

Logic Synthesis and Optimization of Reversible and Quantum Circuits for Arithmetic Logic Units

Md Belayet Ali

Department of Electrical Engineering and Computer Science
Graduate School of Engineering
Iwate University

A thesis submitted in partial fulfillment of the requirements for the
degree of

Doctor of Philosophy

March 2019

Dedication

To my parents, who taught me the most important values in life.

ACKNOWLEDGEMENT

First and foremost, I would like to thank my advisor Takashi Hirayama from the Department of Electrical Engineering and Computer Science, Faculty of Engineering, Iwate University, Japan. It has been an honor to be his first Ph.D. student. I appreciate all his contributions, especially his time and ideas, to make my Ph.D. experience productive and stimulating. The joy and enthusiasm he has for his research were contagious and motivated me during tough times in the pursuit of the degree. I am also thankful to him for serving as an excellent example, both as a successful supervisor and human being.

I would also like to thank Yasuaki Nishitani, retired Professor of the Department of Electrical Engineering and Computer Science, Faculty of Engineering, Iwate University, Japan, for his valuable discussions, comments, and encouragements in this research.

I would also like to thank Katsuhisa Yamanaka, Assistant Professor of the Department of Electrical Engineering and Computer Science, Faculty of Engineering, Iwate University, Japan, for being part of my thesis committee, for his support, and for being an inspiration in many ways.

I would also like to thank the members of the Hirayama-Yamanaka (former Nishitani-Hirayama) laboratory contributing to my personal and professional time at Iwate University. They have been a source of friendships as well as good advice and collaboration. I am especially grateful to the following members of the former Nishitani-Hirayama laboratory who stuck with me during my Master's course: Tadayoshi Miura, Takafumi Omori, Yoji Sato, Kota Abe, and Gaku Yasui. I would like to especially acknowledge Miura-san and his parents; he was my Master's classmate, lab mate, and tutor. I would also like to thank the following past and present lab members that I have had the pleasure to work with: graduate students Hayato Sugawara, Masatoshi Murakami, Hiroki Kaga, Rin Suzuki, Akihiro Yokota, Takafumi Maki, and Koki Kainuma; undergraduate students Koki Takahashi, Toshiki Matsuno, Seunghun

Shin, and Dai Ito; and the numerous 3rd and 4th (grade) students who have come through the lab.

I grateful to several funding sources that made my Ph.D. work possible. I was funded by the Ministry of Education, Culture, Sports, Science, and Technology, also known as MEXT, Monkasho, and formerly the Ministry of Education, Science, and Culture, which is a ministry of the Japanese government who supported my initial six months as a research student, two years as a Master's student, and three years as a Ph.D. student. I was also supported by the Faculty of Engineering, Iwate University.

My time at Morioka, Iwate was made enjoyable in large part due to my wife, my dearest son, and many friends and groups that have become a part of my life. I am grateful for the time spent with my wife Nishat Tasmin, and also with my son Mohammad Yusuf Ibn Ali, and for preparing delicious meals, for the memorable trips to Miyako city and Mount. Fuji, and for her great support as I finished up my degree. My time at Morioka, Iwate was also enriched by the International Students Association, Morioka Cricket Club, and the International Students Soccer team.

I would like to thank my family for all their love and encouragement. First, to my parents who raised and supported me in all my pursuits. And most of all to my loving, supportive, encouraging, and patient wife Nishat, whose faithful support during the final stages of the Ph.D. is very much appreciated. Thank you.

Finally, I thank the Almighty Lord for keeping me in good health and spirits throughout my research work.

March 2019 Md Belayet Ali

Abstract

Reversible computing is generally considered to be an unconventional form of computing, which has drawn considerable attention from researchers in order to design low-power computing devices. Furthermore, reversible logic plays an important role in the field of quantum computation. However, the synthesis of reversible logic for building quantum circuits is notably dissimilar from the synthesis of non-reversible logic. In the process of reversible logic synthesis, the design of reversible functions is very important and the embedding of irreversible functions into reversible functions is required before they can be applied to existing synthesis methods.

In this thesis, we focus on logic synthesis and optimization of reversible and quantum circuits for arithmetic logic units (ALUs). The main concept of our approach is different from those of existing related studies; in particular, we put emphasis on function design. ALU is a key element for any programmable computing device. In the design of ALUs, an adder/subtractor block is another important key element. A faster adder/subtractor block help improve the efficiency of the ALUs performance and that of the whole system. In this thesis, we propose a design of reversible adder/subtractor blocks and ALUs. Our approach to investigate the reversible functions includes (a) the embedding of irreversible functions into incompletely-specified reversible functions, (b) the operation assignment (OA), and (c) the permutation of function outputs. Moreover, we provide some extensions of these techniques to further improve the design of reversible functions.

We propose a minimization algorithm to obtain minimum multiple-control Toffoli (MCT) circuits. The algorithm comprises the OA and the permutation of function outputs to reduce the number of gates in a circuit. We use hash tables of minimum MCT circuits, which are simply constructed using an exhaustive enumeration of gate combinations. The resulting reversible circuits are smaller than existing designs in terms of the number of MCT gates. To evaluate the quality of the reversible circuits, we also propose a greedy algorithm and obtain reduced quantum circuits. The results

demonstrate the superiority of our realization of adder/subtractor blocks and ALUs with respect to quantum cost.

Contents

DEDICATION	3
ACKNOWLEDGEMENT	5
Abstract	7
1 Introduction	1
1.1 Introduction	1
2 Basic Definitions	7
2.1 Boolean Logic Functions	7
2.2 Reversible Logic Functions	8
2.3 Reversible Gates	9
2.4 Reversible Circuits	10
2.5 Quantum Computation	10
2.6 Quantum Gates	11
2.7 Quantum Circuits	14
3 Literature Review	17
3.1 Related Works	17
4 Design of Reversible Functions	19
4.1 Embedding Irreversible Functions into Incompletely-Specified Reversible Functions	19
4.2 Operation Assignment (OA)	21
4.3 Permutation of Function Outputs	23
5 Synthesis of Minimum Reversible Circuits	25
5.1 Minimization Algorithm	25
5.2 Reversible Adder/Subtractor	26

5.2.1	Half Adder/Subtractor	26
5.2.2	Full Adder/Subtractor	30
5.3	Simple Reversible Arithmetic Logic Units (ALUs)	33
5.3.1	1-bit Arithmetic Logic Unit (ALU) Benchmarks	34
5.3.2	Revised Arithmetic Logic Units (ALUs)	34
5.4	Experimental Results	35
6	Quantum Circuit Synthesis	41
6.1	Reduction of Quantum Costs (QCs)	42
6.2	Comparison of Quantum Costs (QCs)	44
6.3	Experimental Results	45
7	Conclusions and Future Works	49
A	Truth Table Representation of Reversible Arithmetic Logic Units (ALUs)	53
	Bibliography	57

List of Figures

1.1	Overview of the design flow of reversible synthesis	2
2.1	MCT gate library: (a) NOT, (b) CNOT, and (c) Toffoli.	9
2.2	Reversible circuit.	10
2.3	Splitting and merging rules	12
2.4	Deletion rule.	12
2.5	Merged 2-qubit gate.	13
2.6	Quantum gate representation of the Fredkin Gate.	14
2.7	Quantum circuit.	15
4.1	(a) Circuit realizing ALU. (b) Circuit realizing $OAEC(ALU)$	23
4.2	Circuit realizing $PEC(ALU)$	24
5.1	Simple minimization algorithm with up to m gates	27
5.2	Proposed reversible half adder/subtractor circuit with MCT gates.	29
5.3	Proposed reversible full adder/subtractor circuit with MCT gates.	31
5.4	Mini-ALU: Minimum MCT circuit and its OA.	34
5.5	Gupta's-LU: Minimum MCT circuit and its OA.	34
5.6	ALU: Minimum MCT circuit and its OA.	35
5.7	LU: Minimum MCT circuit and its OA.	35
6.1	Simple algorithm to obtain reduced quantum circuits.	42
6.2	Reduced half adder/subtractor circuit with quantum gates.	43
6.3	Reduced full adder/subtractor circuit with quantum gates.	43
6.4	Reduced ALU circuit with quantum gates.	43

List of Tables

2.1	Truth table of a 2-input OR gate	7
2.2	Truth table of a NOT gate	8
2.3	Classical reversible function	8
2.4	Quantum gate symbols and unitary matrices [1].	11
3.1	Comparison of existing reversible adder/subtractor circuits.	18
3.2	Related works on reversible ALUs.	18
4.1	Incompletely-specified function ALU.	21
4.2	Four operations of ALU.	22
4.3	OA for ALU.	23
5.1	OA for adder/subtractor.	26
5.2	Incompletely specified half adder/subtractor function hAS_1	28
5.3	Incompletely specified half adder/subtractor function hAS_2	28
5.4	$OPT(PEC(hAS_1))$ and its minimum MCT circuits	29
5.5	$OPT(PEC(hAS_2))$ and its minimum MCT circuits	29
5.6	Incompletely specified full adder/subtractor function fAS_1	30
5.7	Incompletely specified full adder/subtractor function fAS_2	31
5.8	$OPT(PEC(fAS_1))$ and its minimum MCT circuits.	32
5.9	$OPT(PEC(fAS_2))$ and its minimum MCT circuits.	33
5.10	Computer specifications.	36
5.11	Summary of all generated reversible functions for adder/subtractors.	36
5.12	Computation time [second] of experiments in Table 5.11.	36
5.13	Comparison of the proposed reversible full adder/subtractor and existing counterparts.	37
5.14	$ OPT(PEC(OAEC(F))) $ of 1-bit ALUs and computation time.	38
5.15	Comparison of 1-bit ALUs.	39
6.1	Computer specifications.	45

6.2	Reduction of QC in the half adder/subtractor hAS_1	45
6.3	Reduction of QC in the full adder/subtractor fAS_1	46
6.4	$ OPT(PEC(F)) $ of adder/subtractor and computation time for QC reduction.	46
6.5	Comparison of the proposed reversible full adder/subtractor and exist- ing counterparts.	47
6.6	Comparison of 1-bit ALUs.	47
A.1	Truth table representation of reversible LU.	54
A.2	Truth table representation of reversible ALU.	54
A.3	Truth table representation of Gupta's-LU.	55
A.4	Truth table representation of Mini-ALU.	56

Chapter 1

Introduction

1.1 Introduction

In the modern age of science and technology, the world of computing is in a transition period. As chips become smaller and faster, the amount of dissipated heat, which is the energy that is entirely wasted, is increasing. According to Rolf Landauer's principle, the thermodynamic cost of a bit of information acquisition or destruction is, at least, $kT \ln 2$ [Joule], where k is the Boltzmann's constant and T is the temperature [2]; at room temperature, the amount of dissipated heat per bit is small (i.e., 2.9×10^{-21} [Joule]). According to Moore's Law, the advancement of technologies is achieved by the exponential downscaling of transistors and the doubling of the number of transistors per area unit every 18 months [3]. However, reducing the transistor's size and placing a high volume of transistors on a chip can lead to severe problems related to stable computing in nanoscale devices.

Logic synthesis is the process of realizing logic functions in terms of primitives (gates). In today's computers, whenever a logical operation is performed part of the information is erased. Logic operations in computing devices with conventional technology are performed using the logic gates AND, OR, and NOT. The AND and OR gates are irreversible in the sense that they destroy information as they give a single output for multiple inputs. To recover the destroyed information, additional computation is needed, which means the system will eventually dissipate heat to generate the destroyed information. Hence, it is essential to find more efficient forms of computation or alternative forms of computation to overcome such problems. One of the outside runners in the race to take the world of logic by storm is reversible computing. Reversible computing is the path to future computing technologies, which all happen to use reversible logic.

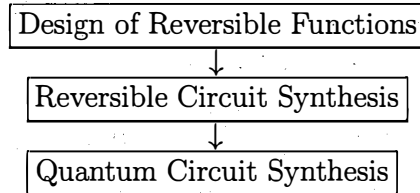


Figure 1.1: Overview of the design flow of reversible synthesis

Synthesis of reversible logic has been an active research area. Reversible logic has the same number of inputs and outputs with a one-to-one mapping between vectors of inputs and outputs; thus, the vector of input states can always be reconstructed from the vector of output states. Consequently, a computation is reversible if it is always possible to uniquely recover the input for a given output. Designs with information loss are not logically reversible [2]. Bennet [4] showed that reversible computation can be potentially performed without losing energy. Systems that are not reversible can be transformed into reversible systems by storing all the input information. This promising approach reduces energy loss. However, many computational problems, such as integer factoring, are exponentially hard to solve.

Recently, several studies have investigated reversible circuit synthesis [5, 6, 7, 8]. Figure 1.1 shows an overview of the design flow of reversible synthesis. The primary goal of designing cost-efficient reversible circuits is to minimize the cost of the circuits in terms of a few metrics such as the number of constant inputs, garbage outputs [9, 10], reversible gates [11], and quantum cost (QC) [12, 13, 14, 15]. Constant inputs are lines that exist on the input side with a certain fixed logical value, either 0 or 1, whereas garbage outputs exist on the output side and do not perform any useful operation to facilitate further computations. Both constant inputs and garbage outputs are simply added to the circuit at the design stage of reversible functions to maintain reversibility.

From published studies, it was observed that multiple-control Toffoli (MCT) gates can be extensively used to synthesize a reversible circuit [16, 17, 18]. Therefore, the number of MCT gates is generally used as the cost metric during the stage of reversible circuit synthesis. A reversible circuit is further decomposed into cascades of elementary quantum gates [19, 20, 21], which is called a quantum circuit. Recently, much attention has been given to the synthesis of quantum circuits because of the potential to exponentially speed-up quantum computation. At the stage of the quantum circuit synthesis, the QC is the most common cost metric, which is measured by counting the number of elementary quantum gates required to implement the quantum circuit.

The fundamental unit of information in quantum computation is the qubit. The computational basis states of a qubit are analogous to the states of a bit in classical computation [22]. According to the principles of quantum mechanics, the logic operation of quantum systems is entirely different from its classical counterpart [22]. The qubit can be described by a state vector in a two-dimensional complex vector space. An infinite state space can be found in this computation by superposition of states [22]. The transition between states of a quantum system is described by a unitary transformation. Unitary operations are reversible and no information is lost in the process. In building a general-purpose quantum computer, it is important to embed Boolean logic functions into quantum gates and devices.

Quantum gates are represented by unitary matrices, which may include complex elements [22], and many sets of gates are universal. The most commonly used quantum gate library includes Toffoli and NCV (NOT, CNOT, Controlled-V, and Controlled- V^\dagger gates). The underlying methods for synthesis include embedding irreversible functions into reversible functions, realizing reversible circuits from reversible functions, and mapping reversible circuits into quantum circuits. Reversible functions are a special case of multiple output Boolean logic functions and can be realized by quantum circuits by cascading quantum gates. The cost factors for quantum circuits are the number of elementary quantum gates used and the number of qubits. Moreover, the QC is the most prevalent criteria to measure the quality of circuits. The resulting decomposed quantum circuits can be further optimized in order to reduce the QC.

The most promising application of reversible logic is quantum computation. Quantum operations are all reversible, and every classical reversible circuit may be implemented in quantum technology. Researchers have used quantum computation to solve many practically relevant problems faster than traditional computing machines. For example, the factorization problem in polynomial time can be solved by quantum computation, whereas, only exponential algorithms are known for traditional machines. Even though research in this area is still on its early stages, promising applications to future computing devices motivate further research.

Reversible computing is considered as an unconventional form of computing, but it has recently drawn considerable attention from researchers in order to design low-power computing devices. The arithmetic logic unit (ALU) is a key element for any programmable computing devices. In the design of ALUs, an adder/subtractor block is an important key element. A faster adder/subtractor block helps improve the efficiency of the ALUs performance and that of the whole system. In this thesis,

we focus on logic synthesis and optimization of reversible and quantum circuits for ALUs by putting emphasis on the function design, which makes our approach different from those of existing related studies. Our approach to investigate reversible functions includes (a) the embedding of irreversible functions into incompletely-specified reversible functions, (b) the operation assignment (OA), and (c) the permutation of function outputs. We also give some extensions of these techniques to further improve the design of reversible functions. The objective of this research is to design ALUs or adder/subtractor blocks that have better performance and lower power consumption than existing systems. Therefore, it is important to have fast reversible ALUs and adder/subtractor blocks because their performance could affect the efficiency of the whole system.

Over the last decades, Toffoli, Peres, and Fredkin are conventionally used to synthesize reversible circuits. Here, we adopt the MCT gate library to design reversible adder/subtractor and simple ALU circuits. As far as we know, there are no existing works on the optimization of reversible full adder/subtractor circuits using only the MCT gate library with the lowest possible number of working lines, constant inputs, and garbage outputs though the MCT gate library is the most fundamental and widely-used gate library for the synthesis of reversible circuits. The major contributions of this work are as follows:

1. Embedding of irreversible functions into incompletely-specified reversible functions. The functions of the adder/subtractor and ALU are irreversible. To obtain a reversible circuit of an irreversible function, we embed the irreversible function into an incompletely-specified function to make it reversible. We set a restriction in which only the minimum necessary ancilla lines are added for embedding. Ancilla lines refer to when additional input Boolean variables are needed to construct the output function. Minimization of the number of reversible gates, ancilla lines, and garbage outputs is one of the major goals in reversible logic design and synthesis.
2. The operation assignment: This is the unique and interesting point is our idea of improvement of reversible circuits. We introduce a new approach called synthesis with OA. It is a permutation of groups of rows in a truth table. Even if the set of operations is the same, the functions consisting of the operations vary according to the assignment of the operations. Our approach is to try all permutations of operations and find the minimum circuit realization.

