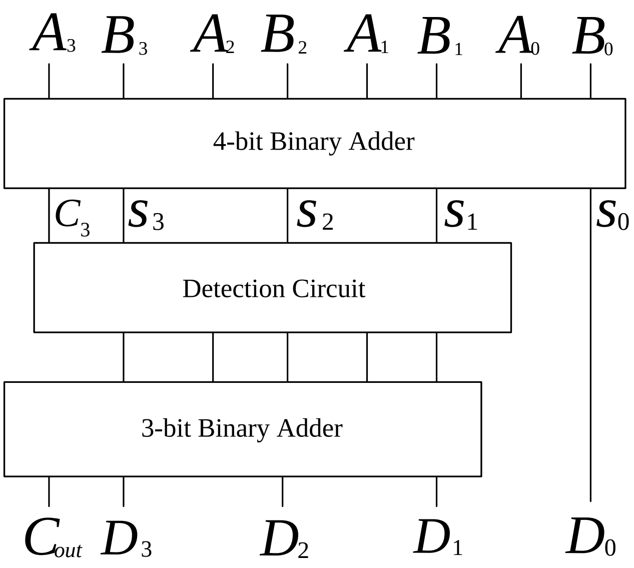
In this chapter, another reversible logic implementation of BCD adder is presented but optimized in terms of the most important aspect for all reversible logic design, the quantum cost. Other important parameters such as ancilla input, garbage output and propagation delay are not neglected and is kept at best possible lowest value. Two BCD adder designs are presented, one with Carry Input C_{in} and another without.

7.2 Design of Reversible BCD Adder

The designs of the reversible BCD adder concepts with C_{in} and without are similar to the one proposed back in Chapter 6. Figure 7.1 shows the two proposed reversible BCD adder designs. There are two major differences between these designs and the designs presented in Chapter 6. The first difference is that the C_{in} for the designs in Chapter 6 are fed to the correction circuit to achieve faster BCD sum whereas the design in Figure 7.1 (a) sums the C_{in} at the early 4-bit binary adder. Since the design in Figure 7.1 (a) does not focus on achieving fast BCD sum, this step is avoided to save quantum cost. The second difference is that the Carry Output C_{out} for this work is generated at the last stage whereas the designs in Chapter 6, the C_{out} is generated at the correction circuit. This requires additional implementation to realize the required function and thus uses slightly more quantum cost. However this situation is avoided in the newly proposed designs as shown in Figure 7.1.


A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0 C_{in}
4-bit Binary Adder
C_3 S_3 S_2 S_1 S_0
Detection Circuit
3-bit Binary Adder
C_{out} D_3 D_2 D_1 D_0

(a)


A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0
4-bit Binary Adder
C_3 S_3 S_2 S_1 S_0
Detection Circuit
3-bit Binary Adder
C_{out} D_3 D_2 D_1 D_0

(b)

Figure 7.1. (a) Proposed reversible BCD adder design with C_{out} (b) Proposed reversbile BCD adder design without C_{out}

7.3 Design of Reversible BCD Adder with Carry Input

In order to have the design as shown in Figure 7.1 (a), the reversible 4-bit binary adder, reversible detection circuit and the reversible 3-bit binary adder are needed to be designed. Details of these designs are explained in the following sub-sections.

7.3.1 Proposed Reversible 4-Bit Binary Adder

The reversible 4-bit binary adder design consists of four full adder circuits. Each of the full adders is designed using the proposed reversible full adder circuit as seen in Figure 7.2. By using the reversible full adder circuit to design the reversible 4-bit binary adder, the overall circuit has a quantum cost of 24, uses 4 ancilla inputs and has 0 garbage output.

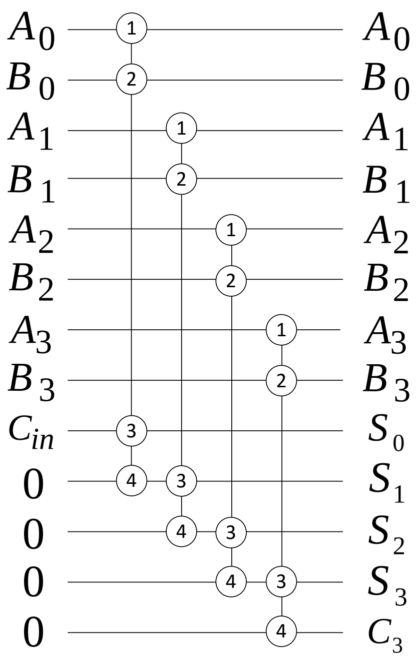


Figure 7.2. Proposed Reversible 4-Bit Binary Adder

7.3.2 Proposed Reversible Detection Circuit

The design of the reversible detection circuit is illustrated in Figure 7.3. In order to get the Boolean function $C_3 + S_3(S_1 + S_2)$, a TR gate with a NOT gate is used to take inputs S_1 and S_2 to generate $\overline{S_1}\overline{S_2}$. Then a TR gate takes the output along with input S_3 to generate output $\overline{S_3} + \overline{S_1}\overline{S_2}$. Then the output is connected to a NOT gate to get $S_3(S_1 + S_2)$. Finally, a CNOT gate with input C_3 is added to obtain the desired function $C_3 \oplus S_3(S_1 + S_2)$. It can be noticed from Figure 7.3 that the function can be written as $C_3 + S_3(S_1 + S_2)$ because C_3 and $S_3(S_1 + S_2)$ cannot be true at the same time. Besides, the input S_1, S_2, S_3 are also regenerated for further use in the reversible 3-bit binary adder. Overall, the circuit has a quantum cost of 13, uses 2 ancilla inputs and has 2 garbage outputs.

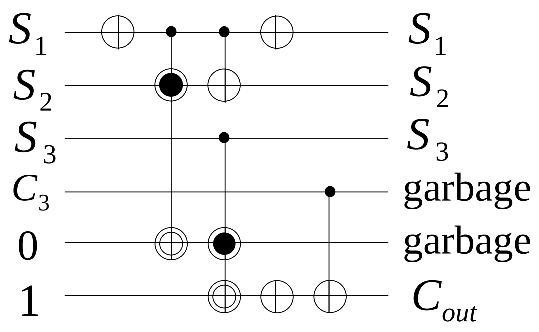


Figure 7.3. Proposed Reversible Detection Circuit

7.3.3 Proposed 3-Bit Binary Adder

The design of the 3-bit binary adder is illustrated in Figure 7.4. The design contains a half adder and two full adders. The half adder is designed using a Peres gate. The full adder at the last bit is used for generating output D_3. In the proposed adder circuit, C_{out} is replaced by a CNOT gate to generate only output D_3. This is because the previous full adder had already generated the carry output C_{out}. Overall, the circuit has a quantum cost of 11, uses 2 ancilla inputs and has 2 garbage outputs.

