# Technology Mapping and Optimization for Reversible and Quantum Circuits

by

Zahra Sasanian
B.Sc., University of Tehran, 2006
M.Sc., Amirkabir University of Technology, 2008

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Technology Mapping and Optimization for Reversible and Quantum Circuits

by

Zahra Sasanian
B.Sc., University of Tehran, 2006
M.Sc., Amirkabir University of Technology, 2008

Supervisory Committee

---

Dr. D. Michael Miller, Supervisor
(Department of Computer Science)

---

Dr. Jon C. Muzio, Departmental Member
(Department of Computer Science)

---

Dr. Kin Fun Li, Outside Member
(Department of Electrical and Computer Engineering)

**Supervisory Committee**

---

Dr. D. Michael Miller, Supervisor
(Department of Computer Science)

---

Dr. Jon C. Muzio, Departmental Member
(Department of Computer Science)

---

Dr. Kin Fun Li, Outside Member
(Department of Electrical and Computer Engineering)

## ABSTRACT

Quantum information processing is of interest as it offers the potential for a new generation of very powerful computers supporting novel computational paradigms. Over the last couple of decades, different aspects of quantum computers ranging from quantum algorithms to quantum physical design have received growing attention. One of the most important research areas is the synthesis and post-synthesis optimization of reversible and quantum circuits. Many synthesis and optimization approaches can be found in the literature, yet, due to the complexity of the problem, finding approaches leading to optimal, or near optimal, results is still an open problem. The synthesized circuits are usually evaluated based on quantum cost models. Therefore, they are often technology mapped to circuits of more primitive gates. To this end, various technology mapping approaches have also been proposed in the past few years.

Related work shows an existing gap in optimized technology mapping for reversible and quantum circuits. In this dissertation, an optimized technology mapping design flow is introduced for mapping reversible circuits to quantum circuits. The contributions of this dissertation are classified as follows:

- New reversible circuit optimization methods.
- Optimized reversible to quantum mapping approaches.

- New quantum gate libraries and new cost models for reversible gates based on the new libraries.
- Quantum circuit optimization approaches.

The steps above, form an optimized flow for mapping reversible circuits to quantum circuits. At each step of the design flow optimized and consistent approaches are suggested with the goal of reducing the quantum cost of the synthesized reversible circuits. The evaluations show that the proposed mapping methodology leads to significant improvement in the quantum cost of the existing benchmark circuits.

# Contents

# List of Tables

# List of Figures

# List of Acronyms

**BDD** Binary Decision Diagram

**CMOS** Complementary Metal-Oxide-Semiconductor

**CNOT** Controlled-NOT

**DDMF** Decision Diagrams for a Matrix Function

**GC** Gate Count

**GRP** Gate Reduction Procedure

**LLP** Line Labeling Procedure

**MCT** Multiple-Control Toffoli

**MPMCT** Mixed Polarity Multiple-Control Toffoli

**NCV** NOT, CNOT, controlled-$V$, and controlled-$V^+$ gate library

**NCVW** NOT, CNOT, controlled-$V$, and controlled-$V^+$, controlled-$W$, and controlled-$W^+$ gate library

**NMR** Nuclear Magnetic Resonance

**PPRM** Positive Polarity Reed Muller

**Qubit** Quantum Bit

**QMDD** Quantum Multiple-valued Decision Diagram

**REVLIB** Reversible Library [9]

**ROBDD** Reduced Ordered Binary Decision Diagram

**SAT** Boolean Satisfiability

**SCQC** Semi-Classical Quantum Circuits

# ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude and appreciation to my supervisor, Prof. D. Michael Miller, for his continuous support, encouragement and patience. If it was not for his invaluable guidance, I would not have gone half of this way. It was a pleasure and honor for me to work under his supervision and to learn from him. I would also like to thank my committee members and my external examiner, Dr. Jon Muzio, Dr. Kin Fun Li and Dr. Alex De Vos, for their time and their constructive comments and suggestions.

I would like to acknowledge my colleagues in Bremen, Robert Wille and Mathias Soeken, for their cooperation in the papers that contributed to this dissertation. I would also like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for funding this work.

Last but not least, I would like to thank my parents, siblings and my husband for their constant support and encouragement.

*Science never solves a problem without creating ten more.*
George Bernard Shaw

# DEDICATION

To my Father and my Mother, may her soul rest in peace.

# Chapter 1

# Introduction

Over the last few decades, the processing power of computers has steadily increased due to the increasing density of transistors on chips, which has been following Moore's law [10] for about half a century. However, this trend cannot continue forever because downsizing the current CMOS process technology will eventually face the fundamental physical limits of atomic structures. As of 2012, leading companies are building devices based on 20nm technology. The 14nm technology is estimated to be reached by semiconductor companies by 2014. However, downsizing transistors will hit the fundamental barriers (such as quantum tunneling) eventually regardless of the materials used. Consequently, the increasing demand for higher processing power and lower power consumption has led researchers to seek alternatives to classical computing.

One of the more promising alternatives appears to be quantum computing which is believed to have a great ability to parallelize information processing. Quantum computing is concerned with processing information using quantum mechanical systems. Quantum mechanics is a mathematical framework for the construction of new physical theories describing the dual particle-like and wave-like behavior and interactions of energy and matter. Various quantum technologies have been introduced so far *e.g.* solution and silicon NMR systems in which information is kept in the spin of the nucleus of an atom, ion trap systems where the information is usually stored in the energy levels of individual ions, and optical quantum systems that use different polarizations of photons to represent information.

In 1961, Rolf Landauer [11] showed that erasing a bit of information leads to a power dissipation of at least $kTln2$ where $k$ is the Boltzmann constant and $T$ is the absolute temperature of the environment. However, the actual amount of heat generated in real systems is far beyond this ideal limit. In irreversible computation,

information is lost because it is not always possible to identify a unique input pattern by knowing the output pattern. Most of the computations in current circuits are irreversible and thus traditional chips suffer from the power dissipation caused by loosing information. In 1973, Charles H. Bennett [12] showed that energy dissipation is reduced or even eliminated if computation is information-lossless.