3. Permutation of function outputs. The idea of permutation of outputs of a reversible function was originally introduced by Wille et al. [23], called as Synthesis with Output Permutation (SWOP). SWOP can be easily applied by encoding all output permutations, synthesizing each in one turn, and keeping the best one.
4. Minimization algorithm. We propose a minimization algorithm to obtain the minimum MCT circuits. It consists of the OA and the permutation of function outputs to further reduce the number of gates in a circuit. We use the hash tables of minimum MCT circuits with up to seven gates for 4-bit functions and five gates for 5-bit functions. The tables are constructed by enumerating all possible gate combinations.
5. Quantum circuit synthesis. We obtain quantum circuits of our reversible adder/subtractors and ALUs by applying a greedy algorithm. The algorithm performs the transformation of a given MCT circuit into a quantum circuit, searches a pair of adjacent gates in the circuit by moving the gate according to the moving rule, and applying different rules to obtain the reduced quantum circuit. The execution of moving and reducing gate is repeated if the quantum gates in the quantum circuit are reduced successively.

This thesis is organized as follows.

- In Chapter 2, we outline some preliminaries of reversible and quantum circuits. We also define irreversible and reversible logic functions. The fundamentals of quantum computation will help readers to understand logic representations in quantum computations.
- In Chapter 3, we analyze and identify problems of previous synthesis and optimization approaches.
- In Chapter 4, we propose design techniques specific to reversible functions of adder/subtractor blocks and ALUs.
- In Chapter 5, we discuss the proposed minimization algorithm for MCT circuits and obtain the minimum MCT circuits of adder/subtractor and ALUs. We also discuss the experimental results of minimum MCT circuits as compared with existing counterparts.

- In Chapter 6, we describe the transformation of MCT circuits into quantum circuits with the NCV gate library and the reduction of the quantum circuits with a simple greedy algorithm. We also show how the experimental results demonstrate the superiority of our realization of adder/subtractor blocks and ALUs with respect to the QC.
- In Chapter 7, we summarize our contributions, culminating with suggestions for future works.

Chapter 2

Basic Definitions

This chapter provides basic definitions and notations to keep this thesis self-contained. In this chapter, some important factors of Boolean logic functions, reversible computing, and quantum computation are explained.

2.1 Boolean Logic Functions

A logic function can be a top-level design entity, which is a logic function that is the root of a design hierarchy, or a lower-level logic function. In binary relation from the set A to B , a function is denoted by

$$f : A \rightarrow B,$$

where for each element $a \in A$ there exists a unique element $b \in B$. The relation may be many-to-one or one-to-one. A k -input Boolean logic function

$$f : B^k \rightarrow B,$$

defined over the set $B = \{0, 1\}$, represents certain propositions whose results would either be 1 (true) or 0 (false).

Table 2.1: Truth table of a 2-input OR gate

Input	Output
00	0
01	1
10	1
11	1

Logic synthesis is the process of realizing logic functions in terms of primitives (gates). Logic operations in computing devices with conventional technology are

Table 2.2: Truth table of a NOT gate

Input	Output
0	1
1	0

performed by using the logic gates AND, OR, and NOT. Table 2.1 shows the logic function of a 2-input OR gate. The AND and OR gates are not reversible because they destroy information whenever as they provide a single output for two inputs. On the other hand, the logic function of the NOT gate has a one-to-one correspondence between input and output. That is, no information is lost in the logic operation of the NOT gate, and hence, it is reversible. An example of a reversible function is shown in Table 2.2.

2.2 Reversible Logic Functions

The main object in reversible logic theory is the reversible function, defined as follows [5, 9].

Definition 1 *The function $f(x_1, x_2, \dots, x_n)$ of n Boolean variables is called reversible if:*

1. *the number of outputs is equal to the number of inputs.*
2. *any input pattern maps to a unique output pattern.*

In the literature, reversible functions are also referred to as classical reversible functions [22]. Table 2.3 shows a classical reversible function.

Table 2.3: Classical reversible function

Input	Output
00	00
01	01
10	11
11	10

In other words, the output of a reversible function is a permutation of the set of its input [5], [24]. For an (n, k) function, i.e., a function with n -inputs k -outputs, it is necessary to add inputs and/or outputs to make it reversible. This leads to the following definition.

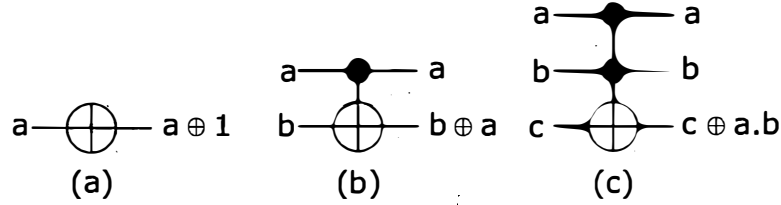


Figure 2.1: MCT gate library: (a) NOT, (b) CNOT, and (c) Toffoli.

Definition 2 “Garbage” is the number of outputs added to make an (n, k) function reversible. While the word “constant inputs” is used to denote the preset value inputs that were added to an (n, k) function to make it reversible. The constant inputs are known as ancilla inputs.

The relation between garbage outputs and constant inputs is as follows [5], [24].

$$\# \text{ input} + \# \text{ constant input} = \# \text{ output} + \# \text{ garbage}$$

2.3 Reversible Gates

Several reversible gates have been proposed in the last few decades, from which the Toffoli, Peres, Feynman, and Fredkin are conventionally used to synthesize reversible circuits. Recently, researchers have proposed different gate libraries such as MCT; multiple-control Fredkin (MCF); Peres (P); NOT, CNOT, and Toffoli (NCT); multiple-control Toffoli and Peres (MCT+P); multiple-control Toffoli and multiple-control Fredkin (MCT+MCF). Among them, the MCT gate library is the most widely used for reversible logic synthesis. Thus, we use it to design a reversible circuit.

Definition 3 A multiple-control Toffoli (MCT) gate has $(k-1)$ control lines $\{x_1, x_2, \dots, x_{k-1}\}$ and one target line x_k , whose function is a mapping from $(x_1, x_2, \dots, x_{k-1}, x_k)$ to $(x_1, x_2, \dots, x_{k-1}, (x_1 x_2 \cdots x_{k-1}) \oplus x_k)$. For the case $k = 1$, the gate maps x_1 to $1 \oplus x_1$.

Here, an MCT gate with $(k-1)$ controls is denoted by Toffoli- k . Toffoli-1 and Toffoli-2 are also called NOT and Controlled-NOT (CNOT), respectively, whereas Toffoli-3 is the original Toffoli gate. The schematics of MCT gates are shown in Fig. 2.1. The control lines are denoted by black dots (\bullet), whereas the target line is denoted by \oplus .

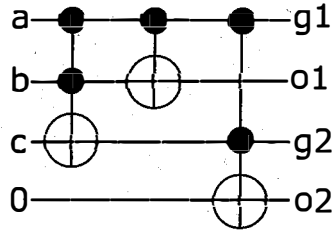


Figure 2.2: Reversible circuit.

2.4 Reversible Circuits

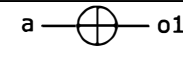
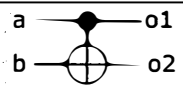
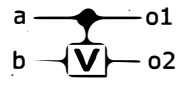
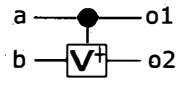
A reversible function can be realized by cascading reversible gates where fan-out and feedback are not directly allowed in the realization of a reversible circuit. In reversible circuits, each variable of the function is represented using a circuit line. A Boolean function that is not reversible can be transformed into a reversible function by adding extra-working lines to ensure reversibility. The extra inputs in a reversible function can be preset to a constant value, which can either be 0 or 1. The extra outputs are referred to as garbage outputs. Figure 2.2 shows a reversible circuit with three reversible gates, one constant input (0), two garbage outputs (g_1, g_2), and two outputs (o_1, o_2). Minimum numbers of reversible gates, constant inputs, and garbage outputs are the properties of a good quality of reversible circuit. In this work, we focus on minimizing the MCT gate count in reversible circuits.

2.5 Quantum Computation

The logic representation in quantum computation is quite different from that in classical computation. The basic unit of information in quantum computation is a qubit represented by a state vector. The states $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ or $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ are known as the computational basis states. The state of an arbitrary qubit $\alpha|0\rangle + \beta|1\rangle$ is described by the vector $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, where α and β are complex numbers that satisfy the constraint $|\alpha|^2 + |\beta|^2 = 1$. The measurement of a qubit can either be 0 with probability $|\alpha|^2$ or 1 with probability $|\beta|^2$. Similarly, a generalized 2-qubit state can be described [1] as

$$|\Psi\rangle = \lambda_1 |00\rangle + \lambda_2 |01\rangle + \lambda_3 |10\rangle + \lambda_4 |11\rangle = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{pmatrix},$$

Table 2.4: Quantum gate symbols and unitary matrices [1].

Gate Name	Gate Symbols	Matrix
NOT		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
CNOT		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
Controlled- V		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{(1+i)}{2} & \frac{(1-i)}{2} \\ 0 & 0 & \frac{(1-i)}{2} & \frac{(1+i)}{2} \end{pmatrix}$
Controlled- V^\dagger		$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{(1-i)}{2} & \frac{(1+i)}{2} \\ 0 & 0 & \frac{(1+i)}{2} & \frac{(1-i)}{2} \end{pmatrix}$

where $\lambda_1\lambda_4 = \lambda_2\lambda_3$ for non-entanglement.

On the contrary, a classical bit has a state that can either be 0 or 1, which is analogous to the measurement of a qubit state of $|0\rangle$ or $|1\rangle$ respectively. The main difference between bits and qubits is that the former can be in the state 0 or 1, whereas the latter can be in a superposition of $|0\rangle$ and $|1\rangle$.

2.6 Quantum Gates

Quantum gates are the building blocks of quantum circuits, similar to classical logic gates, e.g., AND, OR, and NOT, being the building blocks of conventional digital circuits. Several quantum gates have been defined and studied, but we concentrate on the elementary quantum gates NOT, CNOT, Controlled- V , and Controlled- V^\dagger , which are also known as quantum primitives. This set of gates is known as the NCV gate library. These gates have been widely used in the synthesis of binary reversible functions. Elementary gates are represented by unitary matrices. A gate which acts on k qubits is represented by a $2^k \times 2^k$ unitary matrix [22], which may include complex elements, as shown in Table 2.4.

The 2-line Controlled- V gate changes the target line using the transformation defined by the matrix $V = \frac{1+i}{2} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$ if the single control line has a value of 1. The 2-line Controlled- V^\dagger gate changes the target line using the transformation

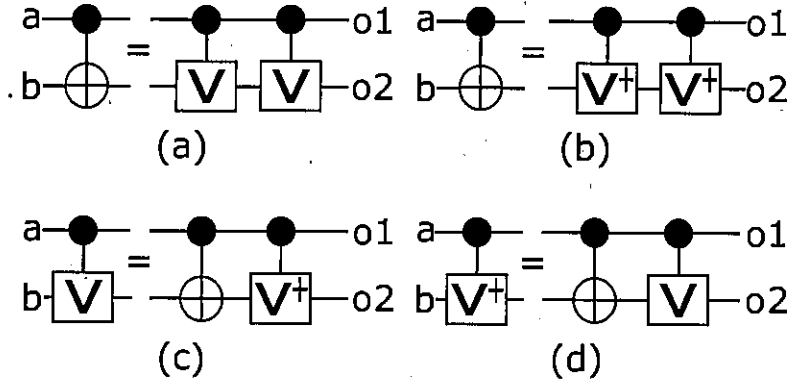


Figure 2.3: Splitting and merging rules

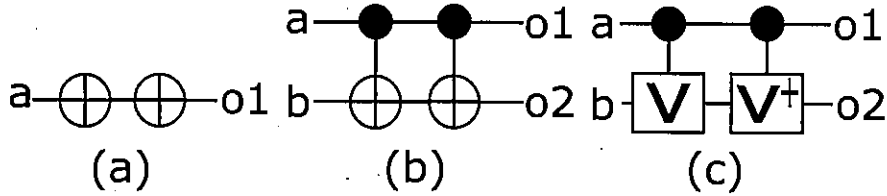


Figure 2.4: Deletion rule.

defined by the matrix $V^\dagger = V^{-1} = \frac{1-i}{2} \begin{pmatrix} 1 & i \\ i & 1 \end{pmatrix}$ if the single control line has a value of 1. The gates V and V^\dagger are referred to as square-root-of-NOT gates because $V^2 = (V^\dagger)^2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$.

The Controlled- V and Controlled- V^\dagger gates are inverses of each other, whereas NOT and CNOT (generally MCT gates) are self-inverse gates. Two adjacent Controlled- V (or Controlled- V^\dagger) gates with the same target and control can be replaced by a CNOT. A primitive from CNOT, Controlled- V , and Controlled- V^\dagger can be represented by a pair of other primitives, as shown in Fig. 2.3; this is referred to as the splitting rule. The inverse of the splitting rule is the merging rule.

If two adjacent gates form the identity function, then they can be deleted, this known as the deletion rule. Therefore, two NOT gates, two CNOT gates, and an adjacent (Controlled- V , Controlled- V^\dagger) pair (any order) with the same target and control can be eliminated, as depicted in Fig. 2.4.

The mobility of gates is determined using the following property [11], which is called the moving rule.

Property 1 (Moving rule) *Two adjacent gates g_1 and g_2 with controls c_1 and c_2 and targets t_1 and t_2 , respectively, can be interchanged if $t_2 \notin c_1$ and $t_1 \notin c_2$.*

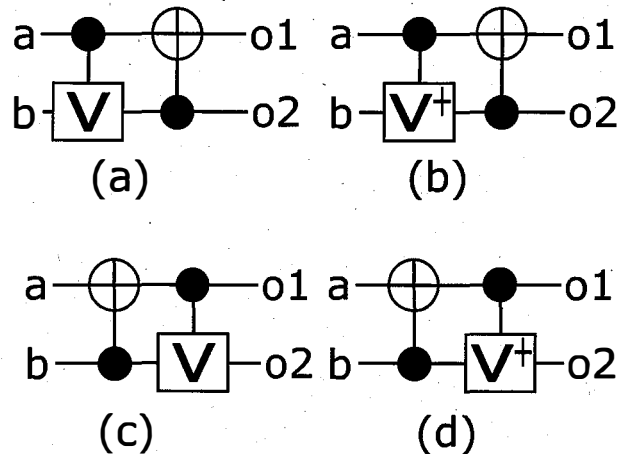


Figure 2.5: Merged 2-qubit gate.

Example 1 *An elementary quantum gate has zero or one control and one target. Only a NOT gate has no control. In Fig. 2.7 (a), gates g_1 and g_2 with controls $\{a\}$ and $\{b\}$ and targets c and c , respectively, can be interchanged because $c \notin \{a\}$ and $c \notin \{b\}$. Subsequently, gates g_1 and g_3 with controls $\{a\}$ and $\{a\}$ and targets c and b , respectively, can be interchanged because they satisfy the same condition. Thus, the gate g_1 can be moved to any other location in the circuit.*

The deletion and merging rules reduce the quantum gates in a circuit, whereas the moving rule enhances the applicability of the deletion and merging rules.

Hung et al. [25] showed that when both CNOT and Controlled-V (or Controlled- V^\dagger) are operating on the same two qubits in a symmetric pattern as shown in Fig. 2.5, their total QC is considered to be one as well. We call this the 2-qubit gate rule in the QC metrics.

The main concept of Hung et al. [25] came from the work by J. Smolin and D. DiVincenzo [26]. They considered the merged 2-qubit gate as another type of fundamental 2×2 reversible logic gate referred to as the integrated qubit gate [26]. This gate is implemented with a Feynman gate with either a Controlled-V or Controlled V^\dagger gate. The QC of the integrated qubit gate is 1; and its worst-case delay is 1. They also presented an analytic construction of 3-bit quantum Fredkin gate that uses only five quantum gates, each acting on only two qubits. It is realized using two Feynman gates, a Controlled-V gate, and two integrated qubit gates. The quantum representation of the Fredkin gate is shown in Fig. 2.6. The second and third gates in Fig. 2.6 are each acting on the same two bits, and therefore can be replaced by a

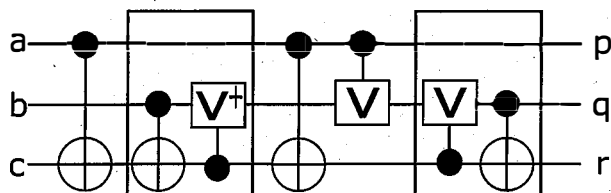


Figure 2.6: Quantum gate representation of the Fredkin Gate.

single 2-bit gate. Similarly, the last two gates are two adjacent gates acting on the same two bits. By merging these two gates, a 5-gate design is produced.

2.7 Quantum Circuits

A quantum circuit is a model for quantum computation and can be realized using cascading quantum gates. Quantum operations are all reversible, and every classical reversible circuit may be implemented in quantum technology. The number of elementary quantum gates required to implement a reversible circuit is the QC of a reversible circuit. The QC is an important parameter to determine the quality of quantum and reversible circuits. Note that a reversible circuit may have multiple quantum circuit realizations.