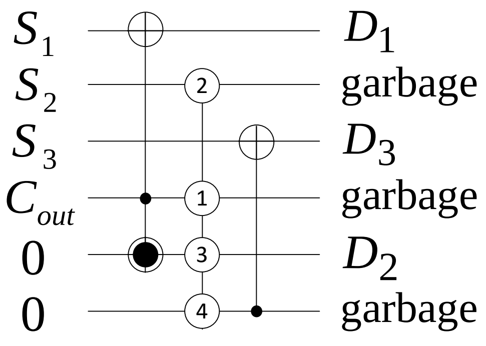


Figure 7.4. Proposed Reversible 3-Bit Binary Adder

7.3.4 Overall Circuit

Integrate all the proposed reversible subcomponents together into Figure 7.1 (a) to design the 1-digit reversible BCD adder with carry input C_{in} is as shown in Figure 7.5. For easier reference, it will be named as QC-BCD1 further in this work. The overall design has a total quantum cost of 48, uses 8 ancilla inputs and has 4 garbage outputs.

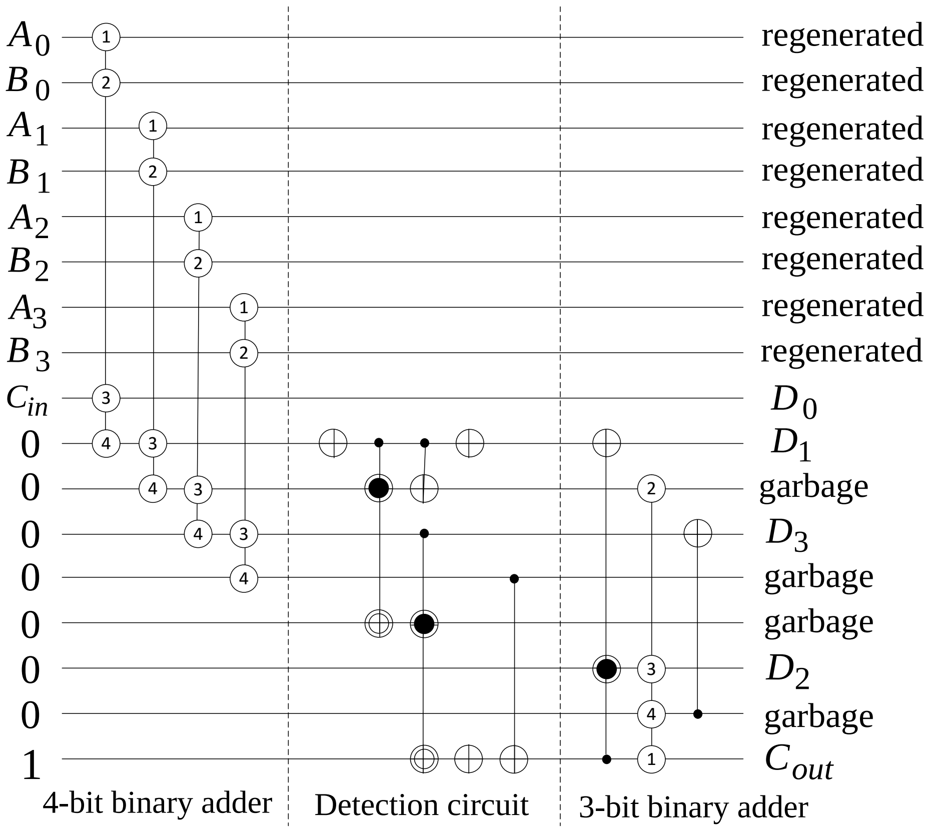


Figure 7.5. Proposed Reversible BCD Adder with Carry Input (QC-BCD1)

7.4 Design of Reversible BCD Adder with No Carry Input

To design the reversible BCD adder with no carry input C_{in} as shown in Figure 7.1 (b), the exact same reversible BCD adder circuit presented in Section 7.3 is used but with a redesigned 4-bit binary adder.

7.4.1 Proposed Reversible 4-Bit Binary Adder

The reversible 4-bit binary adder used for this design is the same as the design proposed back in Chapter 6. Figure 7.6 shows the reversible 4-bit binary adder design. By using this design, the reversible 4-bit adder has a quantum cost of 22, uses 4 ancilla inputs and has 0 garbage outputs.

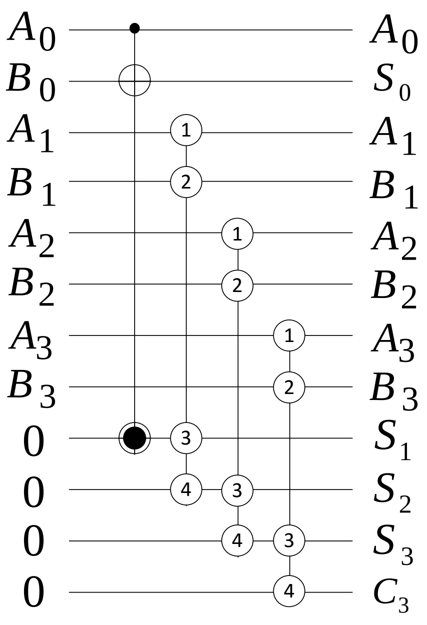


Figure 7.6. Proposed Reversible 4-Bit Binary Adder

7.4.2 Overall Circuit

Integrate all the proposed reversible subcomponents together into Figure 7.1 (b) to design the 1-digit reversible BCD adder with no carry input C_{in} is as shown in Figure 7.7. For easier reference, it will be named as QC-BCD2 further in this work. The overall design has a total quantum cost of 46, uses 8 ancilla input and has 4 garbage outputs.