Since information is not lost during a reversible operation, reversible circuits offer the potential to significantly reduce the power consumption [13–15]. Reversible circuits are composed of reversible gates that are applied to a set of inputs consecutively. Reversible gates have the same number of inputs and outputs and implement permutation functions. In 2002, CMOS reversible circuits were built to exploit this observation [16]. The only source of power in the examined circuits was their input signals.

Reversible computing has application in various domains such as Optical Computing [17, 18], DNA Computing [19, 20], Nanotechnologies [21], and Cryptography [22]. One of the most important applications of reversible logic is in quantum computing [23]. Quantum circuits are inherently reversible. The physical structure of quantum circuits is completely different from that of a classical circuit in that a set of reversible operations are applied to a set of quantum particles holding quantum states rather than the transistors and wires that are used in classical circuits.

Quantum computers are good candidates to be the next generation of computers. They will be more powerful than classical computers and are believed to have the ability to solve intractable problems that cannot be solved by the current classical computers due to the speed-up provided by quantum algorithms. For example Shor's factorization algorithm [24] has polynomial time complexity while classical factorization algorithms require exponential time. Quantum computers provide the required physical infrastructure to implement quantum algorithms.

The reversible and quantum design flow is quite similar to classical design automation flows and consists of different levels of abstraction from high level descriptions of algorithms at the highest level to the technology dependant description of circuits which describes physical operations at the lowest level. Synthesis of reversible and quantum circuits is one of the most important steps in the quantum design automation flow and has been extensively studied in the past few years [25–34]. A reversible or quantum synthesizer maps an input specification to its corresponding gate level description that is usually technology independent. Synthesis approaches use different representations of the input function as their intermediate format e.g. truth

tables, matrices, decision diagrams, Positive Polarity Reed Muller (PPRM) forms [4], cycle-based representations [5], etc. The target gate level description contains gates that belong to a reversible or quantum library and may be subsequently technology mapped to more elementary quantum gates that are further directly implemented.

The synthesis of quantum circuits is significantly more complex than is the case for traditional irreversible logic circuits because of its extended search space, being non-trivial what cost function will consistently move us towards an optimal solution, and various constraints imposed by the reversibility property. For example, feedbacks and fan-outs are not allowed in quantum circuits and each gate must have the same number of inputs and outputs [13]. As a result, brute-force search is considered to be impossible for functions with more than 4 variables [30]. The only exact solutions proposed for this problem are synthesis approaches based on *Boolean Satisfiability (SAT)* [25, 35, 36]. The main drawback of the SAT-based synthesis approaches is that the computational time of such methods is so high that these approaches can only be used to synthesize functions with a small number of variables and gates. For larger functions, other heuristic methods can be found in the literature such as transformation-based synthesis approaches [37–40], evolutionary algorithms [41, 42], decision diagram-based synthesis methods [43, 44], etc. Circuits resulting from heuristic synthesis approaches are not usually minimal cost and need to be optimized. Although various optimization approaches can be found in the literature, none of the existing methods guarantees optimality of the results.

In addition, the circuits resulting from synthesis are frequently mapped to circuits of elementary quantum gates. This mapping is performed to calculate the quantum cost of reversible circuits (as the number of elementary quantum gates in their realizations) or for physical implementation of reversible circuits in quantum technologies. The technology mapping approaches that can be found in the literature are not optimized and introduce even more redundancies while propagating the ones existing in the synthesized circuit. Many of the generated redundancies in technology mapping approaches come from the non-optimal cost models that are used. So far, no optimized flow for technology mapping reversible circuits to quantum circuits exists in the literature and an important gap exists in the existing reversible/quantum design flows [45].

The main focus of this dissertation is on technology mapping and optimization of reversible and quantum circuits. A design flow is suggested for technology mapping reversible circuits to quantum circuits and different approaches are introduced at

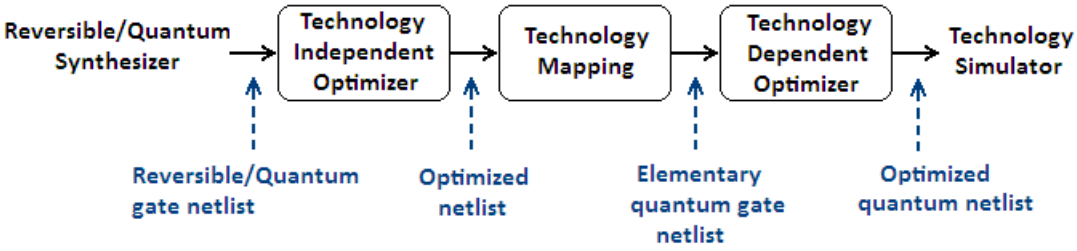



Figure 1.1: Reversible/Quantum technology mapping design flow

each step of the proposed flow. The proposed flow consists of three major steps as illustrated in Figure 1.1.

The first step is the high level optimization in which a circuit resulting from a synthesis approach is optimized most often in terms of the number of gates or the number of lines (the number of qubits of the system). The existing approaches for this part are usually heuristic and employ certain techniques to find local optimizations. Usually, reductions are found based on gate rearrangement. At this stage, we introduce a new moving rule for gate rearrangement that enables reversible gates to move beyond the constraints of the existing moving rules. Moreover, we utilize the new moving rule in our new optimization techniques that reduce the quantum cost of reversible circuits.