For example, Figure 2.7 shows four realizations of the Toffoli-3 gate. Figure 2.7 (b) is the reverse representation of Figure 2.7 (a). Figures 2.7 (c) and (d) are obtained by interchanging the Controlled-V and Controlled-V[†] in Figures 2.7 (a) and (b), respectively. The QC of the circuits in Figure 2.7 is 5 as they each consists of five quantum gates.

However, during the transformation of MCT circuits into quantum circuits, the QC of MCT circuits may differ according to the selection of the quantum gate realization of Toffoli gates.

For example, consider the MCT circuit in Figure 2.2. The circuit has two Toffoli-3 gates. If the two Toffoli gates are replaced by the realization in Figure 2.7 (a), the QC of the circuit becomes 9. However, if the first Toffoli gate is replaced by the realization in Figure 2.7 (a) and if the second gate is replaced by the realization in Figure 2.7 (b), the QC becomes 8. The selection of an appropriate realization of the Toffoli gate does not affect the functionality of the circuits; however, it may provide better quality circuits in terms of the quantum gate count. This concept is used in the reduction of the QC in Chapter 6.

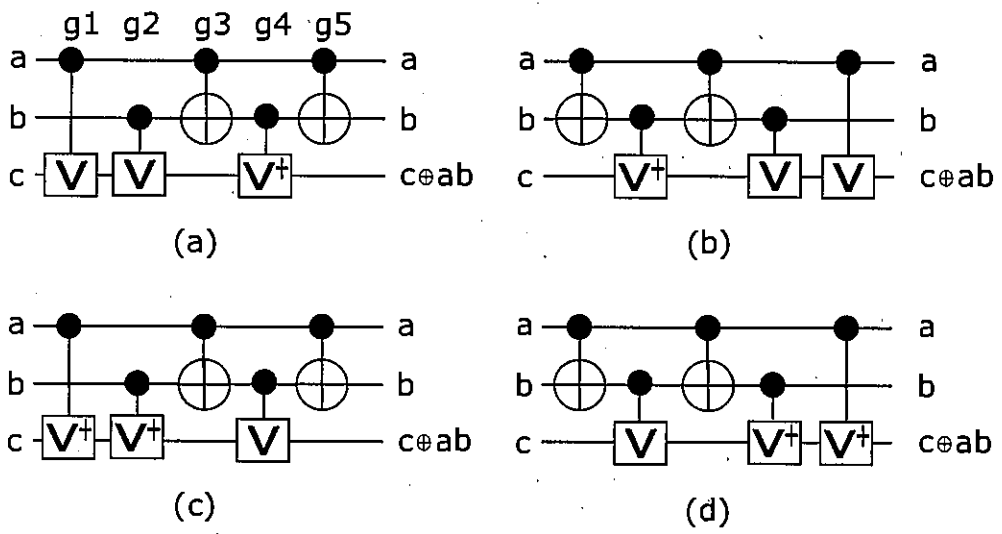


Figure 2.7: Quantum circuit.

Chapter 3

Literature Review

This chapter provides previous synthesis and optimization approaches used to design the ALUs and adder/subtractor block. We know that the logic unit is a key element in programmable computing devices and an adder/subtractor block is also an important element in designing ALU. A faster adder/subtractor block will increase the efficiency of the ALUs performance and that of the whole system. Here, the objective is to design ALUs or adder/subtractor blocks that have better performance and lower power consumption than existing systems. In this chapter, we discuss several designs of ALUs and adder/subtractor blocks based on reversible logic.

3.1 Related Works

Rangaraju et al. [27] proposed three designs for reversible half and full adder/subtractor circuits using a different gate library that includes Feynman [28], Fredkin, TR, and Peres gates. In some of the cases, their designs require five or more working lines, a large number of reversible gates, constant inputs, and garbage outputs with a high QC. Moghimi et al. [29] proposed a new 4x4 universal reversible gate as a cost-efficient full adder/subtractor in terms of reversible and quantum metrics. They have shown improvements over existing full adder/subtractor designs by comparing the number of reversible gates, garbage outputs, constant inputs, and the QC. Sultan et al. [30] proposed a full adder/subtractor circuit using Feynman and Peres gates, showing improvements over the previous work [27] in terms of the number of reversible gates, garbage outputs, constant inputs, and the QC. Table 3.1 shows a summary of related works on reversible adder/subtractor circuits.

Gupta et al. [33] proposed a reversible LU with eight operations using five inputs. The Mini-ALU reported in Revlib [34] can perform four operations, i.e., OR, AND,

Table 3.1: Comparison of existing reversible adder/subtractor circuits.

Design	Reversible and quantum metrics			
	Reversible Gates	Constant Inputs	Garbage Outputs	QC
Design 1 [27]	8	3	5	21
Design 2 [27]	4	1	3	14
Design 3 [27]	4	1	3	10
Design 1 [31]	8	5	7	28
Design 2 [31]	10	5	8	24
Design [32]	3	2	4	18
Design [29]	1	0	2	11
Design [30]	4	3	1	10

ADD, and Identity. Both ALUs were implemented using the MCT gate library. Table 3.2 shows a summary of related works on reversible ALUs.

Table 3.2: Related works on reversible ALUs.

	Reversible Gates	QC
Design [33]	18	114
Design [34]	6	62

As seen in Figure 1.1, the design flow of reversible synthesis comprises three stages. We found that related works for reversible adder/subtractors and ALUs have limited the analysis of the level of reversible function design. Here, we investigate the function design thoroughly to improve reversible circuits. The minimization problem for MCT circuits is also extended accordingly. As a result, we obtain the minimum MCT circuits of the adder/subtractors and the ALUs. We also provide the quantum circuits of our reversible adder/subtractors. There were very few works that proposed the quantum circuit implementation for the adder/subtractors, while obtaining the quantum circuit is the goal of the reversible synthesis.

In this thesis, we adopt the MCT gate library to design reversible adder/subtractor and simple ALU circuits. As far as we know, there are no existing works on the optimization of reversible full adder/subtractor circuits using only the MCT gate library with the lowest possible number of working lines, constant inputs, and garbage outputs even though the MCT gate library is the most fundamental and widely-used gate library for the synthesis of reversible circuits.

Chapter 4

Design of Reversible Functions

In the process of reversible logic synthesis, the design of reversible functions is very important and the embedding of irreversible functions into reversible functions is required before they can be applied to existing synthesis methods. Irreversible functions can be embedded into incompletely-specified or completely specified reversible functions [23] depending on which synthesis methods used. In this chapter, we embed irreversible functions into incompletely-specified reversible functions because the synthesis method we used supports it. Note that many syntheses approaches require a completely specified function so that often all don't care must be assigned to a concrete value.

This chapter also discusses the unique and interesting method to improve reversible circuits. The whole process of reversible circuit synthesis is different from that of related works; we try improvements on the level of the functional design. Our approach to investigate the reversible functions includes the following:

- Embedding irreversible functions into incompletely-specified reversible functions
- OA,
- Permutation of the function outputs

Moreover, we provide some extensions of these techniques to deal with a set of incompletely-specified reversible functions.

4.1 Embedding Irreversible Functions into Incompletely-Specified Reversible Functions

The functions of the adder/subtractor and ALU are irreversible. To obtain a reversible circuit of an irreversible function, we embed the irreversible function into an

incompletely-specified function to make it reversible. Here, we set a restriction in which only the minimum necessary ancilla lines are added for embedding. Ancilla lines refer to when additional input Boolean variables are needed to construct the output function. Consider the functional specification of ALU in Table 4.1, where ‘-’ denotes the don’t-care value. We added two garbage outputs to the irreversible function to embed it into a reversible function. The garbage outputs are the columns of the don’t-care values in the truth table. Therefore, this function is called an incompletely-specified reversible function. The required garbage outputs depend on the maximum number of repetition of an output pattern in the truth table. If the number of repetitions is M , $\lceil \log_2 M \rceil$ garbage outputs are necessary [10]. In Table 4.1, the output patterns (0) and (1) are repeated four times for the input $\{0100, 1000, 1001, 1010\}$ and $\{0101, 0110, 0111, 1011\}$, respectively, which is the maximum number of repetitions. When $\lceil \log_2 4 \rceil = 2$, two garbage outputs are added. However, no additional lines are required in this reversible circuit because the number of outputs is equal to the number of inputs.

Generally, an irreversible function can be embedded into a reversible function by adding necessary garbage outputs; the resulting reversible function is incompletely specified. To discuss incompletely-specified functions, we provide some necessary definitions: Hereafter, ‘a reversible function’ indicates a fully-specified reversible function unless otherwise noted.

Definition 4 *An incompletely-specified reversible function is abbreviated as ISRF. Let F be a (fully-specified) reversible function. We say that F matches an ISRF F' if $F(\mathbf{X}) = F'(\mathbf{X})$ for all inputs $\mathbf{X} \in \{0, 1\}^n$ except when $F'(\mathbf{X}) = \text{'-'}$, where ‘-’ is the don’t-care value. $F \in: F'$ denotes that F matches F' . As an extension, for a set of ISRFs \mathcal{F} , $F \in: \mathcal{F}$ denotes that $F \in: F'$ holds for some ISRF F' in \mathcal{F} . We also say that F matches \mathcal{F} if $F \in: \mathcal{F}$.*

The minimality, in this thesis, is defined by the gate count of MCT circuits. Below, an extension of the minimality to a set of ISRFs is given.

Definition 5 *Let F be a reversible function of n variables. Among all n -bit MCT circuits that realize F , those with the exact minimum number of MCT gates are called the minimum MCT circuits of F . The size of F is defined by the gate count of the minimum MCT circuit of F , which is denoted by $\gamma(F)$. We extend γ to that for a set of ISRFs \mathcal{F} ; $\gamma(\mathcal{F})$ is defined by the minimum $\gamma(F)$ for all $F \in: \mathcal{F}$, namely, $\gamma(\mathcal{F}) = \min\{\gamma(F) \mid F \in: \mathcal{F}\}$. $OPT(\mathcal{F})$ is the set of reversible functions $F \in: \mathcal{F}$ with the minimum size $\gamma(\mathcal{F})$; $OPT(\mathcal{F}) = \{F \mid F \in: \mathcal{F}, \gamma(F) = \gamma(\mathcal{F})\}$.*

Table 4.1: Incompletely-specified function ALU.

S_1	S_2	A	B	g_1	g_2	O_1	O_2
0	0	0	0	-	-	0	0
0	0	0	1	-	-	0	1
0	0	1	0	-	-	0	1
0	0	1	1	-	-	1	0
0	1	0	0	-	-	-	0
0	1	0	1	-	-	-	1
0	1	1	0	-	-	-	1
0	1	1	1	-	-	-	1
1	0	0	0	-	-	-	0
1	0	0	1	-	-	-	0
1	0	1	0	-	-	-	0
1	0	1	1	-	-	-	1
1	1	0	0	-	-	0	0
1	1	0	1	-	-	1	1
1	1	1	0	-	-	0	1
1	1	1	1	-	-	0	0

4.2 Operation Assignment (OA)

In this thesis, we introduce a new approach called synthesis with OA. It is a permutation of groups of rows in the truth table. This approach is applicable to circuits that have more than one operations such as adder/subtractor and ALU. This type of circuit has selector bits to choose the desired operation. Even if the set of operations is the same, the functions with those operations vary according to the assignment of the selector bits to the operations. Our approach is to try all permutations of operations and find the minimum circuit realization. If a function has m selector bits, then $2^m!$ variants are considered. To utilize the concept of the permutation of operations, we give a formal definition of an equivalence relation to OA.

Definition 6 Suppose that a multiple-output function F has m selector bits $\{S_1, S_2, \dots, S_m\}$ and is represented by $F(S_1, S_2, \dots, S_m, \mathbf{X})$. F may be an ISRF. For a non-negative integer i ($0 \leq i \leq 2^m - 1$), $F_i(\mathbf{X})$ denotes the cofactor $F(i_1, i_2, \dots, i_m, \mathbf{X})$, where (i_1, i_2, \dots, i_m) is an m -bit binary representation of i . The cofactors $F_i(\mathbf{X})$ are called operations of $F(S_1, S_2, \dots, S_m, \mathbf{X})$. The literals \bar{S} and S are denoted by S^0 and S^1 , respectively. By using these notations, the Shannon expansion of F with respect to variables S_1, S_2, \dots, S_m is represented by $F(S_1, S_2, \dots, S_m, \mathbf{X}) = \bigvee_{0 \leq i \leq 2^m - 1} S_1^{i_1} S_2^{i_2} \dots S_m^{i_m} \cdot F_i(\mathbf{X})$. We say that a function defined by $\bigvee_{0 \leq i \leq 2^m - 1} S_1^{i_1} S_2^{i_2} \dots S_m^{i_m} \cdot F_{\pi(i)}(\mathbf{X})$ is OA-equivalent to $F(S_1, S_2, \dots, S_m, \mathbf{X})$ if π is a permutation of $\{0, 1, \dots, 2^m - 1\}$.

Table 4.2: Four operations of ALU.

		A	B	g_1	g_2	O_1	O_2
ADD: $F_0(A, B) =$		0	0	-	-	0	0
		0	1	-	-	0	1
		1	0	-	-	0	1
		1	1	-	-	1	0
		A	B	g_1	g_2	O_1	O_2
OR: $F_1(A, B) =$		0	0	-	-	-	0
		0	1	-	-	-	1
		1	0	-	-	-	1
		1	1	-	-	-	1
		A	B	g_1	g_2	O_1	O_2
AND: $F_2(A, B) =$		0	0	-	-	-	0
		0	1	-	-	-	0
		1	0	-	-	-	0
		1	1	-	-	-	1
		A	B	g_1	g_2	O_1	O_2
SUB: $F_3(A, B) =$		0	0	-	-	0	0
		0	1	-	-	1	1
		1	0	-	-	0	1
		1	1	-	-	0	0

The set of all functions OA-equivalent to F is called the OA-equivalence class of F and denoted by $OAEC(F)$, in which the selector bits $\{S_1, S_2, \dots, S_m\}$ of F is assumed to be specified in the definition of F .

Note that m selector bits can identify up to 2^m operations. Therefore, for a function with m selector bits, there exist at most $2^m!$ OA-equivalent functions by the permutation of 2^m operations.

Property 2 For a function F with m selector bits, $|OAEC(F)| \leq 2^m!$.

For a function F and a circuit C , it is commonly said that C realizes F if the function of C is equal to F . We also extend the concept of circuit realization from a reversible function to a set of ISRFs.

Definition 7 Let C be a reversible circuit and \mathcal{F} be a set of ISRFs. We say that C realizes \mathcal{F} if the function of C matches \mathcal{F} .

S_1	S_2	ALU	ALU'
0	0	ADD	OR
0	1	OR	ADD
1	0	AND	SUB
1	1	SUB	AND

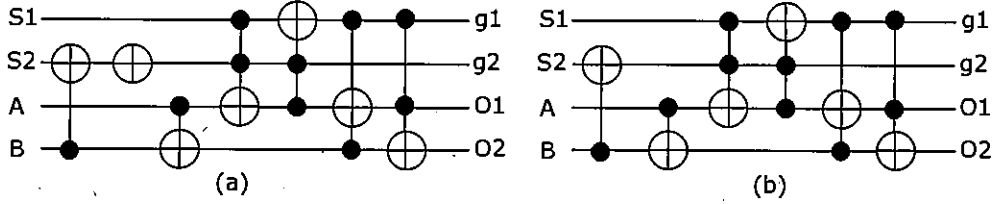


Figure 4.1: (a) Circuit realizing ALU. (b) Circuit realizing $OAEC(ALU)$.

Example 2 Consider the function of an ALU shown in Table 4.1, in which S_1 and S_2 are the selector bits. Table 4.2 shows the four operations F_0, \dots, F_3 of the ALU. With these operations, the ALU is represented by $\bar{S}_1\bar{S}_2F_0(\mathbf{X}) \vee \bar{S}_1S_2F_1(\mathbf{X}) \vee S_1\bar{S}_2F_2(\mathbf{X}) \vee S_1S_2F_3(\mathbf{X})$. For example, the function $ALU' = \bar{S}_1\bar{S}_2F_1(\mathbf{X}) \vee \bar{S}_1S_2F_0(\mathbf{X}) \vee S_1\bar{S}_2F_3(\mathbf{X}) \vee S_1S_2F_2(\mathbf{X})$ made by a permutation of operations is OA-equivalent to ALU. The difference between ALU and ALU' is the assignment of operations, which is summarized in Table 4.3. A function that is OA-equivalent to ALU provides the same set of operations as ALU. However, the cost related to their circuit realizations are generally different. A minimum MCT circuit of ALU with the original assignment in Table 4.3 is shown in Figure 4.1 (a) using seven reversible gates (one NOT gate, two CNOT, and four Toffoli-3 gates) with a QC of 23. If we use a different assignment of operations as ALU', a different circuit realization using six gates (two CNOT and four Toffoli-3 gates) with a QC of 22 is obtained, as shown in Figure 4.1 (b). This example demonstrates that permutation of operations directly affects on the cost of reversible circuit realization.

4.3 Permutation of Function Outputs

The idea of permutation of outputs of a reversible function was originally introduced by Wille et al. [23], called as Synthesis with Output Permutation (SWOP). The following definition is the ISRF version of SWOP and its extension to a set of ISRFs.