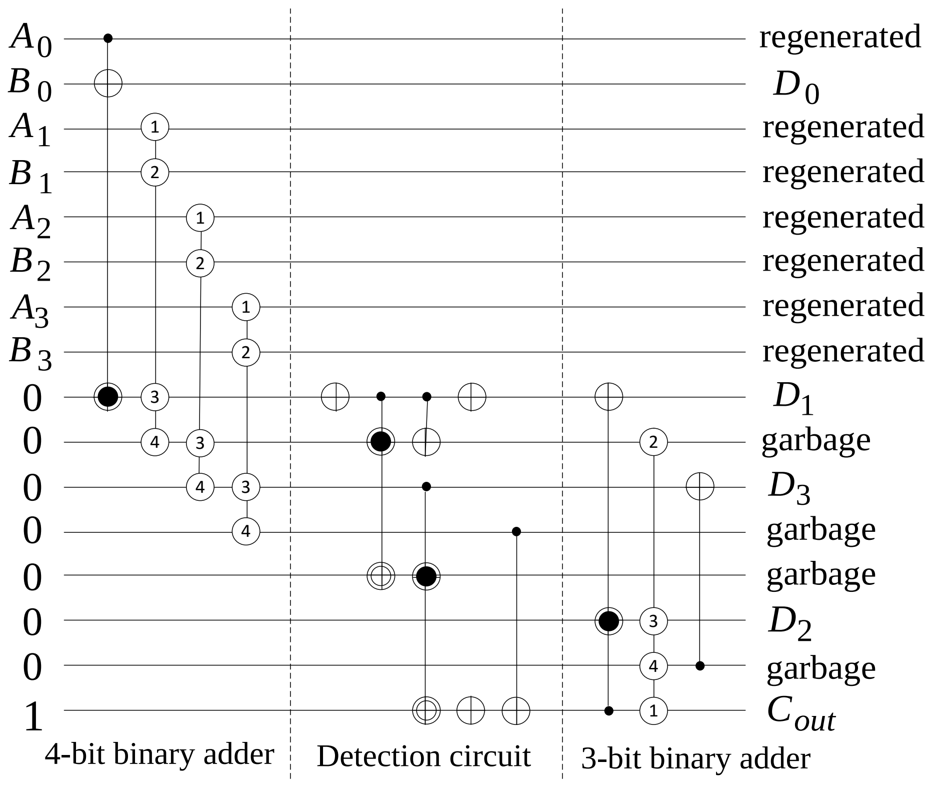


Figure 7.7. Proposed Reversible BCD Adder with No Carry Input (QC-BCD2)

7.5 Optimization in Terms of Propagation Delay

Although the main parameter of the proposed reversible BCD adders is the quantum cost. However other important parameters such as ancilla input, garbage output and propagation delay should also be kept at best possible lowest value. In this section, the propagation delay for the proposed design QC-BCD1 and QC-BCD2 are further optimized.

7.5.1 QC-BCD1 Optimized in Terms of Propagation Delay

From the overall circuit shown in Figure 7.5, the optimized circuit in terms of propagation delay is as illustrated in Figure 7.8. The figure is divided into columns for easier propagation delay calculation. The propagation delay calculation for QC-BCD1 is illustrated in the following steps:

1. Step 1 has 1 reversible full adder circuit. The propagation delay at this step is Δ4.

2. Step 2 has 1 reversible full adder circuit. The propagation delay at this step is Δ4.
3. Step 3 has 1 reversible full adder circuit and 1 NOT gate. The propagation delay at this step is Δ4 because the reversible full adder circuit is working parallel to the NOT gate.
4. Step 4 has 1 reversible full adder circuit and 1 TR gate. The propagation delay at this step is Δ4 because both gates are working parallel.
5. Step 5 has 1 CNOT gate and 1 TR gate. The propagation delay at this step is Δ4 because both of the gates are working parallel.
6. Step 6 has 2 NOT gates. The propagation delay at this step is Δ1 because both gates are working parallel.
7. Step 7 has 1 CNOT gate. The propagation delay at this step is Δ1.
8. Step 8 has 1 Peres gate. The propagation delay at this step is Δ4.
9. Step 9 has 1 reversible full adder circuit. The propagation delay at this step is Δ4.
10. Step 10 has 1 CNOT gate. The propagation delay at this step is Δ1.

Thus the total propagation delay produced is Δ31.

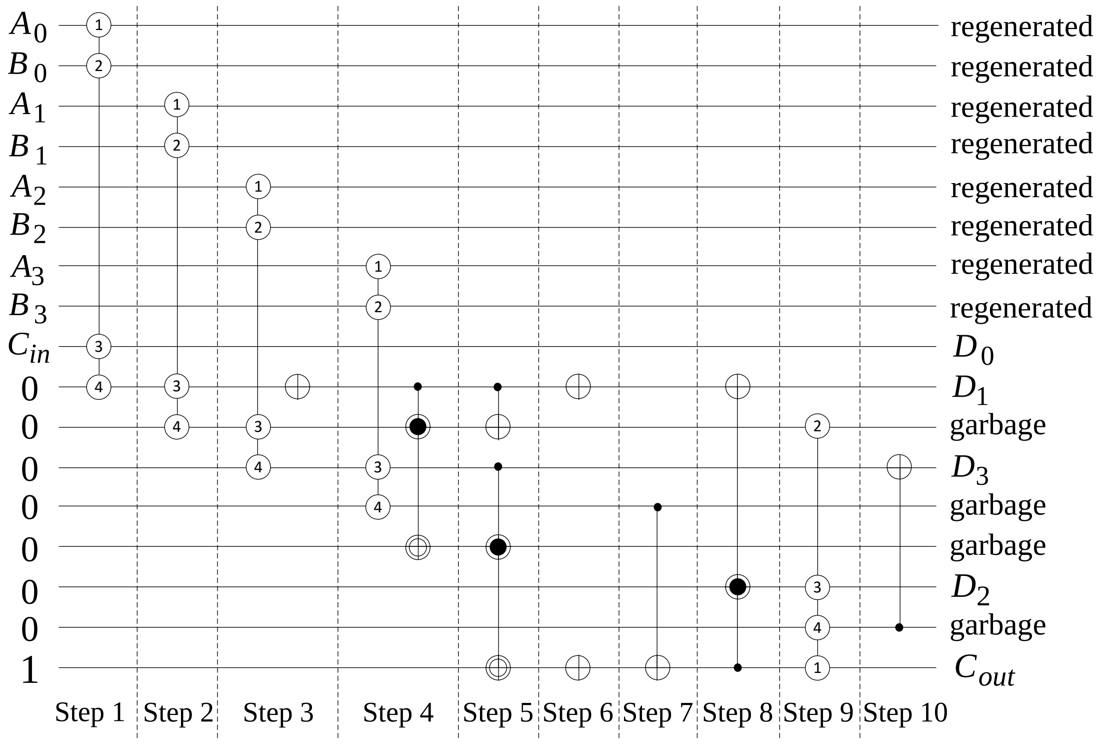


Figure 7.8. Propagation Delay Optimized for QC-BCD1

7.5.2 QC-BCD2 Optimized In Terms of Propagation Delay

From the overall circuit as shown in Figure 7.7, the optimized circuit in terms of propagation delay is illustrated in Figure 7.9. The figure is divided into columns for easier propagation delay calculation. The propagation delay calculation for QC-BCD2 is illustrated in the following steps:

1. Step 1 has 1 Peres gate. The propagation delay at this step is $\Delta 4$.
2. Step 2 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
3. Step 3 has 1 reversible full adder circuit and 1 NOT gate. The propagation delay at this step is $\Delta 4$ because the reversible full adder circuit is working parallel to the NOT gate.
4. Step 4 has 1 reversible full adder circuit and 1 TR gate. The propagation delay at this step is $\Delta 4$ because both gates are working parallel.
5. Step 5 has 1 CNOT gate and 1 TR gate. The propagation delay at this step is $\Delta 4$ because both of the gates are working parallel.
6. Step 6 has 2 NOT gates. The propagation delay at this step is $\Delta 1$ because both gates are working parallel.
7. Step 7 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$.
8. Step 8 has 1 Peres gate. The propagation delay at this step is $\Delta 4$.
9. Step 9 has 1 reversible full adder circuit. The propagation delay at this step is $\Delta 4$.
10. Step 10 has 1 CNOT gate. The propagation delay at this step is $\Delta 1$.

Thus the total propagation delay produced is $\Delta 31$.

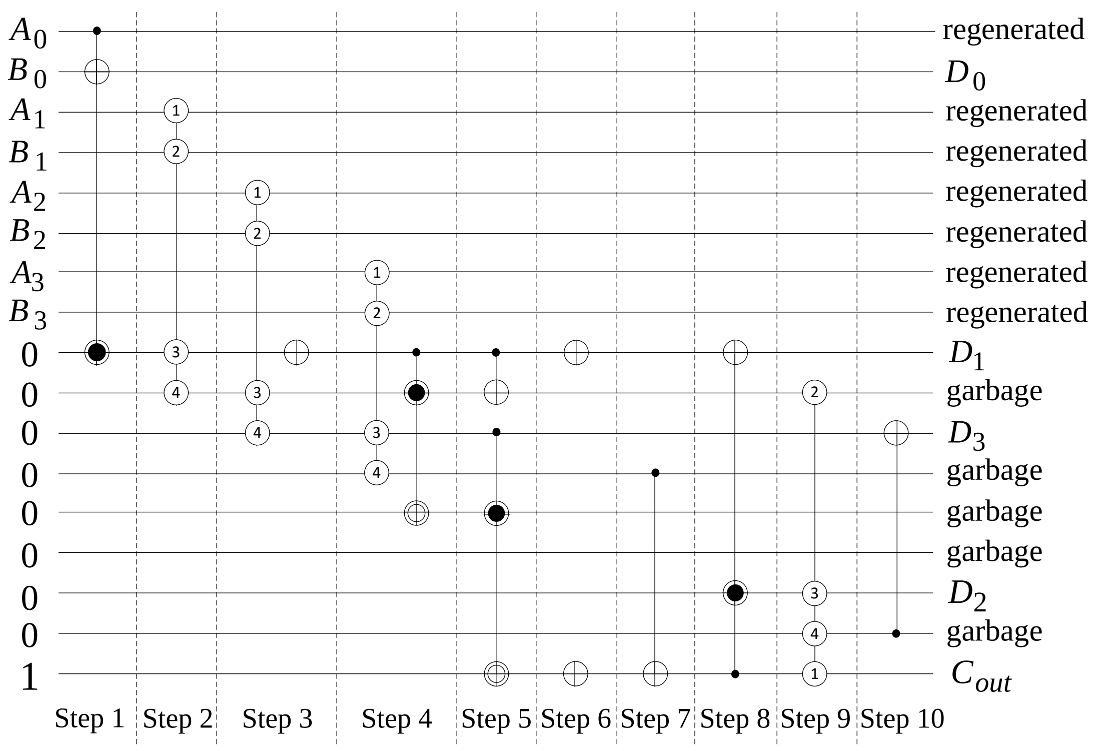


Figure 7.9. Propagation Delay Optimized for QC-BCD2

7.6 Evaluation of the Proposed Designs

The overall design for n digit reversible BCD adder with carry input C_{in} using QC-BCD1 has a total ancilla input of $8n$ and $4n$ garbage outputs. It has a total quantum cost of $48n$ and propagation delay of $\Delta 31$. Whereas for n digit reversible BCD adder with no carry input C_{in} required using QC-BCD1 and QC-BCD2, it has a total ancilla input of $8n$ and $4n$ garbage outputs. It has a total quantum cost of $48n - 2$ and propagation delay of $\Delta 31n$.

A comparison with the existing design is shown in Table 7-1 where n refers to the number of BCD digits connecting together as a n digit BCD adder. According to literature, the reversible BCD adder work proposed by Biswas et al. [103] has the lowest quantum cost, approach 2 of [64] has the lowest ancilla input and garbage output. The proposed work C-BCD2 with NC-BCD2 has the overall least delay. From the comparison, it can be observed that the proposed work improves the quantum cost as compared to [103] by around 4%. Comparing the other parameters with all the existing designs, it can be seen that in the proposed reversible BCD adder designs, these parameters are well maintained at acceptable range.

Table 7-1. Comparison of n digit reversible BCD Adders

	Ancilla Inputs	Garbage Outputs	Quantum Cost	Delay Δ
Babu et al. [104]	$17n$	$18n$	$143n$	$68n+11$
James et al. [96]	$35n$	$35n$	$190n$	$36n+55$
Biswas et al. [103]	$7n$	$7n$	$50n$	$39n+1$
Mohamadi et al. [98]	$16n$	$16n$	$70n$	$52n$
Thomsen et al. [105] (with carry input)	$4n$	$3n$	$200n$	$65n+115$
Thomsen et al. [105] (without carry input)	$4n$	$3n+1$	$200n-113$	$\begin{cases} 67 & n=1 \\ 65n+117 & n>1 \end{cases}$
James et al. [106] (Approach 1)	$8n$	$8n$	$80n$	$35n+24$
James et al. [106] (Approach 2)	$12n$	$12n$	$102n$	$11n+61$
Thapliyal et al. [64] (Approach 1 with carry input)	$2n$	n	$88n$	$66n+1$
Thapliyal et al. [64] (Approach 1 without carry input)	$2n$	n	$88n-12$	$66n+3$
Thapliyal et al. [64] (Approach 2 with carry input)	n	0	$70n$	$52n+1$
Thapliyal et al. [64] (Approach 2 without carry input)	n	0	$70n-12$	$52n+2$
C-BCD1 (with carry input)	$13n$	$9n$	$65n$	$10n+29$
C-BCD2 (with carry input)	$15n$	$11n$	$67n$	$9n+31$
C-BCD1 and NC-BCD1 (without carry input)	$13n-4$	$9n-4$	$65n-19$	$\Delta(10n+21)$
C-BCD2 and NC-BCD2 (without carry input)	$15n-6$	$11n-6$	$67n-22$	$\Delta(9n+24)$
Proposed design 1 (with carry input)	$8n$	$4n$	$48n$	$31n$
Proposed design 2 (without carry input)	$8n$	$4n$	$48n-2$	$31n$