At the second level, the reversible circuit resulting from the post-synthesis optimizations is technology mapped to a target quantum library using a quantum cost model. Previous work on this step usually includes direct mapping of individual gates to their realizations based on an identified cost model and there is no approach that considers optimized mapping. Optimized mapping approaches depend highly on the quantum cost model that is used. In this dissertation, we propose a general approach for optimized mapping reversible gates to quantum gates. Furthermore, we introduce new quantum gate libraries and the corresponding cost models that reduce the quantum cost of individual reversible gates. We also discuss the application of the proposed mapping on the existing and new target quantum gate libraries.

Finally, in the last stage of the flow, the quantum netlist is optimized based on the number of elementary quantum gates or an identified technology dependant cost metric *e.g.* linear nearest neighbor architectures [46, 47]. We propose new optimizations based on our new moving rule for quantum circuits at this level.

The remainder of this dissertation is structured as follows.

**Chapter 2** gives a more detailed introduction to reversible and quantum logic and provides the required background necessary for this dissertation.

**Chapter 3** is devoted to the first and the last steps of the proposed mapping flow which are optimizations. The new optimization approaches at both the reversible and the quantum circuit levels are discussed in this chapter.

**Chapter 4** gives an alternative functional approach to the optimizations proposed in Chapter 3.

**Chapter 5** describes the improvements that are made in this dissertation on the existing quantum cost models. In addition, new quantum gate libraries are also introduced in this chapter and the corresponding cost models for reversible gates in the proposed libraries are discussed in detail.

**Chapter 6** describes the proposed optimized mapping methods for mapping reversible circuits to quantum circuits in different libraries.

**Chapter 7** concludes the discussions and provides directions for possible future work.

# Chapter 2

# Preliminaries

This chapter provides a brief summary of basic concepts in quantum mechanics and the background required for this dissertation. The chapter consists of three parts. The first section introduces the basic definitions and notations of quantum mechanics. For more information about quantum mechanics, see [13, 23]. The second section gives the background on reversible and quantum circuits required for this dissertation. The third section introduces decision diagrams that are used in the approaches described later in this dissertation. More information is available in [48–50].

## 2.1 Basic Concepts of Quantum Mechanics

Quantum computing deals with quantum information processing and follows the laws of quantum mechanics. In this section, the fundamental concepts of quantum mechanics describing quantum states and their characteristics are briefly introduced. A few quantum technologies are also introduced to give the reader an insight into quantum physical realization models.

### 2.1.1 Quantum States

In quantum information theory, information is encoded in quantum characteristics of materials in the form of *quantum states*. Quantum computing is concerned with the transmission and processing of quantum states. The basic element used to store information in classical computing is a bit, which can be in one of the two classical states 0 or 1. The analogous basic element in quantum computing is called a *quantum bit (qubit)*. In contrast to a classical bit, a qubit is able to store a continuum of

quantum states containing 0, 1 and certain linear combinations of the two basis states 0 and 1 (often called *superposition*). Qubits are denoted by Dirac notation $|\rangle$, also called *ket* [51]. Equation 2.1 shows the quantum state of a qubit as a superposition of the two quantum basis states $|0\rangle$ and $|1\rangle$.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

The $\alpha$ and $\beta$ coefficients are complex numbers. The $\alpha$ and $\beta$ values cannot be determined by examining the state of a qubit. Instead, after measuring a qubit, the result $|0\rangle$ is obtained with a probability of $|\alpha|^2$ and the result $|1\rangle$ is obtained with a probability of $|\beta|^2$. Hence, $|\alpha|^2 + |\beta|^2 = 1$. Thus, as shown in Equation 2.2 we use a unit vector to indicate a qubit's state in a two-dimensional complex vector space, $\mathbb{C}^2$. Special states $|0\rangle$ and $|1\rangle$ form an orthogonal basis for this two-dimensional vector space and are called computational basis states [13].

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.2}$$

The state of a qubit can also be written as shown in Equation 2.3. This equation shows the state of a qubit as a point on the surface of a sphere called the Bloch sphere [13]. The Bloch sphere is illustrated in Figure 2.1.

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{2.3}$$

A qubit can have an infinite number of values corresponding to every point on the sphere, but the outcome of measuring a qubit is one of the basis states $|0\rangle$ and $|1\rangle$, and after measurement the state of a qubit falls into one of the basis states. Thus, in order to use the whole domain of values that a qubit can store, it should not be measured inside a quantum system. However, the continuous values are evolved in a closed system complying with the laws of quantum mechanics.

The process of disturbing a quantum state through the interaction with environment is called *decoherence*. Decoherence is one of the main problems in designing quantum circuits [13].

### 2.1.2 Quantum Technologies

A qubit can be realized in several ways, for instance, as the two different polarizations of a photon, as the alignment of a nuclear spin in a uniform magnetic field, or as two

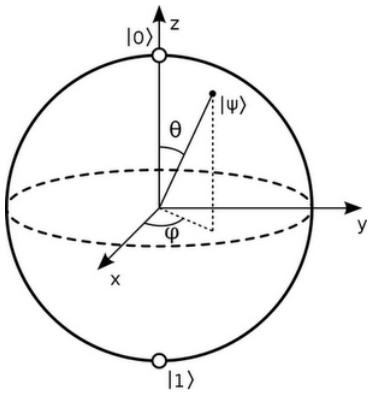



Figure 2.1: Bloch sphere representing the state of a qubit

states of an electron orbiting a single atom.We here briefly introduce some established quantum technologies. For a more complete review please refer to [52].

***Nuclear Magnetic Resonance (NMR) Systems:*** In NMR systems, a qubit is represented by the spin of the nucleus of an atom. In solution NMR [53, 54], many copies of a molecule are held in a liquid solution as a canonical ensemble system meaning that they all receive the same operators and execute the same program on the same data. With the large numbers of molecules, adequate signal strength for readout is provided. In silicon NMR systems [55], qubits are stored in the nuclear spin of silicon atoms laid down on a nuclear-spin-free substrate. In this technology, gates are applied via microwave radiations directed at the device. A micro magnet provides a high field gradient allowing individual atoms to be addressed by frequency. The strength of this physical model is that it has the longest known decoherence time [52].