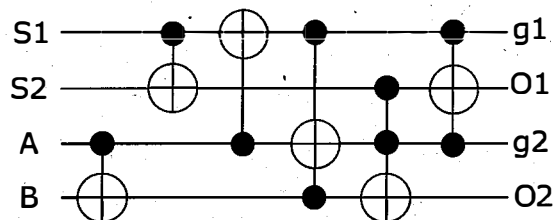


Figure 4.2: Circuit realizing $PEC(ALU)$.

Definition 8 For a given ISRF F , the set of variants made by the permutation of outputs in F is called the P-equivalence class of F denoted by $PEC(F)$. We extend PEC so that it accepts a set of ISRFs \mathcal{F} : $PEC(\mathcal{F}) = \bigcup_{F \in \mathcal{F}} PEC(F)$.

P-equivalence is a permutation of columns in the truth table, whereas OA-equivalence is a permutation of groups of rows in the truth table.

$|PEC(F)|$ for an n -output ISRF F is at maximum $n!$ because $PEC(F)$ is a permutation of the outputs of F .

Property 3 For an n -output ISRF F , $|PEC(F)| \leq n!$.

Note that the P-equivalence class of a reversible function F is different from the conjugacy class [35] of F (also called the line reordering). In the conjugacy class, the inputs and outputs in F are relabeled simultaneously. In the P-equivalence class, the outputs in F are relabeled, but the inputs are not altered. It is known that every function in the conjugacy class of F can be realized in a circuit with the same gate count. However, the gate count of the circuit of a function in the P-equivalence class of F is generally not the same as that of F (e.g., Example 3). This means that there may be a smaller function in the gate count than F among functions matching $PEC(F)$.

Example 3 Consider that the ALU shown in Table 4.1 maps the input (S_1, S_2, A, B) to the output (g_1, g_2, O_1, O_2) . The minimum MCT circuit shown in Fig. 4.1 (a) consists of seven gates. Figure 4.2 shows an MCT circuit realizing $PEC(ALU)$, in which the four outputs of the function are reordered to another position. More precisely, the MCT circuit shown in Fig. 4.2 maps the input (S_1, S_2, A, B) to the output (g_1, O_1, g_2, O_2) . This reduces the overall number of gates from seven to six.

Chapter 5

Synthesis of Minimum Reversible Circuits

The two techniques, *OAEC* and *PEC*, discussed in Chapter 4 can be combined to further reduce the number of gates in a circuit. In this chapter, we discuss the minimization of MCT circuits realizing $PEC(OAEC(F))$ for a given F . By Definition 8, *PEC* is extended to accept a set of ISRFs like $OAEC(F)$.

5.1 Minimization Algorithm

The adder/subtractor and ALUs proposed in this sections are 4-bit or 5-bit functions. The minimum MCT circuits for all the 4-bit reversible functions have been obtained by Golubitsky et al. [35]. Their algorithm, however, does not support ISRFs. An SAT-based algorithm [16] can be applied to ISRFs. The algorithm produces one minimum circuit for a given ISRF; however, it does not provide a list of minimum circuits for other reversible functions that match the ISRF. Thus, instead of using these sophisticated algorithms, we use the hash tables of minimum MCT circuits with up to seven gates for 4-bit functions and five gates for 5-bit functions. The tables are constructed by enumerating all possible gate combinations. The procedure `MAKE_TABLE` in Fig. 5.1 is used to construct such tables. Although the applicability of this simple strategy is limited to small circuits because of high memory consumption, it is sufficient to achieve our purpose.

In the algorithm, the number of input/output lines, or bits, are denoted as n , and minimum MCT circuits with i gates are stored in ht_i . The function `FINDOPT` obtains $OPT(\mathcal{F})$ and its minimum MCT circuits in pairs by searching for ht_1, ht_2, \dots successively. A memory saving technique with symmetry [35] is used, however, the description of this technique is omitted from Fig. 5.1. We combine the hash tables

of these minimum circuits and the techniques proposed in Chapter 4, to obtain the minimum MCT realization of the adder/subtractor and ALU circuits.

We also compare the results obtained using FINDOPT with those using the SAT-based algorithm. The results reveal that the list of minimum MCT circuits obtained using FINDOPT has a better circuit in terms of QC as compared to the one obtained using the SAT-based algorithm.

5.2 Reversible Adder/Subtractor

In this section, we obtain the minimum MCT circuits of half and full adder/subtractors using our proposed technique. An adder/subtractor has two operations, i.e., ‘adder’ and ‘subtractor’, and the 1-bit selector S decides the operations to be performed. Possible assignments of operations are shown in Table 5.1. Along with half and full adders, there are half and full adder/subtractors.

Table 5.1: OA for adder/subtractor.

S	Assignment 1	Assignment 2
0	Adder	Subtractor
1	Subtractor	Adder

5.2.1 Half Adder/Subtractor

According to the OA shown in Table 5.1, two OA-equivalent functions of the half adder/subtractor exist for Assignments 1 and 2. Equations (5.1) and (5.2) provide the logical expressions of these functions. ‘ C/B ’ and ‘ S/D ’ denote Carry/Borrow and Sum/Difference, respectively.

$$\begin{cases} C/B = \bar{S}AB \vee S(AB \oplus B) \\ S/D = A \oplus B \end{cases} \quad (5.1)$$

$$\begin{cases} C/B = \bar{S}(AB \oplus B) \vee SAB \\ S/D = A \oplus B \end{cases} \quad (5.2)$$

Equation (5.1) is embedded in the truth table, as in Table 5.2, which is denoted by hAS_1 . During embedding, two garbage outputs, g_1 and g_2 , are added because they are required by the maximum number of repetitions of an output pattern in the truth table. To balance the numbers of inputs and outputs, a constant input $c = 0$ is added. Similarly, Equation (5.2) is embedded in the truth table as in Table 5.3, which is denoted by hAS_2 .

```

1: var  $ht_0, ht_1, ht_2, \dots$  : Hash Table;
2: procedure MAKE_TABLE( $m$ )
3:                                     ▷ Input:  $m$  is an integer.
4:     ▷ Side Effect: minimum MCT circuits with up to  $m$  gates are stored in the
      hash tables  $ht_0, ht_1, ht_2, \dots$ 
5:     if  $m = 0$  then  $ht_0[\textit{identity}] \leftarrow$  the empty circuit;
6:     else
7:         MAKE_TABLE( $m - 1$ );
8:         for all entry  $C \in ht_{m-1}$  do
9:             for all  $G \in \mathcal{G}_n$  do
10:                if  $ht_m[\textit{func}(C | G)] = \emptyset$  then
11:                     $ht_m[\textit{func}(C | G)] \leftarrow C | G$ ;
12:                end if
13:            end for
14:        end for
15:    end if
16: end procedure
17: function FIND_OPT( $\mathcal{F}$ )
18:                                     ▷ Input:  $\mathcal{F}$  is a set of ISRFs.
19:                                     ▷ Output:  $OPT(\mathcal{F})$  and its minimum MCT circuits in pairs.
20: var  $i \leftarrow 1$  : Integer;
21: var  $S \leftarrow \emptyset$  : Set;
22: while ( $S = \emptyset$ ) or ( $ht_i$  is not empty) do
23:     for all key  $F \in ht_i$  do
24:         if  $F \in \mathcal{F}$  then  $S \leftarrow \{(F, ht_i[F])\} \cup S$ ;
25:         end if
26:     end for
27:      $i \leftarrow i + 1$ ;
28: end while
29: return  $S$ ;
30: end function

```

\mathcal{G}_n : set of all MCT gates with up to n bits.

$C | G$: concatenation of circuit C and gate G .

$\textit{func}(C | G)$: reversible function of the circuit $C | G$.

Figure 5.1: Simple minimization algorithm with up to m gates

Table 5.2: Incompletely specified half adder/subtractor function hAS_1 .

c	S	A	B	g_1	g_2	C/B	S/D
0	0	0	0	-	-	0	0
0	0	0	1	-	-	0	1
0	0	1	0	-	-	0	1
0	0	1	1	-	-	1	0
0	1	0	0	-	-	0	0
0	1	0	1	-	-	1	1
0	1	1	0	-	-	0	1
0	1	1	1	-	-	0	0
1	0	0	0	-	-	-	-
1	0	0	1	-	-	-	-
1	0	1	0	-	-	-	-
1	0	1	1	-	-	-	-
1	1	0	0	-	-	-	-
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	-	-	-	-

Table 5.3: Incompletely specified half adder/subtractor function hAS_2 .

c	S	A	B	g_1	g_2	C/B	S/D
0	0	0	0	-	-	0	0
0	0	0	1	-	-	1	1
0	0	1	0	-	-	0	1
0	0	1	1	-	-	0	0
0	1	0	0	-	-	0	0
0	1	0	1	-	-	0	1
0	1	1	0	-	-	0	1
0	1	1	1	-	-	1	0
1	0	0	0	-	-	-	-
1	0	0	1	-	-	-	-
1	0	1	0	-	-	-	-
1	0	1	1	-	-	-	-
1	1	0	0	-	-	-	-
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	-	-	-	-

Table 5.4 shows the minimum MCT realization of the P-equivalence class of hAS_1 , or $OPT(PEC(hAS_1))$, obtained by our FINDOPT program. The six reversible functions have minimum MCT circuits with three gates. On the right-hand side of Table 5.4, a circuit is represented by a sequence of reversible gates, in which $(x y)$ represents a CNOT gate and $(x y z)$ represents a Toffoli gate. The last parameter is the target line, and the remaining parameters are the control lines. For example, “(2 1) (3 2) (1 3 0)” in the first row represents the MCT circuit of Fig. 5.2, where the numbering of the lines is $(c, S, A, B) = (0, 1, 2, 3)$.

Reversible Function	Minimum MCT Circuit
(0 3 6 13 4 15 2 1 8 11 14 5 12 7 10 9)	(2 1) (3 2) (1 3 0)
(0 3 2 13 4 15 6 1 8 11 10 5 12 7 14 9)	(2 3 1) (3 2) (1 3 0)
(0 3 2 9 4 15 6 5 8 11 10 1 12 7 14 13)	(2 3 0) (3 2) (1 3 0)
(0 1 7 14 4 13 3 2 8 9 15 6 12 5 11 10)	(2 1) (1 3 0) (2 3)
(0 1 3 14 4 13 7 2 8 9 11 6 12 5 15 10)	(2 3 1) (1 3 0) (2 3)
(0 1 3 10 4 13 7 6 8 9 11 2 12 5 15 14)	(2 3 0) (1 3 0) (2 3)

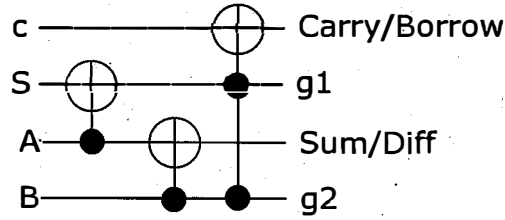


Figure 5.2: Proposed reversible half adder/subtractor circuit with MCT gates.

Reversible Function	Minimum MCT Circuit
(0 15 6 1 4 3 2 13 8 7 14 9 12 11 10 5)	(3 2) (2 1) (1 3 0)
(0 11 2 1 4 7 6 13 8 3 10 9 12 15 14 5)	(1 3 0) (3 2) (2 3 0)
(0 15 2 1 4 3 6 13 8 7 10 9 12 11 14 5)	(3 2) (2 3 1) (1 3 0)

Similar experiments are conducted for hAS_2 . The results, however, are not better than those for hAS_1 . The $OPT(PEC(hAS_2))$ are shown in Table 5.5. Three functions have minimum MCT circuits with three gates.

Since the possible OAs for the half adder/subtractor are hAS_1 and hAS_2 , we have $OAEC(hAS_1) = \{hAS_1, hAS_2\}$. Based on the experiments performed using hAS_1 and hAS_2 , we have confirmed that their minimum size is $\gamma(PEC(OAEC(hAS_1))) =$

$\gamma(PEC(hAS_1)) = \gamma(PEC(hAS_2)) = 3$; additionally, hAS_1 and hAS_2 can be realized by MCT circuits using two CNOT gates and one Toffoli gate.

5.2.2 Full Adder/Subtractor

A full adder/subtractor acts as a full adder or a full subtractor depending on the value of the selector bit S . As seen in Table 5.1, two full adder/subtractor functions are possible, whose logic expressions can be represented using Equations (5.3) and (5.4), respectively.

$$\begin{cases} C/B = \bar{S}(AB \vee AC \vee BC) \vee S(\bar{A}B \vee BC \vee \bar{A}C) \\ S/D = A \oplus B \oplus C \end{cases} \quad (5.3)$$

$$\begin{cases} C/B = \bar{S}(\bar{A}B \vee BC \vee \bar{A}C) \vee S(AB \vee AC \vee BC) \\ S/D = A \oplus B \oplus C \end{cases} \quad (5.4)$$

Table 5.6: Incompletely specified full adder/subtractor function fAS_1 .

c	S	A	B	g_1	g_2	C/B	S/D
0	0	0	0	-	-	0	0
0	0	0	1	-	-	0	1
0	0	1	0	-	-	0	1
0	0	1	1	-	-	1	0
0	1	0	0	-	-	0	1
0	1	0	1	-	-	1	0
0	1	1	0	-	-	1	0
0	1	1	1	-	-	1	1
1	0	0	0	-	-	0	0
1	0	0	1	-	-	1	1
1	0	1	0	-	-	1	1
1	0	1	1	-	-	1	0
1	1	0	0	-	-	0	1
1	1	0	1	-	-	0	0
1	1	1	0	-	-	0	0
1	1	1	1	-	-	1	1

Similarly to half adder/subtractor, Equations (5.3) and (5.4) are embedded in the truth table, as in Tables 5.6 and 5.7, denoted by fAS_1 and fAS_2 , respectively. To balance the number of inputs and outputs, two garbage outputs, g_1 and g_2 , are added, but no additional lines are added. fAS_1 and fAS_2 can be realized in 4-line MCT circuits.

Table 5.7: Incompletely specified full adder/subtractor function fAS_2 .

c	S	A	B	g_1	g_2	C/B	S/D
0	0	0	0	-	-	0	0
0	0	0	1	-	-	1	1
0	0	1	0	-	-	1	1
0	0	1	1	-	-	1	0
0	1	0	0	-	-	0	1
0	1	0	1	-	-	0	0
0	1	1	0	-	-	0	0
0	1	1	1	-	-	1	1
1	0	0	0	-	-	0	0
1	0	0	1	-	-	0	1
1	0	1	0	-	-	0	1
1	0	1	1	-	-	1	0
1	1	0	0	-	-	0	1
1	1	0	1	-	-	1	0
1	1	1	0	-	-	1	0
1	1	1	1	-	-	1	1

In Table 5.8, the list of $OPT(PEC(fAS_1))$ is presented; the list is obtained using the FINDOPT program. It is observed that 30 functions have minimum MCT circuits with five gates, i.e., $\gamma(PEC(fAS_1)) = 5$. As an example of the minimum MCT circuit of the full adder/subtractor, “(1 0) (3 2) (2 1) (3 0) (0 2 3)” is shown in Fig. 5.3, where the numbering of the lines is $(S, A, B, C) = (0, 1, 2, 3)$. The circuit comprises four CNOT gates and one Toffoli gate.

We also obtained the list of $OPT(PEC(fAS_2))$, as shown in Table 5.9. However, the results are not better than those for $OPT(PEC(fAS_1))$. In summary, $OPT(PEC(fAS_2))$ comprises 26 functions and $\gamma(PEC(fAS_2)) = 5$. Thus, we have $\gamma(PEC(OAEC(fAS_1))) = \gamma(PEC(fAS_1)) = \gamma(PEC(fAS_2)) = 5$.

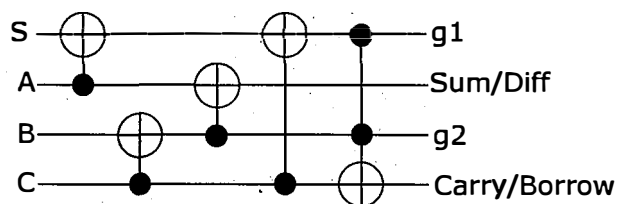


Figure 5.3: Proposed reversible full adder/subtractor circuit with MCT gates.

Table 5.8: $OPT(PEC(fAS_1))$ and its minimum MCT circuits.