Proposed design 1 is designed using only QC-BCD1
Proposed design 2 is designed using only QC-BCD1 and QC-BCD2

7.7 Simulation and Verification

The simulation and verification of the proposed reversible BCD adders is the same as the one discussed in Section 6.6 of Chapter 6. Firstly, all reversible gates used are created into library using VHDL. Next, each of the proposed reversible BCD adder and its subcomponents circuits are coded individually with the help of the reversible gates created from the library. Finally, a test bench waveform is generated for each of the proposed designs. The test bench waveform shows the input and output behavior of the proposed designs and it is use for comparison with the theoretical result to ensure no mistake is present. All reversible circuits are test bench waveform are simulated and verified using Xilinx ISE 10.1.

7.8 Chapter Summary

In conclusion, two efficient reversible BCD adder designs primarily optimized in terms of quantum cost are presented. Other important parameters such as ancilla input, garbage output and delay are also maintained at acceptable range. Experiment results show that the proposed work has significant improvement in terms of quantum cost as compared to all existing designs.

CHAPTER 8 CONCLUSION AND FUTURE WORK

The promising application of reversible logic has offered attractive significant research attention in the aspect of design and synthesis. The research efforts reported in this dissertation represent a solid contribution towards the advancement in the area of synthesis of reversible logic and designs of reversible logic arithmetic applications. Some final remarks and possible future works are given in Section 8.1 and Section 8.2 respectively.

8.1 Concluding Remarks

In this dissertation, reversible logic synthesis algorithms and reversible logic based arithmetic applications have been investigated. In Chapter 4, a heuristic search-based reversible logic synthesis algorithm using mixed-polarity based Toffoli gates is proposed. This is the first attempt in literature to integrate polarity control capability into search-based synthesis algorithm. Ability to synthesize reversible functions in terms of quantum cost is also added which had not addressed for most existing synthesis algorithms. The proposed synthesis algorithm is then applied to synthesize all three variables based reversible functions and reversible benchmarks circuits and achieved results that are closed to the optimal solutions.

In Chapter 5, a new reversible gate is designed and developed mainly to serve as a reversible full adder circuit and it is proven to be efficient in terms of ancilla input, garbage output, quantum cost and delay. The proposed reversible gate is designed by implementing the proposed synthesis algorithm in Chapter 4 to synthesize the ideal reversible full adder. The resulting circuit is then optimized in terms of quantum cost and delay. Quantum cost optimization is done by decomposing the circuit into its primitive gate form and striking off all repeated terms. Delay reduction is performed on the quantum cost optimized circuit by applying the moving rules described in [99].

In Chapter 6, several designs of minimal delay reversible BCD adders are presented. The reversible BCD adders are designed to delay the need of the carry input C_{in} and generate the required carry output C_{out} for the next BCD digit faster. This method allows simultaneous run of multiple BCD digits and achieves high speed BCD sum. The reversible 4-bit binary adder of the proposed reversible adders are designed with the help of the earlier proposed reversible full adder circuit which is introduced in Chapter 5 and the reversible correction circuits are partially designed with the help of the earlier proposed synthesis algorithm which is described in Chapter 4. The proposed reversible BCD adder designs have shown to be better in terms of propagation delay as compared to other existing literature designs as discussed.

In Chapter 7, two quantum cost optimized reversible BCD adder designs are presented, one with Carry Input C_{in} and one without. Quantum cost is one of the most important parameter to consider for all reversible logic design and synthesis and it should not be neglected in any condition. The BCD adder designs are designed with the help of the earlier work presented in Chapter 6. The proposed reversible BCD adder designs have better performance in terms of quantum cost as compared to other existing literature designs as discussed.

8.2 Future Works

As research on reversible logic continues to grow, further improvement can always be done to improve the existing algorithms using better design tools and reversible gates. Some of the future works and interesting research ideas to improve the current work are listed as follows.

1. The reversible logic synthesis algorithm in Chapter 4 can be optimized to improve the synthesis time.
2. To find better synthesis result, powerful algorithm such as genetic algorithm and ant colony optimization algorithm can be integrated to the synthesis algorithm which is proposed in Chapter 4.
3. The input function to the synthesis algorithm in Chapter 4 is fixed by the user. It is worth to investigate the modification such as allowing the algorithm to determine the best possible ancilla inputs and garbage outputs condition that will yield the best synthesize solution.

4. Synthesize reversible function in terms of propagation delay can be added to the synthesis algorithm in Chapter 4.
5. Reversible gate can be decomposed into their primitive gate form and used for reversible logic synthesis and logic design. This can be integrated into the proposed synthesis algorithm in Chapter 4 and the proposed reversible BCD adder designs in Chapter 6 and Chapter 7 for better results.

REFERENCES

[1] R. Landauer, "Irreversibility and Heat Generation in the Computing Process," *IBM Journal of Research and Development*, vol. 5, pp. 183-191, 1961.

[2] J. Von Neumann, *Theory of Self-Reproducing Automata*. Urbana: University of Illinois Press, 1966.

[3] R. Wille, M. Soeken, and R. Drechsler, "Reducing the Number of Lines in Reversible Circuits," in *Proceedings of the 47th Design Automation Conference*, New York, 2010, pp. 647-652.

[4] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, pp. 82-85, 1998.

[5] C. A. Mack, "Fifty Years of Moore's Law," *Semiconductor Manufacturing, IEEE Transactions on*, vol. 24, pp. 202-207, 2011.

[6] C. H. Bennett, "Logical Reversibility of Computation," *IBM Journal of Research and Development*, vol. 17, pp. 525-532, 1973.