***Ion Trap Technology:*** In ion trap systems, information is stored in the energy levels of individual ions [56]. In this technology, a single ion is adequate to keep the state of a qubit. In order to apply gates, qubits are moved to locations designed for gate operations using magnetic fields. Gates are applied via laser pulses. Readout is also accomplished by laser pulses creating fluorescence (interpreted as a 1) or not (interpreted as a 0) depending on the state of the atom

***Optical Quantum Systems:*** In optical quantum systems, different polarizations of photons are used to store information [17]. Qubits are represented by constantly moving photons and gates are physical devices which affect qubits as they flow through them. A single photon is adequate to keep the state of a

qubit and measurement is done via high quality single photon optical detectors to find the polarization of single photons.

### 2.1.3 Multiple-qubit quantum systems

The amount of parallelism in quantum systems increases exponentially with the size of a system. A quantum system with $n$ qubits is a $2^n$ dimensional state space with $2^n$ basis states forming a computational basis. Quantum states in this state space are obtained by the superposition of the basis states. The general state of an $n$-qubit quantum system is shown in Equation 2.4.

$$|\psi\rangle = \sum_{X \in \{0,1\}^n} \alpha_X |X\rangle \tag{2.4}$$

For example, the state vector of a 2-qubit quantum system is a superposition of the four basis states $|00\rangle$,$|01\rangle$, $|10\rangle$ and $|11\rangle$ as shown in Equation 2.5.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2.5}$$

The $\alpha_{ij}$ coefficients are probability amplitudes with complex values. Hence,

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1. \tag{2.6}$$

The state of a 2-qubit quantum system shown in Equation 2.5 cannot be written as any combination of the states of the individual qubits. In other words, the state of the system cannot be decomposed into contributions of its individual particles. This property is called *Entanglement*. Entanglement is one of the reasons for the high computational power of quantum systems. In the entangled state of Equation 2.5, the first qubit is 0 with the probability of $\alpha_{00}^2 + \alpha_{01}^2$. The post measurement state is shown in Equation 2.7.

$$|\psi'\rangle = \frac{\alpha_{00}|0\rangle + \alpha_{01}|1\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \tag{2.7}$$

Note that the post-measurement state is re-normalized by the factor $\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}$ so that it still satisfies the normalization condition.

Two quantum particles can be in an entangled state even when they are far apart. Quantum entanglement is used in *quantum teleportation* to transport data from one

end to the other even in the absence of a quantum communications channel linking the sender to the recipient [13].

The state of a quantum system cannot be copied. This property that qubits cannot be copied is known as the *no-cloning* theorem [57] and is one of the main differences between quantum and classical information processing [13, 23]. An intuitive description of the theorem is that in order to copy a qubit, it must be measured and as was mentioned earlier, measuring a qubit transforms its quantum state to one of the basis states; therefore, the information inside the original qubit is effectively removed after measurement. However, note that it is possible to copy classical values using quantum gates.

## 2.2 Reversible and Quantum Circuits

This section discusses the reversibility property and its applications in designing reversible circuit elements. The first part briefly reviews reversible logic. The second part introduces some common reversible and quantum gates that are used in this dissertation. Finally, reversible and quantum circuits and their characteristics are discussed.

### 2.2.1 Reversible Functions

In classical computing, logic operations are defined as functions over Boolean variables $\mathbb{B} \in \{0, 1\}$.

**Definition 2.1.** *A* **multi-output Boolean function** *is a mapping* $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$*, where* $\mathbb{B} = \{0, 1\}$ *is a Boolean domain and* $n, m \in \mathbb{N}$*. In fact, it is a system of* $m$ *Boolean functions* $f_i(x_1, x_2, \ldots, x_n)$*, where* $1 \leq i \leq m$*.*

**Example 2.1.** *Table 2.1 shows the truth table of the 3-input, 2-output Full Adder function.*

Reversible computation deals with any computational process that is time-invertible, meaning that the process can also be computed backwards to construct inputs from their corresponding outputs and thus no information is lost during a reversible computation. The Full Adder function in Table 2.1 is irreversible in that it is not always possible to obtain a unique input pattern by knowing the output pattern.

Table 2.1: The irreversible full Adder function.

| $\mathbf{c}_{in}$ | **x** | **y** | $\mathbf{c}_{out}$ | **s**$um$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

Boolean reversible functions are a subset of Boolean functions with certain conditions.

**Definition 2.2.** *Let $A$ be any set and define $f : A \to A$ as a one-to-one and surjective function. The function $f$ is called a* **permutation function** *if applying $f$ to $A$ leads to a set with the same elements as $A$ and possibly in a different order. For example, if $A = \{1, 2, 3, \ldots, m\}$, for any $a_i \in A$ there exists a unique $a_j \in A$ such that $f(a_i) = a_j$.*

**Definition 2.3.** *An $n$-input, $n$-output fully specified Boolean function $f : \mathbb{B}^n \to \mathbb{B}^n$ is called* **reversible** *if it is a permutation function.*

An irreversible function can be embedded into a reversible function by adding extra inputs/outputs and assigning values to the generated don't-cares to create a permutation function.

**Definition 2.4.** *The extra output bits (qubits) that are added to a function in order to make it reversible are called* **garbage outputs** *and the extra input bits (qubits) are called* **constant inputs**. *The term constant comes from the fact that the added inputs must have certain values in order to achieve the desired functionality in the outputs.*

The minimum number of garbage outputs required to convert an irreversible function to a reversible function is $\lceil \log_2(M) \rceil$, where $M$ is the maximum number of times that an output pattern is repeated in the corresponding truth table [58].