Reversible Function	Minimum MCT circuit
(0 14 6 9 12 3 11 5 8 7 15 1 4 10 2 13)	(1 0) (3 2) (2 1) (3 0) (0 2 3)
(0 5 13 10 12 11 3 6 8 15 7 2 4 1 9 14)	(1 0) (2 3) (3 1) (2 0) (0 3 2)
(0 14 6 1 12 3 11 13 8 7 15 9 4 10 2 5)	(3 2) (1 0) (2 1) (2 3 0) (0 2 3)
(0 5 13 2 12 11 3 14 8 15 7 10 4 1 9 6)	(2 3) (1 0) (3 1) (2 3 0) (0 3 2)
(0 10 2 9 14 5 13 7 8 3 11 1 6 12 4 15)	(3 0) (1 0) (3 2) (0 2 3) (1 2)
(0 1 9 10 13 14 6 7 8 11 3 2 5 4 12 15)	(2 0) (1 0) (2 3) (0 3 2) (1 3)
(0 14 6 9 4 3 11 13 8 7 15 1 12 10 2 5)	(3 2) (1 2 0) (2 1) (3 0) (0 2 3)
(0 5 13 10 4 11 3 14 8 15 7 2 12 1 9 6)	(2 3) (1 3 0) (3 1) (2 0) (0 3 2)
(0 10 2 1 14 5 13 15 8 3 11 9 6 12 4 7)	(3 2) (1 0) (2 3 0) (0 2 3) (1 2)
(0 1 9 2 13 14 6 15 8 11 3 10 5 4 12 7)	(2 3) (1 0) (2 3 0) (0 3 2) (1 3)
(0 14 6 9 4 11 3 13 8 7 15 1 12 2 10 5)	(3 0) (3 2) (1 2 3) (2 1) (0 2 3)
(0 5 13 10 4 3 11 14 8 15 7 2 12 9 1 6)	(2 0) (2 3) (1 3 2) (3 1) (0 3 2)
(0 1 9 10 5 6 14 15 8 11 3 2 13 12 4 7)	(2 0) (2 3) (1 3 2) (0 3 2) (1 3)
(0 10 2 9 6 13 5 15 8 3 11 1 14 4 12 7)	(3 0) (3 2) (1 2 3) (0 2 3) (1 2)
(0 10 2 9 6 5 13 15 8 3 11 1 14 12 4 7)	(3 2) (3 0) (1 2 0) (0 2 3) (1 2)
(0 1 9 10 5 14 6 15 8 11 3 2 13 4 12 7)	(2 3) (2 0) (1 3 0) (0 3 2) (1 3)
(0 6 14 9 4 3 11 13 8 15 7 1 12 10 2 5)	(2 0) (3 2) (2 1) (1 2 3) (0 2 3)
(0 13 5 10 4 11 3 14 8 7 15 2 12 1 9 6)	(3 0) (2 3) (3 1) (1 3 2) (0 3 2)
(0 5 13 10 12 11 2 7 8 15 6 3 4 1 9 14)	(2 1) (1 0) (3 1) (0 2 3) (0 3 2)
(0 14 6 9 12 1 11 7 8 5 15 3 4 10 2 13)	(3 1) (1 0) (2 1) (0 3 2) (0 2 3)
(0 5 13 2 4 11 3 6 8 15 7 10 12 1 9 14)	(1 3 0) (2 3) (3 1) (1 2 0) (0 3 2)
(0 14 6 1 4 3 11 5 8 7 15 9 12 10 2 13)	(1 2 0) (3 2) (2 1) (1 3 0) (0 2 3)
(0 14 6 1 4 3 11 13 8 7 15 9 12 10 2 5)	(1 2 0) (3 2) (2 1) (1 2 3 0) (0 2 3)
(0 5 13 2 4 11 3 14 8 15 7 10 12 1 9 6)	(1 3 0) (2 3) (3 1) (1 2 3 0) (0 3 2)
(0 5 13 2 4 3 11 6 8 15 7 10 12 9 1 14)	(2 3) (2 3 0) (1 3 2) (3 1) (0 3 2)
(0 14 6 1 4 11 3 5 8 7 15 9 12 2 10 13)	(3 2) (2 3 0) (1 2 3) (2 1) (0 2 3)
(0 10 2 1 6 5 13 7 8 3 11 9 14 12 4 15)	(3 2) (2 3 0) (1 2 0) (0 2 3) (1 2)
(0 1 9 2 5 14 6 7 8 11 3 10 13 4 12 15)	(2 3) (2 3 0) (1 3 0) (0 3 2) (1 3)
(0 1 9 2 5 6 14 7 8 11 3 10 13 12 4 15)	(2 3) (2 3 0) (1 3 2) (0 3 2) (1 3)
(0 10 2 1 6 13 5 7 8 3 11 9 14 4 12 15)	(3 2) (2 3 0) (1 2 3) (0 2 3) (1 2)

Table 5.9: $OPT(PEC(fAS_2))$ and its minimum MCT circuits.

Reversible Function	Minimum MCT circuit
(0 15 7 10 12 1 9 6 8 5 13 2 4 11 3 14)	(0 1) (2 0) (3 1) (3 2) (1 0 3)
(0 7 15 9 12 10 2 5 8 14 6 1 4 3 11 13)	(1 0) (2 1) (3 0) (3 2) (0 1 3)
(0 15 7 9 4 10 2 13 8 6 14 1 12 3 11 5)	(1 0) (2 1) (3 0) (0 1 3) (0 1 2)
(0 7 15 10 4 1 9 14 8 13 5 2 12 11 3 6)	(0 1) (2 0) (3 1) (1 0 3) (1 0 2)
(0 15 7 10 4 1 9 14 8 5 13 2 12 11 3 6)	(0 1) (2 0) (3 1) (3 0 2) (1 0 3)
(0 7 15 9 4 10 2 13 8 14 6 1 12 3 11 5)	(1 0) (2 1) (3 0) (3 1 2) (0 1 3)
(0 3 11 9 14 12 4 7 8 10 2 1 6 5 13 15)	(3 1) (3 2) (1 0) (0 1 3) (1 2)
(0 11 3 10 13 4 12 7 8 1 9 2 5 14 6 15)	(3 0) (3 2) (0 1) (1 0 3) (0 2)
(0 15 7 10 9 4 12 3 8 5 13 2 1 14 6 11)	(0 2) (3 0) (0 1) (2 0) (1 2 3)
(0 7 15 9 10 12 4 3 8 14 6 1 2 5 13 11)	(1 2) (3 1) (1 0) (2 1) (0 2 3)
(0 15 7 2 12 1 9 14 8 5 13 10 4 11 3 6)	(0 1) (2 0) (3 2) (3 0 1) (1 0 3)
(0 7 15 1 12 10 2 13 8 14 6 9 4 3 11 5)	(1 0) (2 1) (3 2) (3 0 1) (0 1 3)
(0 15 6 10 12 1 9 7 8 5 13 3 4 11 2 14)	(2 0) (3 2) (2 1) (0 1 3) (1 0 3)
(0 5 15 9 12 10 2 7 8 14 6 3 4 1 11 13)	(2 1) (3 2) (2 0) (1 0 3) (0 1 3)
(0 15 3 10 9 4 12 7 8 1 13 2 5 14 6 11)	(0 2) (3 0) (0 1) (2 0 3) (1 2 3)
(0 3 15 9 10 12 4 7 8 14 2 1 6 5 13 11)	(1 2) (3 1) (1 0) (2 1 3) (0 2 3)
(0 15 7 1 4 10 2 5 8 6 14 9 12 3 11 13)	(1 0) (2 1) (3 0 1) (0 1 3) (0 1 2)
(0 7 15 2 4 1 9 6 8 13 5 10 12 11 3 14)	(0 1) (2 0) (3 0 1) (1 0 3) (1 0 2)
(0 15 7 2 4 1 9 6 8 5 13 10 12 11 3 14)	(0 1) (2 0) (3 0 1) (3 0 2) (1 0 3)
(0 7 15 1 4 10 2 5 8 14 6 9 12 3 11 13)	(1 0) (2 1) (3 0 1) (3 1 2) (0 1 3)
(0 15 7 10 4 9 1 14 8 5 13 2 12 3 11 6)	(3 0) (0 1) (1 0 2) (2 0) (1 0 3)
(0 7 15 9 4 2 10 13 8 14 6 1 12 11 3 5)	(3 1) (1 0) (0 1 2) (2 1) (0 1 3)
(0 3 11 9 6 4 12 15 8 10 2 1 14 13 5 7)	(3 1) (1 0) (0 1 2) (0 1 3) (1 2)
(0 11 3 10 5 12 4 15 8 1 9 2 13 6 14 7)	(3 0) (0 1) (1 0 2) (1 0 3) (0 2)
(0 3 11 9 6 12 4 15 8 10 2 1 14 5 13 7)	(3 1) (1 0) (3 1 2) (0 1 3) (1 2)
(0 11 3 10 5 4 12 15 8 1 9 2 13 14 6 7)	(3 0) (0 1) (3 0 2) (1 0 3) (0 2)

5.3 Simple Reversible Arithmetic Logic Units (ALUs)

The OA works more efficiently for the design of ALUs than adder/ subtractors since an ALU has more operations assigned by the selector bits. This section reports the minimum MCT circuits of some benchmarks of simple reversible ALUs consisting of only four or eight 1-bit operations. It should be noted that this is not a limitation of the OA technique but of the minimizer. Instead of the exact minimizer, it is possible to use a fast heuristic synthesizer supporting ISRFs to obtain MCT circuits of larger ALUs without guaranteeing the minimality. In this thesis, however, we focus on obtaining minimum MCT circuits. The minimum results are valuable as scientific and theoretical data in related fields. Our next task is to apply a heuristic method to equivalence classes for larger ALUs [36].

5.3.1 1-bit Arithmetic Logic Unit (ALU) Benchmarks

We applied our function design technique to 1-bit ALU benchmarks, i.e., Mini-ALU [34] and Gupta's-LU [33]. The embedding in the truth table of the reversible ALU is similar to that of the reversible adder/subtractor. In this chapter, we do not show the truth table representation of the reversible ALUs. The truth table representation of those reversible ALUs; it can be found in Appendix A.

The FINDOPT program confirmed that $\gamma(PEC(OAEC(\text{Mini-ALU}))) = 5$. The list of $OPT(PEC(OAEC(\text{Mini-ALU})))$ is omitted, but its number is $|OPT(PEC(OAEC(\text{Mini-ALU})))| = 266$. One of the minimum MCT circuits is shown in Fig. 5.4. The MCT circuit reported by Revlib [34] consists of six gates. Here, we reduced the number of reversible gates to five.

For Gupta's-LU, $\gamma(PEC(OAEC(\text{Gupta's-LU}))) = 3$ and $|OPT(PEC(OAEC(\text{Gupta's-LU})))| = 48$. One of the minimum MCT circuits is shown in Fig. 5.5. Originally Gupta's-LU [33] was constructed using 18 reversible gates. Here, we reduced the number of reversible gates to three.

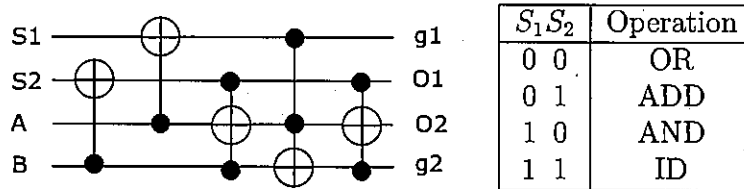


Figure 5.4: Mini-ALU: Minimum MCT circuit and its OA.

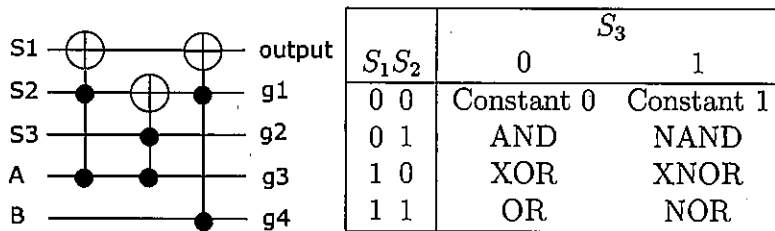


Figure 5.5: Gupta's-LU: Minimum MCT circuit and its OA.

5.3.2 Revised Arithmetic Logic Units (ALUs)

As seen in Sec. 5.2, the addition and subtraction operations are essential in computation. It is preferable for an ALU to have these two operations. Since Mini-ALU does not have subtraction, we propose a revised ALU, denoted by ALU, where subtraction

is adopted instead of ID in Mini-ALU. Using the truth tables in Tables 4.1 and 4.2 as examples of the OA and output permutation, we confirmed that $\gamma(\text{ALU}) = 7$, $\gamma(\text{OAEC}(\text{ALU})) = 6$, and $\gamma(\text{PEC}(\text{ALU})) = 6$, as in Examples 2 and 3. By combining *OAEC* and *PEC*, further reduction to $\gamma(\text{PEC}(\text{OAEC}(\text{ALU}))) = 5$ is possible. The list of $\text{OPT}(\text{PEC}(\text{OAEC}(\text{ALU})))$ is omitted, but its number is $|\text{OPT}(\text{PEC}(\text{OAEC}(\text{ALU})))| = 48$. One of the minimum MCT circuits is shown in Fig. 5.6.

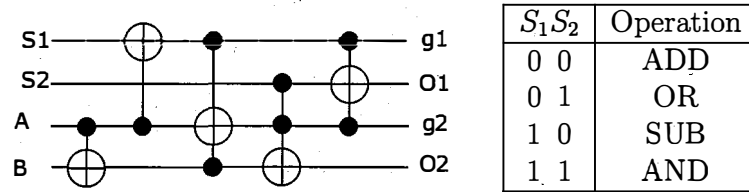


Figure 5.6: ALU: Minimum MCT circuit and its OA.

In contrast, we consider a simpler version of Gupta's-LU. Gupta's-LU implements eight operations with five inputs. By reducing the operations to AND, OR, and XOR, a compact version with four inputs can be defined, which is denoted by LU. We select AND, OR, and XOR because they are the most fundamental and common logic operations. The NOT operation is included in XOR. NOT can be executed by assigning the value 1 to one of the XOR operands: $\bar{x} = x \oplus 1$. The compact LU is useful when this set of operations are sufficient for the use of logic units. The FIND-OPT program confirmed that $\gamma(\text{PEC}(\text{OAEC}(\text{LU}))) = 3$ and $|\text{OPT}(\text{PEC}(\text{OAEC}(\text{LU})))| = 28$. One of the minimum MCT circuits is shown in Fig. 5.7.

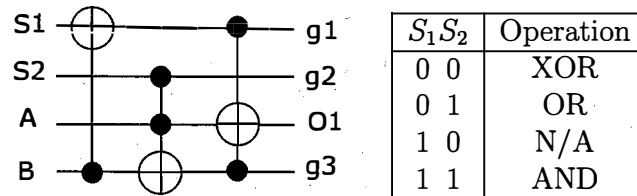


Figure 5.7: LU: Minimum MCT circuit and its OA.

5.4 Experimental Results

In this section, a comparison of our proposed adder/subtractor and ALUs and their existing counterparts is performed. Then, we discuss the results and show the superiority of the proposed reversible circuits against other reversible structures in the literature in terms of reversible metrics.

The proposed minimization algorithm was implemented using the Common LISP programming language. The program was run on a computer with specifications given in Table 5.10. Since the adder/subtractors and ALUs function specifications are different, we executed the program separately for adder/subtractors and ALUs. Our algorithm provides a list of minimum circuits for reversible functions, and we recorded all the minimum circuits for analysis. Table 5.4 shows six reversible functions that are minimum MCT circuits with three gates for the half adder/subtractor hAS_1 . Table 5.5 shows another three reversible functions that are minimum MCT circuits with three gates for the half adder/subtractor hAS_2 . In addition to Tables 5.4 and 5.5, we also recorded 254 and 166 reversible functions with a size of 4 gates and 5858 and 4325 reversible functions with a size of 5 gates for hAS_1 and hAS_2 , respectively. The results of Table 5.5 and other recorded reversible functions are not better than those of Table 5.4. Hence, we confirmed that the half adder/subtractor minimum size is 3. Figure 5.2 shows the proposed reversible realization of half adder/subtractors using two CNOT gates and one Toffoli gate.

Table 5.10: Computer specifications.

CPU	Intel Core i7-3820 @ 3.60GHz
Memory	32GB
OS	Linux: Ubuntu 16.04 LTS
Language	Common LISP (SBCL 1.3.1.debian)

Table 5.11: Summary of all generated reversible functions for adder/subtractors.

Design	Gate Count			
	Three	Four	Five	Six
Half adder/subtractor (hAS_1)	6	254	5858	95915
Half adder/subtractor (hAS_2)	3	166	4325	76694
Full adder/subtractor (fAS_1)	-	-	30	662
Full adder/subtractor (fAS_2)	-	-	26	609

Table 5.12: Computation time [second] of experiments in Table 5.11.

Design	Gate Count			
	Three	Four	Five	Six
Half adder/subtractor (hAS_1)	0.9	11.8	185.9	2764.6
Half adder/subtractor (hAS_2)	0.9	11.8	183.8	2726.5
Full adder/subtractor (fAS_1)	-	-	198.6	2632.9
Full adder/subtractor (fAS_2)	-	-	189.1	2599.8

Table 5.8 shows 30 reversible functions that are minimum MCT circuits with five gates for the full adder/subtractor fAS_1 . Table 5.9 shows another 26 reversible functions that are minimum MCT circuits with five gates for the full adder/subtractor fAS_2 . In addition to Tables 5.8 and 5.9, we also recorded 662 and 609 reversible functions with a size of 6 gates for fAS_1 and fAS_2 , respectively. The results of Table 5.9 and other recorded reversible functions are not better than those of Table 5.8. Hence, we confirmed that the full adder/subtractor minimum size is 5. Figure 5.3 shows the proposed reversible realization of full adder/subtractors using four CNOT gates and one Toffoli gate. A summary of all the generated reversible functions for half and full adder/subtractors are shown in Table 5.11. Precisely, the figures in Table 5.11 represent $|\{F' \mid F' \in PEC(F), \gamma(F') = r\}|$ for design F and gate count r . Table 5.12 shows the computation time of the experiments in Table 5.11. Note that we recorded all the generated reversible functions for ALUs, but only the essential functions are discussed in here.