[7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*: Cambridge University Press, 2000.

[8] V. Vedral, A. Barenco, and A. Ekert, "Quantum networks for elementary arithmetic operations," *Physical Review A*, vol. 54, p. 147, 1996.

[9] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM journal on Computing*, vol. 26, pp. 1510-1523, 1997.

[10] D. Bacon and W. VAn DAm, "Recent progress in quantum algorithms," *Communications of the ACM*, vol. 53, pp. 84-93, 2010.

[11] D. P. DiVincenzo, "Quantum computation," *Science*, vol. 270, pp. 255-261, 1995.

[12] N. D. Mermin, *Quantum computer science: an introduction*: Cambridge University Press, 2007.

[13] A. M. Childs and W. Van Dam, "Quantum algorithms for algebraic problems," *Reviews of Modern Physics*, vol. 82, p. 1, 2010.

[14] D. Maslov, "Reversible Logic Synthesis," Ph.D, Dept. Comput. Sci., Univ. New Brunswick, Fredericton, NB, Canada, 2003.

[15] J. G. Koller and W. C. Athas, "Adiabatic switching, low energy computing, and the physics of storing and erasing information," in *Proceedings of Physics of Computation Workshop*, 1992.

[16] W. C. Athas and L. Svensson, "Reversible logic issues in adiabatic CMOS," in *Physics and Computation, 1994. PhysComp'94, Proceedings., Workshop on*, 1994, pp. 111-118.

[17] E. Knill, R. La Amme, and G. L. Milburn, "A Scheme for Efficient Quantum Computation with Linear Optics," *Nature,* vol. 409, pp. 46-52, 2001.

[18] C. Taraphdar, T. Chattopadhyay, and J. N. Roy, "Mach–Zehnder interferometer-based all-optical reversible logic gate," *Optics & Laser Technology,* vol. 42, pp. 249-259, 2010.

[19] R. Forsati, S. V. Ebrahimi, K. Navi, E. Mohajerani, and H. Jashnsaz, "Implementation of all-optical reversible logic gate based on holographic laser induced grating using azo-dye doped polymers," *Optics & Laser Technology,* vol. 45, pp. 565-570, 2013.

[20] R. C. Merkle, "Reversible electronic logic using switches," *Nanotechnology,* vol. 4, p. 21, 1993.

[21] C. S. Lent, P. D. Tougaw, W. Porod, and G. H. Bernstein, "Quantum cellular automata," *Nanotechnology,* vol. 4, p. 49, 1993.

[22] H. Cho and E. E. Swartzlander, "Adder designs and analyses for quantum-dot cellular automata," *Nanotechnology, IEEE Transactions on,* vol. 6, pp. 374-383, 2007.

[23] X. Ma, J. Huang, C. Metra, and F. Lombardi, "Reversible gates and testability of one dimensional arrays of molecular QCA," *Journal of Electronic Testing,* vol. 24, pp. 297-311, 2008.

[24] R. Garipelly, P. M. Kiran, and A. S. Kumar, "A review on reversible logic gates and their implementation," *International Journal of Emerging Technology and Advanced Engineering,* vol. 3, pp. 417-423, 2013.

[25] K. Iwama, Y. Kambayashi, and S. Yamashita, "Transformation Rules for Designing CNOT-based Quantum Circuits," in *Proceedings of the 39th annual Design Automation Conference*, New York, 2002, pp. 419-424.

[26] D. M. Miller, D. Maslov, and G. W. Dueck, "A Transformation Based Algorithm for Reversible Logic Synthesis," in *Proceedings of the 40th annual Design Automation Conference*, New York, 2003, pp. 318-323.

[27] 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, pp. 710-722, 2003.

[28] G. W. Dueck, D. Maslov, and D. M. Miller, "Transformation-based synthesis of networks of Toffoli/Fredkin gates," in *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, 2003, pp. 211-214.

[29] D. Maslov, G. W. Dueck, and D. M. Miller, "Synthesis of Fredkin-Toffoli reversible networks," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 13, pp. 765-769, 2005.

[30] J. Donald and N. K. Jha, "Reversible Logic Synthesis with Fredkin and Peres Gates," *ACM Journal on Emerging Technologies in Computing Systems,* vol. 4, 2008.

[31] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, *et al.*, "Elementary Gates for Quantum Computation," *Physical Review A,* vol. 52, pp. 3457-3467, 11/01/ 1995.

[32] C. Moraga, "Using negated control signals in quantum computing circuits," *Facta universitatis-series: Electronics and Energetics,* vol. 24, pp. 423-435, 2011.

[33] M. Arabzadeh, M. Saeedi, and M. Saheb Zamani, "Rule-Based Optimization of Reversible Circuits," in *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, New Jersey, 2010, pp. 849-854.

[34] M. Szyprowski and P. Kerntopf, "Reducing Quantum Cost in Reversible Toffoli Circuits," *arXiv preprint arXiv:1105.5831*, 2011.

[35] C. Moraga, "Hybrid GF(2) – Boolean Expressions ..for Quantum Computing Circuits," in *Reversible Computation*. vol. 7165, A. Vos and R. Wille, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 54-63.

[36] M. Szyprowski and P. Kerntopf, "Optimal 4-bit Reversible Mixed-Polarity Toffoli Circuits," in *Reversible Computation*. vol. 7581, R. Glück and T. Yokoyama, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 138-151.

[37] D. Maslov, G. Dueck, and N. Scott, "Reversible logic synthesis benchmarks page," *Online: http://www.cs.uvic.ca/~dmaslov*, 2005.

[38] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler, "RevLib: An online resource for reversible functions and reversible circuits," in *Multiple Valued Logic, 2008. ISMVL 2008. 38th International Symposium on*, 2008, pp. 220-225.

[39] S. C. Chua, A. K. Singh, and L. Gopal, "Efficient Three Variables Reversible Logic Synthesis Using Mixed-polarity Toffoli Gate," *Procedia Computer Science*, vol. 70, pp. 362-368, 2015.

[40] Y. Hardy and W.-H. Steeb, "Boolean Functions, Quantum Gates, Hamilton Operators, Spin Systems and Computer Algebra," *arXiv preprint arXiv:1401.2248*, 2014.

[41] J. E. Rice, "An introduction to reversible latches," *The Computer Journal*, vol. 51, pp. 700-709, 2008.

[42] Z. Zilic, K. Radecka, and A. Kazamiphur, "Reversible circuit technology mapping from non-reversible specifications," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*, 2007, pp. 1-6.