**Example 2.2.** *Consider the AND function in Table 2.2 (a). The output pattern that has the most number of occurrences is 0 and is repeated 3 times. Hence, at least two garbage outputs and one constant input are required to make the AND function reversible. Table 2.2 (b) shows a reversible form of the AND function where $g_{o1}$ and*

Table 2.2: The AND function

(a) Irreversible

| **a** | **b** | **f** |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(b) Reversible

| $\mathbf{g}_i$ | **a** | **b** | **f** | $\mathbf{g}_{o1}$ | $\mathbf{g}_{o2}$ |
|---|---|---|---|---|---|
| 0 | **0** | **0** | **0** | 0 | 0 |
| 0 | **0** | **1** | **0** | 0 | 1 |
| 0 | **1** | **0** | **0** | 1 | 0 |
| 0 | **1** | **1** | **1** | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |

*$g_{o2}$ are the garbage outputs and $g_i$ is the constant input. Note that assigning 0 to $g_i$ leads to $f$ working as an AND function. The don't-cares in Table 2.2 (b) are assigned in such a way that the $f$ output gives the NAND of $a$ and $b$ if $g_i$ is set to 1.*

**Example 2.3.** *In the Full Adder function of Table 2.1, the maximum number of times that an output pattern is repeated is 3 (for output patterns 10 and 01). Hence, at least $\lceil \log_2(3) \rceil = 2$ garbage outputs and one constant input are required to make the Full Adder function reversible. Table 2.3 shows a reversible form of the Full Adder function where $g_{o1}$ and $g_{o2}$ are the garbage outputs and $g_i$ is the constant input. Note that assigning 0 to $g_i$ leads to obtaining outputs $c_{out}$ and sum, but the don't-cares are assigned in the way that assigning 1 to $g_i$ inverts the $c_{out}$ output.*

There is more than one reversible function associated with each irreversible function depending on the number of constant inputs and garbage outputs used and don't-care assignments. The circuits that result from synthesizing these embedded reversible functions are different in terms of the number of gates and the number of qubits and hence have different implementation costs. Consequently, the process of embedding an irreversible function into a reversible function is of significant importance and has remained an open problem while being studied in many articles [45, 59–61].

### 2.2.2 Reversible and Quantum Gates

In classical computers, circuits are composed of logic gates connected by wires and information is transferred through wires as electric power. The quantum circuit model on the other hand is quite different from classical structures. In quantum circuits, information is kept as the quantum state of certain quantum particles and the term

Table 2.3: A reversible Full Adder

| $\mathbf{g}_i$ | $\mathbf{c}_{in}$ | $\mathbf{x}$ | $\mathbf{y}$ | $\mathbf{c}_{out}$ | $\mathbf{s}um$ | $\mathbf{g}_{o1}$ | $\mathbf{g}_{o2}$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

'gate' refers to any process such as laser beams, microwave radiations, *etc.* that is applied to change the state of those particles. Consequently, a quantum circuit is the complete process of applying a set of consecutive quantum gates one after each other on the input qubits. Hence, it includes a time factor which justifies the limitation of not being able to use feedbacks in quantum circuits as it is not possible to go backwards through time.

**Definition 2.5.** *A gate is called* **reversible** *if it has the same number of inputs and outputs and realizes a reversible function.*

Quantum gates are inherently reversible meaning that the inverse operation can be applied thereby erasing the effect of the original operation [13]. Reversible gates are linear operators and hence can be represented by matrices. A reversible gate with $n$ inputs and $n$ outputs in an $r$-valued logic is represented by an $r^n \times r^n$ *unitary* matrix.

**Definition 2.6.** *Operator $U$ is called* **unitary** *iff $UU^\dagger = I$ which means that the result of the product of $U$ and its adjoint (complex conjugate transpose) is the identity matrix [13].*

If a matrix is unitary, it is eligible to represent a reversible gate; however, the transformation of a unitary matrix may be too complicated to be physically implementable as a gate.

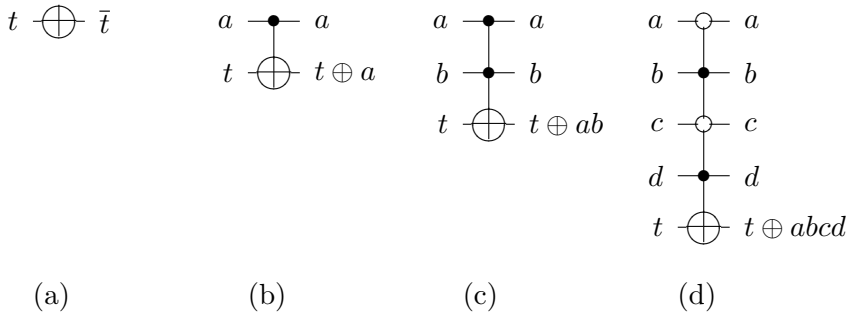



Figure 2.2: (a) $T(;t)$, (b) $T(a;t)$, (c) $T(a,b;t)$ (d) $T(\overline{a},b,\overline{c},d;t)$.

**Definition 2.7.** *Given a gate $G$ with matrix $A$, the inverse of $G$ is denoted by $G^{-1}$ and corresponds to the inverse matrix $A^{-1}$. Since $AA^{-1} = I$, applying a gate and its inverse to a quantum state does nothing and equals the identity transformation.*

In the following discussion, we introduce some quantum and Boolean reversible gates that are commonly used in synthesis and technology mapping approaches.

**Reversible Gates:** In order to distinguish between quantum reversible gates and Boolean reversible gates, we refer to the former as quantum gates and to the latter as reversible gates. Note that quantum gates are also reversible. We here introduce some reversible gates that are used in this work.

- NOT Gate: A reversible *NOT* gate is similar to the conventional NOT gate. It has one input and one output and inverts the input values. The matrix representation and circuit notation of this gate are shown in Equation 2.8 and Figure 2.2 (a) respectively.

$$\boldsymbol{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2.8}$$

- CNOT Gate: A *controlled-NOT (CNOT)* gate has a control and a target, and inverts the target whenever the control is set to 1. The symbol associated to this gate is shown in Figure 2.2 (b). The value on the control line passes through the gate unchanged.