Table 5.13 shows the comparison of the proposed reversible adder/subtractor and existing counterparts. The results in Table 5.13 are obtained by evaluating our proposed one-bit reversible full adder/subtractor unit using our minimization algorithm versus other similar designs described in [27], [29], and [30]. Our proposed structure includes the lowest numbers of reversible gates, constant inputs and garbage outputs among all existing designs considered.

Table 5.13: Comparison of the proposed reversible full adder/subtractor and existing counterparts.

Design	Cost Metrics			
	Input Lines	Reversible Gates	Constant Inputs	Garbage Outputs
1 [27]	7	8	3	5
2 [27]	5	4	1	3
3 [27]	5	4	1	3
[29]	4	4	0	2
[30]	5	3	1	2
Proposed	4	5	0	2

Note, however, that the number of reversible gates of our proposed design is not the lowest because we only used the MCT gate library as opposed to existing works that used different types of gate libraries. In particular, the proposed full adder/subtractor design in [27] used Fredkin, CNOT, TR, and Peres gates. The proposed design in [29] used CNOT, Fredkin, and Peres gates. Finally, the proposed design in [30] used

CNOT, and Peres gates. Our proposed reversible full adder/subtractor block is designed to optimize all significant criteria in reversible logic without imposing high costs to other parameters.

A summary of all the generated reversible functions for 1-bit ALUs is detailed in Table 5.14, showing the number of functions with minimum size and the computation time in second. The specifications of the computer used in the experiments are the same as in Table 5.10. Although the applicability of our minimizer is limited to small benchmarks, it can minimize 1-bit ALUs in practical computation time.

Table 5.14: $|OPT(PEC(OAEC(F)))|$ of 1-bit ALUs and computation time.

Design	$ OPT(PEC(OAEC(F))) $	Time [second]
Gupta's-LU	48	47.7
Mini-ALU	266	197.4
ALU	48	194.8
LU	28	0.9

Table 5.15 shows the comparison of 1-bit ALUs of the proposed ALUs and existing implementations. All the existing implementations are performed by cascading reversible gates, hence the circuits are not optimized, which translates to higher costs. In Table 5.15, Gupta's-LU [33] can perform eight operations, which require five working lines and 18 reversible gates. Mini-ALU [34] can perform four operations, which require four working lines and 6 reversible gates. By implementing the same design our minimization algorithm obtains 3 and 5 reversible gates for Gupta's-LU [33] and Mini-ALU [34], respectively. These results are shown in the fifth column of Table 5.15. The proposed LU realizes basic logic operations such as AND, OR, XOR, and N/A. The proposed ALU realizes the basic logic operations, such as AND and OR, and the basic arithmetic operations such as ADD and SUB; note that SUB is not realized by existing ALUs [34].

The costs of the proposed reversible ALU are shown in Table 5.15, including the number of operations, working lines, original gates for the existing design, and proposed gates with our minimization algorithm. Considering these costs, the proposed reversible ALU is much better and optimized than existing ones. We can consider that the proposed design complies with the standard of reversible optimization circuit as we only used the MCT gate library. Implementing more arithmetic/logic operations in a reversible ALU needs a more complex circuit structure. More powerful ALUs with more arithmetic/logic functions are needed to design a quantum computer.

Table 5.15: Comparison of 1-bit ALUs.

Design	Operations	Lines	Original Gates	Our Gates
Gupta's- LU [33]	AND, NAND, OR, NOR, XOR, XNOR, Constant 1, 0	5	18	3
Mini-ALU [34]	AND, OR, ADD, ID	4	6	5
Proposed ALU	ADD, OR, AND, SUB	4		5
Proposed LU	XOR, OR, AND, N/A	4		3

Chapter 6

Quantum Circuit Synthesis

Circuit synthesis is an integral part of the compilation process. Quantum circuit synthesis is the process where an arbitrary unitary operation is decomposed into a sequence of gates from a universal set. There are several procedures for quantum circuit synthesis with made for a single qubit and others for multiple qubits. Many of the algorithms can perform exact synthesis and heuristic search technique, but the time and space used exponentially increases, which highly depend on the number of qubits and the depth of the circuit. In this work, we propose a simple greedy algorithm for quantum circuit synthesis to transform reversible circuits into quantum circuits and reduce the quantum circuit by applying different quantum circuit reduction rules. The algorithm performs the transformation of a given MCT circuit into a quantum circuit, searches a pair of adjacent gates in the circuit by moving the gate according to the moving rule, and applies different reduction rules to obtain the reduced quantum circuit. The execution of moving and reducing gates is repeated if the quantum gates in the quantum circuit are reduced successively. The experimental results show that our proposed design is better than existing ones in terms of the numbers of input lines, constant inputs, garbage outputs, and the QC.

Several reversible circuits of adder/subtractor blocks [27, 29, 30] have been proposed that use various reversible gate libraries and impose different constraints of constant and garbage lines. To compare the quality of the circuits, calculation of their QCs is commonly performed. In this chapter, we describe our proposed greedy algorithm, discuss how quantum circuits of our reversible adder/subtractors and ALUs are obtained using the algorithm, and compare the QCs of our proposed design with existing counterparts.

6.1 Reduction of Quantum Costs (QCs)

```

1: function OBTAINQUANTUMCIRCUIT(mctC)
2:                                     ▷ Input: mctC is an MCT circuit.
3:                                     ▷ Output: a reduced quantum circuit.
4:   return a circuit with the smallest QC in {REDUCEQUANTUM(qC) | qC ∈
      QUANTUMCIRCUITS(mctC)};
5: end function

6: function REDUCEQUANTUM(qC)
7:                                     ▷ Input: qC is a quantum circuit.
8:                                     ▷ Output: a reduced quantum circuit.
9:   var C : Circuit;
10:  var C* ← qC : Circuit;
11:  repeat
12:    C ← C*;
13:    for all gate g in C do
14:      C* ← MOVEGATEANDREDUCE(C, g);
15:      if |C*| < |C| then
16:        break;                                     ▷ Exiting the for loop.
17:      end if
18:    end for
19:  until |C*| = |C|
20:  return C*;
21: end function

```

|*C*|: the number of gates in the quantum circuit *C*.

Figure 6.1: Simple algorithm to obtain reduced quantum circuits.

This section describes the transformation of MCT circuits into quantum circuits with the NCV gate library and the reduction of the quantum circuits with a simple greedy algorithm. The algorithm is shown in Fig. 6.1. In this algorithm, the function OBTAINQUANTUMCIRCUIT takes an MCT circuit as the argument and returns the reduced quantum circuit. Unlike the MCT circuits in Chapter 5, the resulting quantum circuits are sub-optimal. Although optimality is not guaranteed, the QCs of our quantum circuits of reversible adder/subtractors and ALUs are lower than those of existing counterparts.

The function QUANTUMCIRCUITS transforms an MCT circuit *mctC* into a set of initial quantum circuits. For Toffoli gates in *mctC*, QUANTUMCIRCUITS generates all possible combinations of Toffoli decomposition described in Sec. 2.7 during the transformation.

The function REDUCEQUANTUM reduces the QC of a quantum circuit qC via the greedy approach. In REDUCEQUANTUM, the function MOVEGATEANDREDUCE searches a pair of adjacent gates in C by moving the gate g according to the moving rule of Property 1. If a pair of gates forms the identity function (according to the deletion rules shown in Fig. 2.4), then the pair is deleted. If a pair of gates can be merged into a single gate (according to the merging rules shown in Fig. 2.3), then the pair is replaced by a single gate. The execution of MOVEGATEANDREDUCE is repeated if the quantum gates in the circuit C are reduced successively.

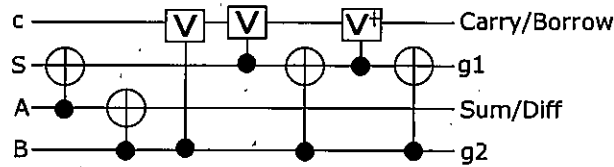


Figure 6.2: Reduced half adder/subtractor circuit with quantum gates.

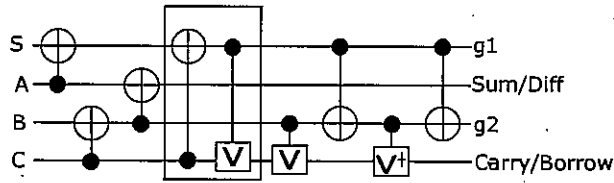


Figure 6.3: Reduced full adder/subtractor circuit with quantum gates.

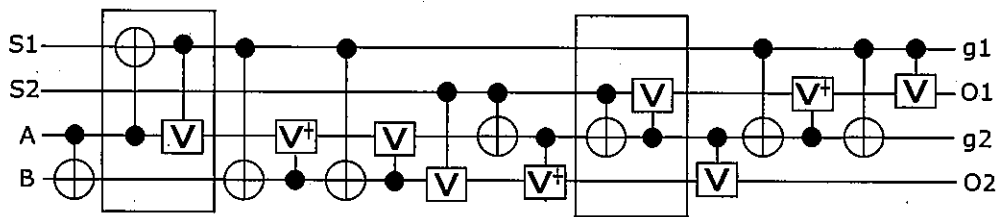


Figure 6.4: Reduced ALU circuit with quantum gates.

In Chapter 5, we discussed the minimization algorithm and obtained a list of reversible functions for adder/subtractors and ALUs. The algorithm returned the minimum MCT circuits for adder/subtractors and ALUs when compared among all the obtained reversible functions. In the experimental results section, we illustrated the improvements offered by the proposed design over existing counterparts. In this chapter, we discuss the greedy algorithm that obtains reduced quantum circuits from MCT circuits. We transform all reversible circuits into quantum circuits for

adder/subtractors and ALUs, which are obtained through our proposed minimization algorithm. Our proposed greedy algorithm returns the best quantum circuit realization among all the transformed reversible circuits. In this work, the best quantum circuit is determined by counting the required number of quantum gates. The less the number of quantum gates is needed to realize a reversible circuit, the better the quantum circuit is. Figures 6.2, 6.3, and 6.4 show the reduced quantum gate realization of half adder/subtractor, full adder/subtractor, and ALU, respectively. The square box in Figs. 6.3 and 6.4 is the 2-qubit gate rule in the QC metrics, as described in Sec. 2.6. According to the 2-qubit gate rule, if both CNOT and Controlled-V (or Controlled-V[†]) are operating on the same two qubits in a symmetric pattern, as shown in Fig. 2.5, their total QC is considered to be one as well.

6.2 Comparison of Quantum Costs (QCs)

In this section, we discuss further details of the transformation of reversible circuits into quantum circuits. Since there are multiple ways to realize a reversible gate into quantum circuits, it is difficult to definitely say whether the realization is optimal or minimal. Moreover, we show that there are multiple reversible functions for any particular function specification. Several techniques can be used to return exactly one circuit for any particular function specification, but that reversible circuit may not provide better quantum gate realization. our approach obtains a list of reversible functions for any particular function specification, and all those functions are minimum reversible circuits. To determine the quality of the obtained reversible circuits, we transform them into quantum circuits. In the previous section, we proposed a greedy algorithm and implemented it using MATLAB, as shown in Fig. 6.1. The program was run on a computer with specifications described in Table 6.1. In this section, we only discuss the adder/subtractor and omitted details of the results for ALUs. Tables 6.2 and 6.3 show the QC of half and full adder/subtractors for all minimum MCT circuits shown in Tables 5.4 and 5.8, respectively, where the ‘After Reduction’ columns indicate the QC after executing our program. The smallest quantum circuits in Tables 6.2 and 6.3 are shown in Figs. 6.2 and 6.3, respectively. Although, the number of elementary quantum gates in Fig. 6.3 is nine, we calculated the QC to be eight according to the 2-qubit gate rule in Fig. 2.5 of Sec. 2.6.

Tables 6.2 and 6.3 show the effect of using our proposed greedy algorithm. The second column (QC) of each table is divided into two columns, i.e., ‘Before Reduction’ and ‘After Reduction’. As shown in Table 6.2, the QC of circuit no. 2 before

Table 6.1: Computer specifications.

CPU	Intel Core i7-4770S @ 3.10GHz
Memory	16GB
OS	Windows 7 Professional 64 bit Operating System
Language	MATLAB R2018a

Table 6.2: Reduction of QC in the half adder/subtractor hAS_1 .

Circuit No	QC	
	Before Reduction	After Reduction
1	7	7
2	11	8
3	11	8
4	7	7
5	11	10
6	11	10

applying the greedy algorithm is 11, and it becomes 8 after applying the algorithm, which means that the reduced quantum circuit requires 3 fewer gates. Note that there are no reduced quantum circuit realizations for circuits no. 1 and 4. The reduced quantum gates are over 25% of the original quantum gates. As shown in Table 6.3, the QC of circuit no. 26 before applying the greedy algorithm is 17, and it becomes 11 after applying the algorithm, which means that the reduced quantum circuit requires 6 fewer gates. There are only few circuits where only 1 gate can be reduced after applying the algorithm. From Table 6.3, the maximum possible quantum gates reduction is over 30%, and the minimum possible quantum gates reduction is around 10%. Table 6.4 shows the computation time of the experiments in Tables 6.2 and 6.3.

6.3 Experimental Results

Table 6.5 shows the comparison of existing full adder/subtractor designs with different reversible gate libraries. As far as we know, the proposed design of the adder/subtractor is better than existing counterparts in terms of the QC, number of constant inputs, and garbage outputs. As shown in Table 6.5, the number of reversible gates in [30, 29, 27] is less than that in our proposed design because of the use of different gate libraries; we only used the MCT gate library in this work. The ‘QC’ column of Table 6.5 demonstrates that our proposed design requires the least number of quantum gates to realize the full adder/subtractor block without imposing any high costs to other parameters.

Table 6.3: Reduction of QC in the full adder/subtractor fAS₁.

Circuit No	QC	
	Before Reduction	After Reduction
1	9	8
2	9	8
3	13	11
4	13	11
5	9	8
6	9	8
7	13	10
8	13	10
9	13	9
10	13	11
11	13	9
12	13	9
13	13	11
14	13	11
15	13	11
16	13	11
17	13	12
18	13	12
19	13	12
20	13	12
21	17	16
22	17	16
23	26	23
24	26	23
25	17	13
26	17	11
27	17	12
28	17	14
29	17	15
30	17	13

Table 6.4: $|OPT(PEC(F))|$ of adder/subtractor and computation time for QC reduction.

Design	$ OPT(PEC(F)) $	Time [second]
hAS ₁	6	0.4
fAS ₁	30	0.5

Table 6.5: Comparison of the proposed reversible full adder/subtractor and existing counterparts.

Design	Cost Metrics				
	Input Lines	Reversible Gates	Constant Inputs	Garbage Outputs	QC
1 [27]	7	8	3	5	21
2 [27]	5	4	1	3	14
3 [27]	5	4	1	3	10
[29]	4	4	0	2	11
[30]	5	3	1	2	9
Proposed	4	5	0	2	8

Table 6.6 shows the comparison of 1-bit ALUs, in which ‘Gates’ and ‘QC’ denote the number of MCT gates and the QC, respectively. ‘QC*’ is the quantum cost of the 2-qubit gate rule. If both CNOT and Controlled-V (or Controlled-V[†]) are operating on the same two qubits in a symmetric pattern, as shown in Fig. 2.5, their total QC is considered to be one as well. The QC of Gupta’s-LU and Mini-ALU are significantly reduced in our design. The original number of reversible gates required to realize Gupta’s-LU is 18 with a corresponding QC of 114. When our minimization algorithm is applied, only 3 reversible gates are enough to realize the Gupta’s-LU with a QC of 15. When the 2-qubit gate rule is considered, the QC is 14. Over 85% QC is reduced in our design compared to that of the original Gupta’s-LU.

Table 6.6: Comparison of 1-bit ALUs.

Design	Operations	Lines	Original Gates/QC	Our Gates /QC(QC*)
Gupta's-LU [33]	AND, NAND, OR, NOR, XOR, XNOR, Constant 1, 0	5	18/114	3/15(14)
Mini-ALU [34]	AND, OR, ADD, ID	4	6/62	5/15(14)
Proposed ALU	ADD, OR, AND, SUB	4		5/17(15)
Proposed LU	XOR, OR, AND, N/A	4		3/11(10)

QC*: QC under the merged 2-qubit gate rule.

The original reversible gates required to realize the Mini-ALU is 6 with a corresponding QC of 62. When our minimization algorithm is applied, 5 reversible gates

are enough to realize the Gupta's-LU with a QC of 15. When the 2-qubit gate rule is considered, the QC is 14. Over 75% QC is reduced in our design compared to that of the original Mini-ALU. The last two rows of Table 6.6 show the proposed ALU and LU. The best QC cost of the proposed ALU is 17 (15, when the 2-qubit gate rule is considered), which requires 5 reversible gates, and the best QC of the proposed LU is 11 (10, when the 2-qubit gate rule is considered), which requires only 3 reversible gates.

Chapter 7

Conclusions and Future Works

For most computing devices, adder/subtractor blocks and ALUs are crucial, and the development of cost-efficient arithmetic blocks helps improve the efficiency of the whole system. In this thesis, we studied the logic synthesis and optimization of reversible and quantum circuits for ALUs by putting an emphasis on the function design. Toffoli, Peres, and Fredkin are conventionally used to synthesize reversible circuits, and we have adopted the MCT gate library to design reversible adder/subtractor and simple ALU circuits. As far as we know, there are no existing works on the optimization of reversible full adder/subtractor circuits using only the MCT gate library with the lowest possible number of working lines, constant inputs, and garbage outputs even though the MCT gate library is the most fundamental and widely-used gate library for the synthesis of reversible circuits.