[43] M. Saeedi and I. L. Markov, "Synthesis and optimization of reversible circuits—a survey," *ACM Computing Surveys (CSUR)*, vol. 45, p. 21, 2013.

[44] T. Thapliyal and N. Ranganathan, "Design of Reversible Sequential Circuits Optimizing Quantum Cost, Delay, and Garbage Outputs," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 6, pp. 1-31, 2010.

[45] S. Xu, "Reversible Logic Synthesis with Minimal Usage of Ancilla Bits," *arXiv preprint arXiv:1506.03777*, 2015.

[46] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, pp. 219-253, 1982/04/01 1982.

[47] M. Perkowski, L. Jozwiak, P. Kerntopf, A. Mishchenko, and A. Al-Rabadi, "A General Decomposition for Reversible Logic," *Proceedings of 5th Reed-Muller Workshop*, pp. 119-138, 2001.

[48] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Optimal Synthesis of Multiple Output Boolean Functions using a Set of Quantum Gates by Symbolic Reachability," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 1652-1663, 2006.

[49] A. De Vos, B. Desoete, A. Adamski, P. Pietrzak, M. Sibiński, and T. Widerski, "Design of reversible logic circuits by means of control gates," in *Integrated Circuit Design*, ed: Springer, 2000, pp. 255-264.

[50] Z. Sasanian and D. M. Miller, "Transforming MCT Circuits to NCVW Circuits," in *Reversible Computation*. vol. 7165, A. De Vos and R. Wille, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 77-88.

[51] D. Maslov and D. M. Miller, "Comparison of the cost metrics for reversible and quantum logic synthesis," *arXiv preprint quant-ph/0511008,* 2005.

[52] D. Maslov and G. W. Dueck, "Improved quantum cost for n-bit Toffoli gates," *Electronics Letters,* vol. 39, pp. 1790-1791, 2003.

[53] T. Toffoli, "Reversible computing," in *Automata, Languages and Programming*. vol. 85, J. de Bakker and J. van Leeuwen, Eds., ed: Springer Berlin Heidelberg, 1980, pp. 632-644.

[54] O. Golubitsky and D. Maslov, "A Study of Optimal 4-Bit Reversible Toffoli Circuits and Their Synthesis," *Computers, IEEE Transactions on,* vol. 61, pp. 1341-1353, 2012.

[55] C. M. Wilmott and P. R. Wild, "On a generalized quantum SWAP gate," *International Journal of Quantum Information,* vol. 10, p. 1250034, 2012.

[56] M. Lukac, M. Perkowski, H. Goi, M. Pivtoraiko, C. H. Yu, K. Chung, *et al.*, "Evolutionary approach to quantum and reversible circuits synthesis," in *Artificial intelligence in logic design*, ed: Springer, 2004, pp. 201-257.

[57] C. P. Williams, *Explorations in quantum computing*: Springer Science & Business Media, 2010.

[58] 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, p. 2855, 1996.

[59] W. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski, "Quantum logic synthesis by symbolic reachability analysis," in *Proceedings of the 41st annual Design Automation Conference*, 2004, pp. 838-841.

[60] G. Yang, W. N. Hung, X. Song, and M. Perkowski, "Majority-based reversible logic gates," *Theoretical computer science,* vol. 334, pp. 259-274, 2005.

[61] H. Chau and F. Wilczek, "Simple realization of the Fredkin gate using a series of two-body operators," *Physical review letters*, vol. 75, p. 748, 1995.

[62] J. Bruce, M. 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*, 2002, pp. 74-79.

[63] A. Peres, "Reversible Logic and Quantum Computers," *Physic Review*, vol. 32, pp. 3266-3276, 1985.

[64] H. Thapliyal and N. Ranganathan, "Design of efficient reversible logic-based binary and BCD adder circuits," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 9, p. 17, 2013.

[65] D. M. Miller, "Lower cost quantum gate realizations of multiple-control Toffoli gates," in *Communications, Computers and Signal Processing, 2009. PacRim 2009. IEEE Pacific Rim Conference on*, 2009, pp. 308-313.

[66] H. Thapliyal and N. Ranganathan, "Design of Efficient Reversible Binary Subtractors Based on A New Reversible Gate," 2009, pp. 229-234.

[67] M. Mohammadi and M. Eshghi, "On figures of merit in reversible and quantum logic designs," *Quantum Information Processing*, vol. 8, pp. 297-318, 2009.

[68] R. Drechsler and R. Wille, "From Truth Tables to Programming Languages: Progress in the Design of Reversible Circuits," *IEEE International Symposium on Multiple-Valued Logic*, pp. 78–85, 2011.

[69] L. K. Wang, M. A. Erle, C. Tsen, E. M. Schwarz, and M. J. Schulte, "A survey of hardware designs for decimal arithmetic," *IBM J. Res. Dev.*, vol. 54, pp. 216-230, 2010.

[70] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. C-35, pp. 677-691, 1986.

[71] A. M. Jabir, D. K. Pradhan, A. K. Singh, and T. L. Rajaprabhu, "A Technique for Representing Multiple Output Binary Functions with Applications to Verification and Simulation," *IEEE Transactions on Computers*, vol. 56, pp. 1133-1145, 2007.

[72] G. D. Hachtel and F. Somenzi, *Logic synthesis and verification algorithms*: Springer Science & Business Media, 2006.

[73] X. Zang, H. Sun, and K. S. Trivedi, "A BDD-based algorithm for reliability analysis of phased-mission systems," *Reliability, IEEE Transactions on,* vol. 48, pp. 50-60, 1999.

[74] T. Sasao, "EXMIN2: a simplification algorithm for exclusive-OR-sum-of-products expressions for multiple-valued-input two-valued-output functions," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 12, pp. 621-632, 1993.

[75] A. Younes* and J. F. Miller, "Representation of Boolean quantum circuits as Reed–Muller expansions," *International journal of electronics,* vol. 91, pp. 431-444, 2004.

[76] R. Drechsler, M. Theobald, and B. Becker, "Fast OFDD-based minimization of fixed polarity Reed-Muller expressions," *Computers, IEEE Transactions on,* vol. 45, pp. 1294-1299, 1996.

[77] D. Maslov, G. W. Dueck, and D. M. Miller, "Toffoli Network Synthesis With Templates," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 24, pp. 807-817, 2005.

[78] D. Maslov, G. W. Dueck, and D. M. Miller, "Techniques for the Synthesis of Reversible Toffoli Networks," *ACM Transactions on Design Automation of Electronic Systems,* vol. 12, 2007.