- **MPMCT Gates**: A *Mixed Polarity Multiple-Control Toffoli (MPMCT)* gate with *target line* $x_t$, *positive controls* $\{x_{i_1}, x_{i_2} \cdots x_{i_k}\}$ and *negative controls* $\{x_{i_{k+1}} \cdots x_{i_m}\}$ maps $x_t$ to $(x_{i_1}x_{i_2}\cdots x_{i_k}\overline{x_{i_{k+1}}}\cdots\overline{x_{i_m}})\oplus x_t$ where $\oplus$ denotes the exclusive-OR operation. Hence, an MPMCT gate inverts the target iff all positive controls are set to 1 and all negative controls are set to 0; otherwise, the value on the

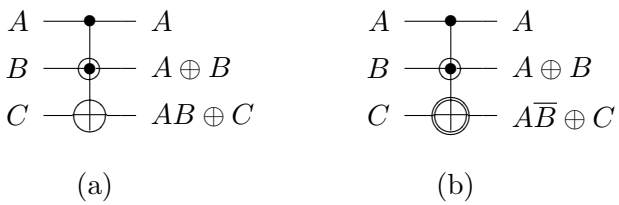



Figure 2.3: (a) Peres gate, (b) TR gate.

target line will pass through the gate unchanged. Note that the values on the control lines also pass through the gate unchanged.

The *size* of an MPMCT gate is the number of controls plus one. An MPMCT gate with no controls is the reversible NOT and an MPMCT gate with a single positive control is a CNOT gate. An MPMCT gate with two positive controls is called a *Toffoli* gate [62]. If controls of an MPMCT gate are all positive, it is called *Multiple-Control Toffoli (MCT)* for simplicity.

MPMCT gates are denoted by $T(C;t)$ with $C$ being the set of controls and $t$ being the target. Negative controls are identified by an over bar. To draw an MPMCT gate, the notation $\bigoplus$ is used to indicate the target line, a $\bullet$ is used to show a positive control connection and a $\circ$ is used to indicate a negative control connection. For example, Figure 2.2 shows a NOT gate (a), the CNOT gate (b), the Toffoli gate (c), and the MPMCT gate $T(\overline{a}, b, \overline{c}, d; t)$ (d).

- **Peres Gate**: The *Peres* gate [63] has a control, a target, and a line that serves as both a control and a target. The Peres gate is illustrated in Figure 2.3 (a).

- **TR Gate**: The *TR* gate [1] is the inverse of the Peres gate and accordingly has a control, a target, and a line that serves as both a control and a target. The Peres gate and its functional operation on the qubits is illustrated in Figure 2.3 (b).

**Quantum Gates:** We here introduce some of the most common single-qubit and two-qubit quantum gates [13].

- **Pauli-X Gate (NOT):** The Pauli-X is the quantum NOT gate and applies the transformation of the matrix in Equation 2.8. The gate symbol is shown in Figure 2.4 (a). Applying a NOT gate to a qubit $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ leads to the role of $|0\rangle$ and $|1\rangle$ being changed $|\psi\rangle = \alpha|1\rangle + \beta|0\rangle$ as shown in the following


(a) (b) (c) (d)

Figure 2.4: (a) NOT gate, (b) Hadamard gate, (c) $V$ gate, (d) $V^+$ gate.

Equation.

$$\boldsymbol{X}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha\end{pmatrix} \tag{2.9}$$

A reversible NOT gate is a Pauli-X gate that only applies to Boolean logic.

- **Hadamard Gate:** The *Hadamard* gate is represented by the matrix shown in Equation 2.10. This gate is sometimes described as being like a square-root of NOT gate in the sense that it turns a $|0\rangle$ into $(|0\rangle + |1\rangle)/\sqrt{2}$ (first column of $\boldsymbol{H}$) which is halfway between $|0\rangle$ and $|1\rangle$, and turns a $|1\rangle$ into $(|0\rangle - |1\rangle)/\sqrt{2}$ (second column of $\boldsymbol{H}$) which is also halfway between $|0\rangle$ and $|1\rangle$. However, it is important to note that $H^2$ is not a NOT gate and applying $H$ to a quantum state twice leads to the identity $\boldsymbol{H}^2 = \boldsymbol{I}$ which means that the Hadamard gate is self inverse $H = H^{-1}$. The Hadamard gate is shown in Figure 2.4 (b).

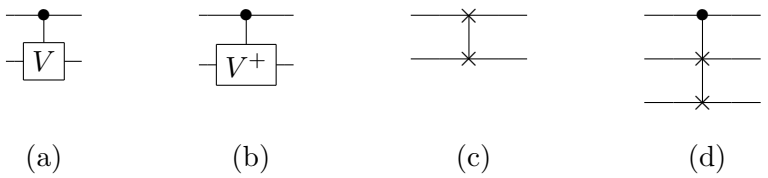
$$\boldsymbol{H} = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{2.10}$$

- $V$ **Gate:** The $V$ gate applies the transformation defined by the matrix in Equation 2.11 to the target qubit. The $V$ gate is referred to as a square-root of the NOT gate since $\boldsymbol{V}^2 = \boldsymbol{X}$ meaning that two consecutive $V$ gates apply the same transformation as the NOT gate. This gate is shown in Figure 2.4 (c).

$$\boldsymbol{V} = \frac{1+i}{2}\begin{pmatrix}1 & -i\\ -i & 1\end{pmatrix} \tag{2.11}$$

- $V^+$ **Gate:** The $V^+$ gate is the inverse of the $V$ gate *i.e.* $VV^+ = I$ and applies the transformation defined by the matrix in Equation 2.12 to the target qubit. The $V^+$ gate is also referred to as a square-root of the NOT gate since $\boldsymbol{V^+}^2 = \boldsymbol{X}$. Figure 2.4 (d) shows the symbol for the $V^+$ gate.

$$\boldsymbol{V}^+ = \frac{1-i}{2}\begin{pmatrix}1 & i\\ i & 1\end{pmatrix} \tag{2.12}$$




(a) (b) (c) (d)

Figure 2.5: (a) controlled-$V$ gate, (b) controlled-$V^+$ gate, (c) Swap gate, (d) Fredkin gate.