To obtain a reversible circuit of an irreversible function, we embedded the irreversible function into an incompletely-specified function to make it reversible. We introduced a new approach called synthesis with OA, where the set of all functions that is OA-equivalent to F is called the OA-equivalence class of F and denoted by $OAEC(F)$. We also used the idea of permutation of outputs of a reversible function, originally introduced by Wille et al. [23], called as SWOP. For a given ISRF F , the set of variants made by the permutation of outputs in F is called the P-equivalence class of F denoted by $PEC(F)$. To improve the function design of the arithmetic blocks, we proposed a combination of the equivalence classes $PEC(OAEC(F))$ of ISRFs F and extended the minimization problem of F into $\gamma(\mathcal{F})$ and $OPT(\mathcal{F})$ for the set of ISRFs \mathcal{F} .

We proposed a minimization algorithm to obtain the minimum reversible circuits. We also used a greedy algorithm to perform the transformation of a given reversible circuit into a quantum circuit. These techniques allowed us to obtain the reduced

quantum circuit. The usefulness of our proposed technique, algorithms, and approaches were verified through experiments. In this work, we applied our function design approach and provided the minimum MCT circuits that realize half and full adder/subtractors and some benchmark ALUs. The experimental results showed that our proposed reversible designs and quantum designs are better than existing ones in terms of the number of input lines, constant inputs, garbage outputs, and the QC.

Even though research in this area is still on its early stages, promising applications to future computing devices motivate further research. In this thesis, the function design for reversible logic synthesis using several approaches was proposed, and the presented synthesis approach can be further investigated. Some possible future works are listed below.

- The minimization algorithm needs to be improved because it is currently limited to small circuits because of high memory consumption. However, even with this limitation, it is sufficient to achieve the purpose of this work.
- The proposed OA approach is very efficient only when the number of operations is small. Hence the proposed approach can be further improved to be used on larger number of operations.
- We focused on the function design of reversible and quantum logic synthesis. The proposed design of ALUs can only perform four operations because implementing more arithmetic/logic operations in a reversible ALU requires more complex circuit structure. More powerful ALUs with more arithmetic/logic operations are needed to design a quantum computer. Our proposed function design approach to optimize reversible circuits can still be further improved, especially for adopting more arithmetic/logic operations.
- We only used the NCV gate library into transform MCT circuits to quantum circuits. Recently, a universal quantum gate set $\{H, Z, S, T, CNOT\}$ has been taken into account in the design of fault-tolerant quantum circuits. Further investigation to test whether the proposed approach to transform of MCT circuits into quantum circuits can be adopted for such circuits is needed.
- Recently, the linear nearest neighbor (LNN) architecture has received significant attention because of its practical uses especially in various nanotechnologies, such as quantum optics, linear ion Trap, and nuclear magnetic resonance (NMR), where every qubit can interact with at most one neighbor above and

below it. Therefore, it is important to develop a design methodology for this new architecture. From the different technological constraints, our future plan is to revise our synthesis approach and design more practical reversible and quantum adder/subtractors and ALUs.

Appendix A

Truth Table Representation of Reversible Arithmetic Logic Units (ALUs)

Table A.1: Truth table representation of reversible LU.

S_1	S_2	A	B	G_1	G_2	G_3	$Output$
0	0	0	0	-	-	-	0
0	0	0	1	-	-	-	1
0	0	1	0	-	-	-	1
0	0	1	1	-	-	-	0
0	1	0	0	-	-	-	0
0	1	0	1	-	-	-	1
0	1	1	0	-	-	-	1
0	1	1	1	-	-	-	1
1	0	0	0	-	-	-	0
1	0	0	1	-	-	-	0
1	0	1	0	-	-	-	0
1	0	1	1	-	-	-	1
1	1	0	0	-	-	-	-
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	-	-	-	-

Table A.2: Truth table representation of reversible ALU.

S_1	S_2	A	B	G_1	G_2	$Output_1$	$Output_2$
0	0	0	0	-	-	0	0
0	0	0	1	-	-	0	1
0	0	1	0	-	-	0	1
0	0	1	1	-	-	1	0
0	1	0	0	-	-	-	0
0	1	0	1	-	-	-	1
0	1	1	0	-	-	-	1
0	1	1	1	-	-	-	1
1	0	0	0	-	-	-	0
1	0	0	1	-	-	-	0
1	0	1	0	-	-	-	0
1	0	1	1	-	-	-	1
1	1	0	0	-	-	0	0
1	1	0	1	-	-	1	1
1	1	1	0	-	-	0	1
1	1	1	1	-	-	0	0

Table A.3: Truth table representation of Gupta's-LU.

S_0	S_1	S_2	A	B	G_0	G_1	G_2	G_3	<i>Output</i>
0	0	0	0	0	-	-	-	-	1
0	0	0	0	1	-	-	-	-	1
0	0	0	1	0	-	-	-	-	1
0	0	0	1	1	-	-	-	-	1
0	0	1	0	0	-	-	-	-	0
0	0	1	0	1	-	-	-	-	1
0	0	1	1	0	-	-	-	-	1
0	0	1	1	1	-	-	-	-	1
0	1	0	0	0	-	-	-	-	1
0	1	0	0	1	-	-	-	-	1
0	1	0	1	0	-	-	-	-	1
0	1	0	1	1	-	-	-	-	0
0	1	1	0	0	-	-	-	-	0
0	1	1	0	1	-	-	-	-	1
0	1	1	1	0	-	-	-	-	1
0	1	1	1	1	-	-	-	-	0
1	0	0	0	0	-	-	-	-	1
1	0	0	0	1	-	-	-	-	0
1	0	0	1	0	-	-	-	-	0
1	0	0	1	1	-	-	-	-	1
1	0	1	0	0	-	-	-	-	0
1	0	1	0	1	-	-	-	-	0
1	0	1	1	0	-	-	-	-	0
1	0	1	1	1	-	-	-	-	1
1	1	0	0	0	-	-	-	-	1
1	1	0	0	1	-	-	-	-	0
1	1	0	1	0	-	-	-	-	0
1	1	0	1	1	-	-	-	-	0
1	1	1	0	0	-	-	-	-	0
1	1	1	0	1	-	-	-	-	0
1	1	1	1	0	-	-	-	-	0
1	1	1	1	1	-	-	-	-	0

Table A.4: Truth table representation of Mini-ALU.

S_1	S_2	A	B	G_1	G_2	$Output_1$	$Output_2$
0	0	0	0	-	-	0	0
0	0	0	1	-	-	0	1
0	0	1	0	-	-	1	0
0	0	1	1	-	-	1	1
0	1	0	0	-	-	-	0
0	1	0	1	-	-	-	1
0	1	1	0	-	-	-	1
0	1	1	1	-	-	-	1
1	0	0	0	-	-	-	0
1	0	0	1	-	-	-	0
1	0	1	0	-	-	-	0
1	0	1	1	-	-	-	1
1	1	0	0	-	-	0	0
1	1	0	1	-	-	0	1
1	1	1	0	-	-	0	1
1	1	1	1	-	-	1	0

Bibliography

- [1] M. M. Rahman, “Synthesis of Reversible Logic,” Ph.D. dissertation, Graduate Academic Unit of Faculty of Computer Science, The University of New Brunswick, December 2014.
- [2] R. Landauer, “Irreversibility and heat generation in the computing process,” *IBM J. Res.*, vol. 5, pp. 183–191, 1961.
- [3] The European Science Foundation, “Nanoscience and the long-term future of information technology,” December 2006. [Online]. Available: <http://www.esf.org/publications/forward-looks.html>
- [4] C. H. Bennett, “Notes on the history of reversible computation,” *IBM J. Res. Dev.*, vol. 32, no. 1, pp. 16–23, January 1988.
- [5] D. Maslov, “Reversible logic synthesis,” Ph.D. dissertation, The Faculty of Computer Science, The University of New Brunswick, Canada, 2003.
- [6] J. Donald and N. K. Jha, “Reversible logic synthesis with Fredkin and Peres gates,” *ACM Journal on Emerging Technologies in Computing Systems*, vol. 4, pp. 1–19, 2008.
- [7] D. Maslov and M. Saeedi, “Reversible circuit optimization via leaving the boolean domain,” *IEEE Trans. on CAD*, pp. 806–816, 2011.
- [8] M. Saeedi and I. L. Markov, “Synthesis and optimization of reversible circuits a survey,” *ACM Computing Surveys*, vol. 45, pp. 21–34, 2013.
- [9] D. Maslov and G. W. Dueck, “Garbage in reversible designs of multiple output functions,” *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technologies, Germany*, pp. 162–170, March 2003.

- [10] —, “Reversible cascades with minimal garbage,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 11, pp. 1497–1509, November 2004.
- [11] D. Miller, R. Wille, and R. Drechsler, “Reducing reversible circuit cost by adding lines,” *IEEE International Symposium on Multiple-Valued Logic*, pp. 217–222, 2010.
- [12] D. M. Miller, “Lower cost quantum gate realizations of multiple-control Toffoli gates,” *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 308–313, 2009.
- [13] D. M. Miller and Z. Sasanian, “Lowering the quantum gate cost of reversible circuits,” *53rd IEEE International Midwest Symposium on Circuits and Systems*, pp. 260–263, 2010.
- [14] M. Szykowski and P. Kerntopf, “Reducing quantum cost in reversible Toffoli circuits,” *Proc. 10th Reed-Muller Workshop*, pp. 127–136, 2011.
- [15] P. Kerntopf and M. Szykowski, “Reducing quantum cost of pairs of multi-control Toffoli gates,” *International workshop on Boolean Problems*, pp. 263–268, 2012.
- [16] D. Grosse, X. Chen, G. W. Dueck, and R. Drechsler, “Exact SATbased Toffoli network synthesis,” *Proc. 17th Great Lakes Symposium on VLSI, Italy*, pp. 96–101, 2007.
- [17] G. W. Dueck, D. Maslov, and D. M. Miller, “Techniques for the synthesis of reversible Toffoli networks,” *ACM Trans. on Design Automation of Electronic Systems*, vol. 12, pp. 1–28, 2007.
- [18] O. Golubitsky, S. M. Falconer, and D. Maslov, “Synthesis of the optimal 4-bit reversible circuits,” *Proc. 47th Design Automation Conference, USA*, pp. 653–656, June 2010.
- [19] Z. Sasanian and D. M. Miller, “Mapping a multiple-control Toffoli gate cascade to an elementary quantum gate circuit,” *Proc. 2nd Workshop on Reversible Computation*, pp. 83–90, June 2010.

- [20] ———, “Transforming MCT circuits to NCVW circuits in: A. De. Vos and R. Wille (Eds),” *Reversible Computation, RC 2011, LNCS, Springer-Verlag 2012, 7165*, pp. 77–88, 2011.
- [21] R. Wille, D. M. Miller, and Z. Sasanian, “Elementary quantum gate realizations for multiple-control Toffoli gates,” *Proc. 41st IEEE International Symposium on Multiple-Valued Logic*, pp. 288–293, 2011.
- [22] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [23] R. Wille and R. Drechsler, *Towards a design flow for reversible logic*. Springer, 2010.
- [24] D. Maslov and G. Dueck, “Garbage in reversible designs of multiple output functions,” *Proceedings of the 6th International Symposium on Representations and Methodology of Future Computing Technologies (RM 2003), Trier, Germany*, pp. 162–170, March 2003.
- [25] W. Hung, X. Song, G. Yang, and M. Perkowski, “Optimal synthesis of multiple output boolean functions using a set of quantum gates by symbolic reachability analysis,” *Transactions on Computer Aided Design*, vol. 25, no. 9, pp. 1652–1663, 2006.
- [26] J. Smolin and D. Divincenzo, “Five Two-bit Quantum Gates Are Sufficient to Implement the Quantum Fredkin Gate,” *Physical Review A*, vol. 53, pp. 2855–2856, 1996.
- [27] H. G. Rangaraju, U. Venugopal, K. Muralidhara, and K. B. Raja, “Low power reversible parallel binary adder/subtractor,” *International Journal of VLSI design and Communication System (VLSICS)*, vol. 1, no. 3, pp. 23–34, 2010.
- [28] R. P. Feynman, “Quantum mechanical computers,” *Optic News*, vol. 11, pp. 11–12, February, 1985.
- [29] S. Moghimi and M. R. Reshadinezhad, “A novel 4×4 universal reversible gate as a cost efficient full adder/subtractor in terms of reversible and quantum metrics,” *International Journal of Modern Education and Computer Science*, vol. 11, pp. 28–34, 2015. [Online]. Available: DOI: 10.5815/ijmecs.2015.11.04

- [30] S. Sultan and K. Radecka, "Reversible architecture of computer arithmetic," *International Journal of Computer Applications*, vol. 93, no. 14, May 2014.
- [31] M. Majid, 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, no. 6, pp. 787–792, 2008.
- [32] M. Haghparast and K. Navi, "A novel reversible BCD adder for nanotechnology based systems," *American Journal of Applied Sciences*, vol. 5, no. 3, pp. 282–288, 2008.
- [33] P. Gupta, A. Agrawal, and N. K. Jha, "An algorithm for synthesis of reversible logic circuits," *IEEE Trans. on CAD of Integrated Circuits and Sys.*, vol. 25, no. 11, pp. 2317–2330, 2006.
- [34] "RevLib: An online resource for reversible functions and circuits." [Online]. Available: <http://revlib.org/>
- [35] O. Golubitsky and D. Maslov, "A study of optimal 4-bit reversible Toffoli circuits and their synthesis," *IEEE Trans. Computers*, vol. 61, no. 9, pp. 1341–1353, September 2012.
- [36] M. Morrison and N. Ranganathan, "Design of reversible ALU based on novel programmable reversible logic gate structures," *IEEE Computer Society Annual Symposium on VLSI*, pp. 126–131, 2011.
- [37] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997.
- [38] M. K. Thomsen, R. Gluck, and H. B. Axelsen, "Reversible arithmetic logic unit for quantum arithmetic," *J. Phys. A : Math. Theor.*, vol. 43, no. 38, 2010.
- [39] R. Aradhaya, K. N. Muralidhara, and B. Kumar, "Design of low power arithmetic unit based on reversible logic," *International Journal of VLSI and Signal Processing Applications*, vol. 1, no. 1, pp. 30–38, 2011.
- [40] B. K. Sikdar, "Design of fault tolerant reversible arithmetic logic unit in QCA," *International Symposium on Electronic System Design*, 2012.

- [41] M. H. A. Khan, "Quantum realization of ternary adder circuits," *Proceedings of Third International Conference on Electrical and Computer Engineering*, pp. 249–252, 2004.
- [42] Y. V. Rentergem and A. D. Vos, "Optimal design of a reversible full adder," *International Journal of Unconventional Computing*, vol. 1, pp. 339–355, 2005.
- [43] H. Thapliyal and A. P. Vinod, "Transistor realization of reversible TSG gate and reversible adder architectures," *Proceedings of IEEE Asia Pacific Conference on Circuits and Systems*, pp. 418–421, 2006.
- [44] H. Thapliyal and M. B. Srinivas, "Novel design and reversible logic synthesis of multiplexer based full adder and multipliers," *Forty Eight Midwest Symposium on Circuits and Systems*, vol. 2, pp. 1593–1596, 2006.
- [45] R. K. James, T. K. Shahana, K. P. Jacob, and S. Sasi, "A new look at reversible logic implementation of decimal adder," *International Symposium on System-On-Chip*, pp. 1–4, 2007.
- [46] L. Ni, Z. Guan, and W. Zhu, "A general method of constructing the reversible full-adder," *Third International Symposium on Intelligent Information Technology and Security Informatics*, pp. 109–113, 2010.
- [47] V. Kamalakannan, V. Shilpakala, and H. N. Ravi, "Design of adder/subtractor circuit based on reversible gates," *International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering*, vol. 2, no. 8, August 2013.
- [48] J. Kaur and H. Kaur, "Synthesis and designing of reversible adder/subtractor circuits," *International Journal of Advanced Research in Electrical Electronics and Instrumentation engineering*, vol. 3, no. 5, 2014.
- [49] C. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, pp. 525–532, 1973.
- [50] M. Perkowski, P. Kerntopf, A. Buller, M. Chrzanowska-Jeske, A. Mishchenko, X. Song, A. Al-Rabadi, L. Jozwiak, A. Coppola, and B. Massey, "Regular realization of symmetric functions using reversible logic," *Proceedings of EURO-MICRO Symposium on Digital Systems Design (Euro-Micro'01)*, pp. 245–252, September 2001.