[79] K. Datta, G. Rathi, R. Wille, I. Sengupta, H. Rahaman, and R. Drechsler, "Exploiting negative control lines in the optimization of reversible circuits," in *Reversible Computation*, ed: Springer, 2013, pp. 209-220.

[80] G. Yang, X. Song, H. W.N.N., F. Xie, and M. Perkowski, "Group Theory Based Synthesis of Binary Reversible Circuits," 2006, pp. 365-374.

[81] 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 of Emerging Technologies in Computing Systems,* vol. 2, pp. 277-293, 2006.

[82] Z. Sasanian, M. Saeedi, M. Sedighi, and M. S. Zamani, "A Cycle-Based Synthesis Algorithm for Reversible Logic," in *Design Automation Conference*, 2009, pp. 745-750.

[83] M. Saeedi, M. Saheb Zamani, M. Sedighi, and Z. Sasanian, "Reversible Circuit Synthesis Using a Cycle-Based Approach," *ACM Journal on Emerging Technologies in Computing Systems,* vol. 6, 2010.

[84] M. Saeedi, M. Sedighi, and M. Saheb Zamani, "A Library-Based Synthesis Methodology for Reversible Logic," *Microelectronics Journal,* vol. 41, pp. 185-194, 2010.

[85] P. Kerntopf, "A New Heuristic Algorithm for Reversible Logic Synthesis," in *Proceedings of 41st Design Automation Conference*, Poland, 2004, pp. 843-837.

[86] R. Wille and R. Drechsler, "BDD-Based Synthesis of Reversible Logic for Large Functions," in *Proceedings of the 46th Annual Design Automation Conference*, New York, 2009, pp. 270-275.

[87] M. Krishna and A. Chattopadhyay, "Efficient Reversible Logic Synthesis via Isomorphic Subgraph Matching," in *Multiple-Valued Logic (ISMVL), 2014 IEEE 44th International Symposium on*, 2014, pp. 103-108.

[88] M. Soeken, L. Tague, G. W. Dueck, and R. Drechsler, "Ancilla-free synthesis of large reversible functions using binary decision diagrams," *Journal of Symbolic Computation,* vol. 73, pp. 1-26, 2016.

[89] 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, pp. 2317-2330, 2006.

[90] M. Saeedi, M. Saheb Zamani, and M. Sedighi, "On the Behavior of Substitution-based Reversible Circuit Synthesis Algorithm Investigation and Improvement," Washington, 2007, pp. 428-436.

[91] Z. Yexin and H. Chao, "A novel Toffoli network synthesis algorithm for reversible logic," in *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, 2009, pp. 739-744.

[92] O. Golubitsky, S. M. Falconer, and D. Maslov, "Synthesis of the Optimal 4-bit Reversible Circuits," in *Proceedings of the 47th Design Automation Conference*, 2010, pp. 653-656.

[93] M. Szyprowski and P. Kerntopf, "An Approach to Quantum Cost Optimization in Reversible Circuits," in *Nanotechnology (IEEE-NANO), 2011 11th IEEE Conference on*, 2011, pp. 1521-1526.

[94] Z. Li, H. Chen, X. Song, and M. Perkowski, "A Synthesis Algorithm for 4-Bit Reversible Logic Circuits with Minimum Quantum Cost," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 11, p. 29, 2014.

[95] A. B. Khlopotine, M. A. Perkowski, and P. Kerntopf, "Reversible Logic Synthesis by Iterative Compositions," in *IWLS*, 2002, pp. 261-266.

[96] R. K. James, T. K. Shahana, K. P. Jacob, and S. Sasi, "Quick Addition of Decimals Using Reversible Conservative Logic," in *15th International Conference on Advanced Computing & Communication ADCOM 2007*, 2007, pp. 191-196.

[97] S. Islam and M. R. Islam, "Minimization of reversible adder circuits," *Asian Journal of Information Technology*, vol. 4, pp. 1146-1151, 2005.

[98] M. Mohammadi, M. Eshghi, M. Haghparast, and A. Bahrololoom, "Design and optimization of reversible BCD adder/subtractor circuit for quantum and nanotechnology based systems," *World Applied Sciences Journal*, vol. 4, pp. 787-792, 2008.

[99] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne, "Quantum Circuit Simplification and Level Compaction," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, pp. 436-444, 2008.

[100] N. M. Nayeem, L. Jamal, and H. M. Babu, "Efficient reversible Montgomery multiplier and its application to hardware cryptography," *Journal of computer science*, vol. 5, p. 49, 2009.

[101] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, 2 ed.: Oxford University Press, 2010.

[102] J. P. Hayes, *Computer Architecture and Organization*, 3 ed.: McGraw-Hill, 1998.

[103] A. K. Biswas, M. M. Hasan, A. R. Chowdhury, and H. M. Hasan Babu, "Efficient approaches for designing reversible Binary Coded Decimal adders," *Microelectronics journal*, vol. 39, pp. 1693-1703, 2008.

[104] H. M. Hasan Babu and A. R. Chowdhury, "Design of a compact reversible binary coded decimal adder circuit," *Journal of Systems Architecture*, vol. 52, pp. 272-282, 5// 2006.

[105] M. K. Thomsen and R. Glück, "Optimized reversible binary-coded decimal adders," *J. Syst. Archit.*, vol. 54, pp. 697-706, 2008.

[106] R. K. James, K. Poulose Jacob, and S. Sasi, "Reversible Binary Coded Decimal Adders using Toffoli Gates," in *Advances in Computational Algorithms and Data Analysis*. vol. 14, S.-I. Ao, B. Rieger, and S.-S. Chen, Eds., ed: Springer Netherlands, 2009, pp. 117-131.

[107] A. M. M. H. Khan, "Design of Full Adder with Reversible Gate," presented at the International Conference on Computer and Information Technology, Dhaka, Bangladesh, 2002.

[108] M. Saeedi, M. Sedighi, and M. Saheb Zamani, "A Novel Synthesis Algorithm for Reversible Circuits," in *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design*, Piscataway, 2007, pp. 65-68.

[109] E. K. Ardestani, M. S. Zamani, and M. Sedighi, "A Fast Transformation-Based Synthesis Algorithm for Reversible Circuits," in *Digital System Design Architectures, Methods and Tools, 2008. DSD'08. 11th EUROMICRO Conference on*, 2008, pp. 803-806.

[110] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products," in *Int'l Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, 2001, pp. 242-250.