- **controlled-$V$ and controlled-$V^+$ Gates**: A *controlled-V* gate is a 2-input/2-output quantum gate with one control and one target qubit. This gate applies the transformation defined by the matrix in Equation 2.11 to the target line if its control line is set to 1. The inverse of the controlled-$V$ gate is another 2-input/2-output quantum gate called *controlled-$V^+$* gate that changes its target line using the transformation shown in Equation 2.12 if its control line is set to 1. The value on the control line is passed through the gate unchanged. Note that in this dissertation we always consider the controls for $V$ and $V^+$ gates to be positive. Figure 2.5 (a) and (b) shows the controlled-$V$ and controlled-$V^+$ gates respectively.

- **Swap Gate**: A *Swap* gate has two inputs and two outputs and swaps the two qubits. It is identified by the matrix of Equation 2.13 and is shown in Figure 2.5 (c).

$$S = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

- **Fredkin Gate**: The *Fredkin* gate was initially defined as a Boolean reversible gate [64], but can be considered as a quantum gate in general. A Fredkin gate is a controlled-Swap gate with three qubits, one control and two targets. If the control qubit is set to 1, the states of the two targets are interchanged; otherwise, no transformation is applied to the targets. Figure 2.5 (d) shows a Fredkin gate.

As a valuable visual aid to the relation between the types of gates just introduced, we note that with reference to the Bloch sphere model of a qubit (Figure 2.1):

- NOT defines a rotation about the $x$-axis by $\pi$ radians,

- Hadamard rotates the input about the $y$-axis by $\pi/2$ radians followed by a rotation about the $x$-axis by $\pi$ radians,
- V (V+) defines a counter-clockwise (clockwise) rotation about the $x$-axis by the angle $\pi/2$ radians.

**Reversible and Quantum Gate libraries:**

Circuits are usually implemented by a universal set of basic gates called a *gate library*. A commonly used gate library in classical circuits is the set of AND, OR, and NOT gates as any Boolean function can be implemented using these gates.

A universal set of reversible gates is called a *reversible gate library* if any reversible function is realizable using the gates in it. Examples of reversible gate libraries include the NCT library (that contains NOT, CNOT, and Toffoli gates), NCTS library that has the same set as NCT plus the Swap gate [13], and the library containing only *Fredkin* (Controlled-Swap) gate [64]. The set of Multiple-Control Toffoli (MCT) gates and the set of Mixed-Polarity Multiple-Control Toffoli (MPMCT) gates are two other examples of reversible gate libraries.

A *quantum gate library* is composed of a universal set of quantum gates in that any reversible function can be implemented using the gates in it. Quantum gate libraries are used in technology mapping reversible circuits to quantum circuits. One of the most common libraries used for technology mapping is the *NCV library* consisting of the NOT, CNOT, controlled-$V$, and controlled-$V^+$ gates with positive controls.

### 2.2.3 Circuits and Examples

Reversible circuits can be implemented in different classical and quantum technologies. Classical reversible circuits consist of reversible gates that can be fired either forwards or backwards depending on the applied driving force. Implementation of reversible circuits in CMOS technology has been studied in [65, 66]. However, to make the best use of the reversibility properties *e.g.* low power dissipation, reversible circuits need to be implemented in reversible infrastructures such as quantum technologies which are inherently reversible.

**Definition 2.8.** *A* **Reversible Circuit** *over $n$ input bits (qubits) $X = \{x_1, x_2, \ldots, x_n\}$ is a cascade of reversible gates $g_1, g_2, \ldots, g_m$. A gate operation is applied to its targets iff all the controls meet the required conditions. Control lines and unconnected lines*

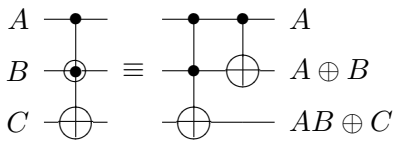



Figure 2.6: Peres gate and its equivalent MCT circuit.

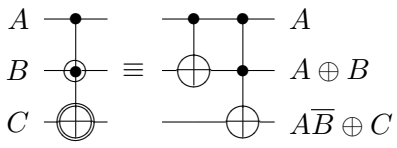



Figure 2.7: TR gate and its equivalent MCT circuit.

*always pass through a reversible gate unchanged. Reversible circuits have the same number of inputs and outputs as do the individual gates.*

A cascade of quantum gates applied to a set of qubits is referred to as a *quantum circuit*. Quantum Circuits are reversible. The physical structure of quantum circuits is completely different from that of classical circuits in that a set of operations are applied to a set of quantum particles holding quantum states instead of transistors and wires used in classical circuits. Feedback is not allowed in quantum circuits because it is not possible to go back in time when applying the gates consecutively. Fan-out is not allowed in quantum circuits due to the no-cloning theorem [13, 57]. Information is not lost as it passes through a reversible circuit hence there is no power dissipation due to information loss [11]. The state of a quantum circuit with $n$ qubits is represented by a vector in $2^n$-dimensional Hilbert Space $H_2^n$ [13].