- [51] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*. Oxford University Press Inc., Oxford, 2007.
- [52] A. Barenco, C. H. Bennett, R. Cleve, D. P. Divincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, “Elementary gates for quantum computation,” *Physical Review A*, vol. 52, no. 5, pp. 3457–3467, 1995.
- [53] T. Toffoli, “Reversible computing,” *Tech memo MIT/LCS/TM-151, MIT Lab for Comp. Sci.*, 1980.
- [54] E. Fredkin and T. Toffoli, “Conservative logic,” *International Journal of Theoretical Physics*, vol. 21, pp. 219–253, 1982.
- [55] A. Peres, “Reversible logic and quantum computers,” *Phys. Rev. A*, vol. 32, no. 6, pp. 3266–3276, December 1985.
- [56] D. Maslov and G. W. Dueck, “Reversible cascades with minimal garbage,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 23, no. 11, pp. 1497–1509, 2004.
- [57] D. Grosse, R. Wille, G. W. Dueck, and R. Drechsler, “Exact multiple-control Toffoli network synthesis with SAT techniques,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 28, no. 5, pp. 703–715, 2009.
- [58] O. Golubitsky, S. M. Falconer, and D. Maslov, “Synthesis of the optimal 4-bit reversible circuits,” *Proceedings of the 47th Design Automation Conference, ser. DAC’10. New York, NY, USA: ACM*, pp. 653–656, 2010.
- [59] A. Mishchenko and M. Perkowski, “Logic synthesis of reversible wave cascades,” *International Workshop on Logic Synthesis*, June, 2002.
- [60] D. M. Miller, D. Maslov, and G. W. Dueck, “A transformation based algorithm for reversible logic synthesis,” *Design Automation Conference*, pp. 318–323, June, 2003.
- [61] K. Iwama, Y. Kambayashi, and S. Yamashita, “Transformation rules for designing CNOT-based quantum circuits,” *Design Automation Conference, New Orleans, Louisiana, USA*, pp. 10–14, June 2002.
- [62] D. M. Miller, D. Maslov, and G. W. Dueck, “Toffoli network synthesis with templates,” *Transactions on Computer Aided Design*, vol. 24, no. 6, pp. 807–817, 2005.

- [63] D. Grosse, X. Chen, and R. Drechsler, “Exact Toffoli network synthesis of reversible logic using boolean satisfiability,” *IEEE Dallas/CAS Workshop*, pp. 51–54, 2006.
- [64] D. Grosse, X. Chen, G. W. Dueck, and R. Drechsler, “Exact SAT-based Toffoli network synthesis,” *Proc. 17th Great Lakes Symposium on VLSI, Italy*, pp. 96–101, 2007.
- [65] R. Wille and R. Drechsler, “BDD-based synthesis of reversible logic for large functions,” *Design Automation Conference*, pp. 270–275, 2009.
- [66] —, “Effect of BDD optimization on synthesis of reversible and quantum logic,” *Electronic Notes in Theoretical Computer Science, Proceedings of the Workshop on Reversible Computation (RC 2009)*, vol. 253, no. 6, pp. 57–70, 2010.
- [67] M. Soeken, R. Wille, G. W. Dueck, and R. Drechsler, “Window optimization of reversible and quantum circuits,” *International Symposium on Design and Diagnostics of Electronic Circuits and Systems*, 2010.
- [68] D. M. Miller, R. Wille, and Z. Sasanian, “Elementary quantum gate realizations for multiple-control Toffoli gates,” *Proceedings of the International Symposium on Multiple-Valued Logic*, pp. 288–293, 2011.
- [69] 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*, vol. 27, no. 3, pp. 436–444, 2008.
- [70] D. Maslov, C. Young, G. W. Dueck, and D. M. Miller, “Quantum circuit simplification using templates,” *DATE–Design, Automation and Test in Europe*, pp. 1208–1213, 2005.
- [71] N. Scott and G. Dueck, “Pairwise Decomposition of Toffoli Gates in a Quantum Circuit,” *Great Lakes Symposium on VLSI*, pp. 231–235, 2008.
- [72] P. Kerntopf and M. Szyprowski, “Reducing Quantum Cost of Pairs of Multi-Control Toffoli Gates,” *International workshop on Boolean Problems*, pp. 263–268, 2012.
- [73] A. D. Vos, *Reversible Computing*. Wiley-VCH Verlag, 2010.

- [74] M. Saeedi and I. L. Markov, "Synthesis and Optimization of Reversible Circuits A Survey," *ACM Computing Surveys*, pp. 21–34, 2013.
- [75] D. M. Miller, "Lower Cost Quantum Gate Realizations of Multiple-Control Toffoli Gates," *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 308–313, 2009.
- [76] D. M. Miller and Z. Sasanian, "Lowering the Quantum Gate Cost of Reversible Circuits," *IEEE International Midwest Symposium on Circuits and Systems*, pp. 260–263, 2010.
- [77] Z. Sasanian, "Technology Mapping and Optimization for Reversible and Quantum Circuits," Ph.D. dissertation, Department of Computer Science, University of Victoria, Victoria, BC, Canada, 2012.
- [78] Z. Sasanian and D. M. Miller, "Transforming MCT Circuits to NCVW Circuits, in: A. De. Vos and R. Wille (Eds)," *Reversible Computation, RC 2011, LNCS, Springer-Verlag*, vol. 7165, pp. 77–88, 2011.
- [79] —, "Reversible and Quantum Circuit Optimization: A Functional Approach," *Reversible Computation, RC 2012, LNCS, Springer-Verlag*, vol. 7581, pp. 112–124, 2013.
- [80] M. Szyprowski and P. Kerntopf, "Reducing Quantum Cost in Reversible Toffoli Circuits," *Proc. 10th Reed- Muller Workshop*, pp. 127–136, 2011.
- [81] —, "A Study of Optimal 4-bit Reversible Circuit Synthesis from Mixed-Polarity Toffoli Gates," *Proc. 12th IEEE Conference on Nanotechnology*, pp. 1–6, 2012.
- [82] D. Maslov and G. W. Dueck, "Improved Quantum Cost for n -bit Toffoli Gates," *IEEE Electronic Letters*, vol. 39, pp. 1790–1791, 2003.
- [83] G. Dueck, D. Maslov, and D. Miller, "Techniques for the Synthesis of Reversible Toffoli Networks," *ACM Trans. On Design Automation of Electronic Systems*, vol. 12, pp. 1–28, 2007.
- [84] D. Maslov, "Reversible Logic Synthesis Benchmarks Page," 2009. [Online]. Available: <http://www.csc.uvic.ca/dmaslov>

- [85] J. Donald and N. K. Jha, "Reversible Logic Synthesis with Fredkin and Peres Gates," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 4, pp. 1–19, 2008.
- [86] G. W. Dueck, R. Wille, H. M. Le, and D. Grosse, "Quantified Synthesis of Reversible Logic," *Proc. DATE*, pp. 1015–1020, 2008.
- [87] Z. Sasanian and D. M. Miller, "Mapping a Multiple-Control Toffoli Gate Cascade to an Elementary Quantum Gate Circuit," *Proc. 2nd Workshop on Reversible Computation*, pp. 83–90, 2010.
- [88] D. Maslov and M. Saeedi, "Reversible circuit optimization via leaving the Boolean domain," *IEEE Trans. on CAD*, pp. 806–816, 2011.
- [89] R. Jozsa, "Entanglement and Quantum Computation," *arXiv*, pp. 11–20, 1997. [Online]. Available: <http://arxiv.org/abs/quant-ph/9707034>
- [90] A. Ekert and R. Jozsa, "Quantum algorithms: Entanglement-enhanced information processing," *The Geometric Universe: Science, Geometry, and the Work of Roger Penrose*. University Press, pp. 1769–1782, 1998.
- [91] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, 1994.
- [92] C. H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels," *Physical Review Letters*, vol. 70, no. 13, pp. 1895–1899, 1993.
- [93] C. Bennett and S. Wiesner, "Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states," *Physical Review Letters*, vol. 69, no. 20, pp. 2881–2884, 1992.
- [94] P. W. Shor, "Scheme for reducing decoherence in quantum computer memory," *Physical Review A*, vol. 52, no. 4, pp. 2493–2496, 1995.
- [95] A. K. Ekert, "Quantum cryptography based on Bell's theorem," *Physical Review Letters*, vol. 67, no. 6, pp. 661–663, 1991.

- [96] C. H. Bennett, G. Brassard, and N. D. Mermin, "Quantum cryptography without Bell's theorem," *Physical Review Letters*, vol. 68, no. 5, pp. 557–559, 1992.
- [97] E. Biham, B. Huttner, and T. Mor, "Quantum cryptographic networks based on quantum memories," *Physical Review Letters*, vol. 54, no. 4, pp. 2651–2658, 1996.
- [98] M. L. Horodeck, "Entanglement Measures," *Quantum Information and Computation*, vol. 1, no. 1, pp. 3–26, 2001.
- [99] A. Ezhov, "Role of interference and entanglement in quantum neural processing," *Proceedings- Spie The International Society For Optical Engineering*, pp. 367–379, 2001.
- [100] W. S. Frederick, R. J. Philip, J. D. Alex, C. Lobb, J. Anderson, and F. Wellstood, "Quantum logic gates for coupled superconducting phase qubits," *Phys. Rev. Lett.*, vol. 91, pp. 167 005–167 009, 2003.
- [101] Arvind, "Quantum entanglement and quantum computational algorithms," *Pramana–Journal of Physics*, vol. 56, no. 2-3, pp. 357–365, 2001.
- [102] K. Dorai, Arvind, and A. Kumar, "Implementation of a Deutsch-like quantum algorithm utilizing entanglement at the two-qubit level on an NMR quantum information processor," *Phys. Rev. A*, vol. 63, pp. 101–105, 2000.
- [103] S. Lloyd, "Quantum search without entanglement," *Phys. Rev. A*, vol. 61, pp. 301–305, 2000.
- [104] W. Wootters and W. Zurek, "A single quantum cannot be cloned," *Nature*, vol. 299, no. 5886, pp. 802–803, 1982.
- [105] D. Dieks, "Communication by EPR devices," *Physics Letters A*, vol. 92, no. 6, pp. 271–272, 1982.
- [106] N. Abdessaied, M. Soeken, and R. Drechsler, "Quantum circuit optimization by Hadamard gate reduction," *Reversible Computation–series: Lecture Notes in Computer Science*, vol. 8507, pp. 149–162, 2014.
- [107] M. Perkowski, M. Lukac, D. Shah, and M. Kameyama, "Synthesis of quantum circuits in linear nearest neighbor model using positive Davio lattices," *Facta universitatis–series: Electronics and Energetics*, vol. 4, no. 1, pp. 73–89, 2011.

- [108] M. Saeedi, R. Wille, and R. Drechsler, "Synthesis of quantum circuits for linear nearest neighbor architectures," *Quantum Information Processing*, vol. 10, no. 1, pp. 73–89, 2011.
- [109] R. M. Mazder, A. Banerjee, G. W. Dueck, and A. Pathak, "Two-Qubit Quantum Gates to Reduce the Quantum Cost of Reversible Circuit," *41st IEEE International Symposium on Multiple-Valued Logic*, pp. 86–92, 2011.
- [110] R. Wille, D. Miller, and R. Drechsler, "Reducing reversible circuit cost by adding lines," *IEEE International Symposium on Multiple-Valued Logic*, pp. 217–222, 2010.

Index

- 2-qubit gate rule, 13, 44, 47
- adder, 26
- adder/subtractor, 3, 17, 19, 21, 41
- additional line, 20
- adjacent gate, 12, 41
- ALU, 3, 4, 17, 19, 21, 33, 38, 44, 48
- analogous, 3
- ancilla input, 9
- ancilla line, 4, 20
- AND gate, 1, 8
- arbitrary qubit, 10
- arithmetic logic unit, 3
- arithmetic/logic operation, 38
- benchmark, 33
- binary, 7
- binary reversible function, 11
- bit, 3
- black dot, 9
- Boltzmann, 1
- boolean logic function, 3, 7
- boolean variable, 4, 20
- Carry/Borrow, 26
- cascading, 3, 14
- circuit synthesis, 41
- classical bit, 11
- classical computation, 10
- classical reversible function, 8
- CNOT gate, 3, 23, 31
- completely specified reversible function, 19
- complex element, 3, 11
- complex vector space, 3
- computational basis state, 3
- computing device, 1
- computing machine, 3
- conjugacy class, 24
- constant input, 2, 4, 9, 10, 17, 26, 37, 50
- constraint, 10
- control line, 9, 12, 29
- Controlled- V^\dagger , 3
- Controlled-NOT, 9
- Controlled-V, 3, 11
- Controlled- V^\dagger , 11
- cost factor, 3
- cost metric, 2
- cost-efficient, 2, 17
- decomposition, 42
- deletion rule, 12
- design flow, 2, 18
- digital circuit, 11
- dissipate, 1
- don't-care, 20
- elementary gate, 11
- elementary quantum gate, 2, 3, 44
- embedding, 3, 4, 19
- energy loss, 2
- enumerating, 25
- exact synthesis, 41
- exponential algorithm, 3

extra output, 10
 extra-working line, 10
 factorization problem, 3
 fan-out, 10
 fault-tolerant quantum circuit, 50
 feedback, 10
 Feynman gate, 17
 Fredkin gate, 4, 13, 17
 full adder/subtractor, 4, 30, 37, 44, 45
 function design, 4
 function specification, 44
 garbage output, 2, 4, 9, 10, 17, 20, 37, 50
 gate count, 24
 gate library, 17, 18, 45, 50
 general-purpose quantum computer, 3
 greedy algorithm, 5, 41, 43, 44
 Gupta's-LU, 34, 48
 half adder/subtractor, 29, 36, 44
 hash table, 25
 heuristic method, 33
 heuristic search technique, 41
 identity function, 12, 43
 incompletely-specified function, 4, 49
 incompletely-specified reversible function,
 4, 19
 information loss, 1, 2
 input line, 50
 input state, 2
 integer factoring, 2
 integrated qubit gate, 13
 irreversible, 1, 5
 irreversible function, 3, 4, 19
 ISRF, 20, 22, 25
 line reordering, 24
 linear ion Trap, 50
 linear nearest neighbor, 50
 LNN, 50
 logic expression, 30
 logic function, 1, 7
 logic gate, 1, 8
 logic operation, 3, 8
 logic synthesis, 1, 4
 logic unit, 17
 logical, 2
 logical operation, 1
 low-power computing, 3
 lower power consumption, 4, 17
 lower-level logic function, 7
 LU, 38
 many-to-one, 7
 mapping, 3, 9
 MATLAB, 44
 MCF, 9
 MCT gate, 2, 20
 MCT gate library, 4, 18
 merging rule, 12, 43
 metric, 2
 Mini-ALU, 17, 34, 47
 minimality, 20
 minimization algorithm, 5, 25, 36, 38, 43,
 47, 49, 50
 minimization problem, 18
 minimize, 2
 minimum MCT circuit, 23, 25
 Moore's Law, 1
 moving rule, 5, 12, 41
 multiple input, 1
 multiple output boolean logic function, 3
 multiple qubit, 41

multiple-control Fredkin, 9
 multiple-control Toffoli, 2, 9
 multiple-output function, 21

 nanoscale, 1
 NCV, 3, 42
 NCV gate library, 11
 NMR, 50
 NOT gate, 1, 8, 23
 nuclear magnetic resonance, 50

 OA, 21, 26, 35
 OA-equivalence class, 49
 OA-equivalent, 21, 23
 OAEC, 25
 one-to-one, 2, 7
 operation, 21
 operation assignment, 4, 19, 49, 50
 optimization, 4
 optimization approaches, 5
 OR gate, 1, 8
 output, 2
 output pattern, 20
 output permutation, 35
 output state, 2

 P-equivalence, 24
 P-equivalence class, 29, 49
 PEC, 25
 Peres gate, 4, 17
 permutation, 4, 5, 8, 19, 21, 23, 49
 polynomial time, 3
 primitives, 1, 7
 programmable computing device, 3, 17

 QC, 2, 14, 17, 23, 41
 quantum circuit, 3, 14, 18, 41
 quantum circuit synthesis, 2

 quantum computation, 3, 10, 14
 quantum computer, 38, 50
 quantum cost, 2, 13, 50
 quantum gate, 3, 11, 14, 44
 quantum gate library, 3
 quantum mechanic, 3
 quantum metrics, 17
 quantum operation, 3
 quantum optic, 50
 quantum primitive, 11
 quantum system, 3
 qubit, 3, 10

 reversibility, 2, 10
 reversible, 1, 2
 reversible adder/subtractor, 17
 reversible circuit, 2, 3, 10, 20
 reversible circuit synthesis, 2
 reversible computation, 2
 reversible computing, 1, 3
 reversible function, 2–4, 8, 10, 18–20, 37
 reversible gate, 2, 17, 37
 reversible logic, 2, 17, 38
 reversible logic synthesis, 19, 50
 reversible metrics, 17
 reversible synthesis, 2, 18
 reversible systems, 2
 Replib, 17
 rolf Landauer's principle, 1
 rows, 21

 SAT-based algorithm, 25, 26
 selector bit, 21, 22, 33
 self-inverse gate, 12
 set, 7
 shannon expansion, 21
 single output, 1, 8

single qubit, 41
splitting rule, 12
square-root-of-NOT, 12
state vector, 3, 10
subtractor, 26
Sum/Difference, 26
superposition, 3, 11
SWOP, 5, 23, 49
symmetric pattern, 13, 47
synthesis, 1
synthesis method, 3, 19

target line, 9, 29
technology, 1
thermodynamic, 1
Toffoli, 3
Toffoli gate, 4, 31
Toffoli-3 gate, 14
Toffoli-3 gates, 23
top-level design, 7
TR gate, 17
transformation, 14, 42
transformed, 2
transistor, 1
truth table, 4, 20, 24, 26
two-dimensional, 3

unique element, 7
unitary matrice, 3
unitary matrix, 11
unitary operation, 3, 41
unitary transformation, 3
universal, 3
universal reversible gate, 17

vector, 2
vector space, 3
working line, 4, 17