**Example 2.4.** *Figure 2.6 shows the MCT realization of a Peres gate. It consists of a Toffoli gate $T(A, B; C)$ followed by a CNOT gate $T(A; B)$. Assigning the 0 value to the third bit $C = 0$ leads the Peres gate to act like a one-bit half adder with the second bit being the sum $= A \oplus B$ and the third bit being the carry-out $= AB$.*

**Example 2.5.** *The MCT realization of the TR (inverse Peres) gate is shown in Figure 2.7.*

**Example 2.6.** *Figure 2.8 shows an MCT circuit equivalent to the Swap gate which is composed of three consecutive CNOT gates. The CNOT gates implement the following*

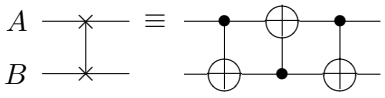



Figure 2.8: Swap gate and its equivalent MCT circuit.

*ordered exclusive-OR operations to swap the inputs.*

$$\begin{aligned} B &= A \oplus B \\ A &= A \oplus B \\ B &= A \oplus B \end{aligned} \tag{2.14}$$

As shown in the above examples, a reversible function can be represented by corresponding Boolean expressions for each output that consist of only positive form of variables and $AND$ and $XOR$ operations. This form of representation is called *Positive Polarity Reed Muller(PPRM)* [67]. In PPRM expressions, the negated form of a variable, $\overline{a}$, is shown by $a \oplus 1$. Any Boolean expression can be written in PPRM format.

The matrix representation of a reversible circuit is obtained by the gate matrices as described in the following steps. The order of the variables must be the same for all matrices in all operations:

- The equivalent matrix of two consecutive gates $G_1 G_2$ is the outer product of the respective matrices in reverse order *i.e.* $\boldsymbol{A}_2 \boldsymbol{A}_1$.
- The equivalent matrix of two concurrent gates that are applied on different qubits is the *Kronecker Product* [13, 23, 68] of the respective matrices.

The Kronecker Product is a way of putting vector spaces together to form larger vector spaces and is denoted by $\otimes$. More precisely, the Kronecker product of two Hilbert spaces $H_n$ and $H_m$ is a Hilbert space with dimension $n \times m$.

$$H_{nm} = H_n \otimes H_m \tag{2.15}$$

**Example 2.7.** *Consider the circuit of Figure 2.6. The matrix representation of the circuit is obtained by the outer product of the matrix equivalent to the second part, $G_2$, that contains a CNOT gate $T(A; B)$ and a line which acts as an identity gate by the matrix of the Toffoli gate, $G_1$. The matrix of $G_2$ is calculated by the Kronecker*

*product of the CNOT matrix and the identity matrix as shown in Equation 2.16. The matrix of the cascade is obtained by multiplying the matrices* $\boldsymbol{M}_{G_2}$ *and* $\boldsymbol{M}_{G_1}$ *as shown in Equation 2.17.*

$$M_{G_2} = M_{CNOT} \otimes M_I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2.16}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$M_{Peres} = M_{G_2} \times M_{G_1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.17}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

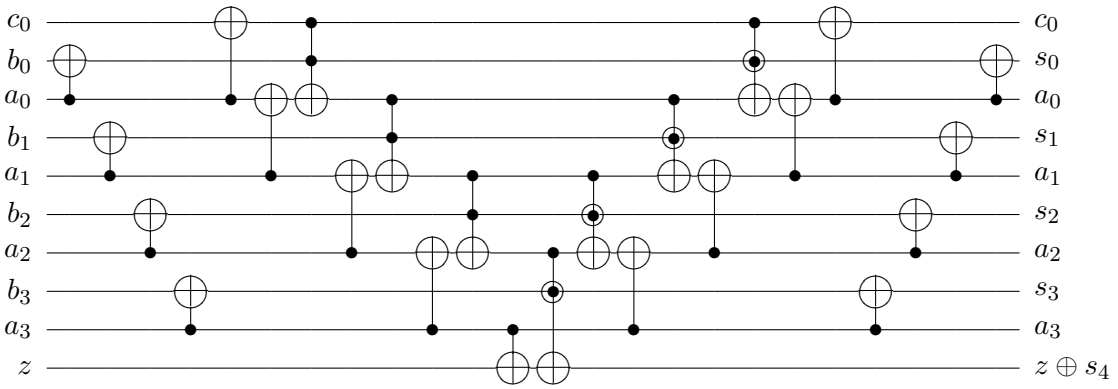



Figure 2.9: A 4-bit reversible ripple-carry adder [1].

If the reversible circuit realizing a reversible function is available, the reversible circuit realizing the inverse function is easily obtained using the following property.

**Property 2.1.** *Given a cascade of reversible gates $G_1 G_2 \dots G_k$ realizing the reversible function $F$, the cascade $G_k^{-1} \dots G_2^{-1} G_1^{-1}$ realizes the function $F^{-1}$, where $G_i^{-1}$ is the inverse gate for $G_i$.*

*Proof.* The gates in the circuit $G_1 G_2 \dots G_k$ apply the transformation $\boldsymbol{S} = \prod_{i=k}^{1} \boldsymbol{A}_i$ in which $\boldsymbol{A}_i$ is the corresponding matrix for gate $G_i$. The inverse operation $S^{-1}$ is obtained by $\boldsymbol{S}^{-1} = \prod_{i=1}^{k} \boldsymbol{A}_i^{-1}$ in which $\boldsymbol{A}_i^{-1}$ is the corresponding matrix for gate $G_i^{-1}$ using Definition 2.7. Hence, the cascade $G_k^{-1} \dots G_2^{-1} G_1^{-1}$ realizes the inverse function. □

**Property 2.2.** *MPMCT gates are self-inverse and two consecutive identical MPMCT gates cancel each other and yield the identity mapping.*

**Property 2.3.** *Since an MPMCT gate is self-inverse, applying Property 2.1 to a realization of the gate yields an alternate realization for the same gate which is called the* **inverse realization**.

**Property 2.4.** *Since an MPMCT gate is self-inverse, reversing the realization of the gate yields an alternate realization for the same gate which is called its* **reverse realization**.

**Example 2.8.** *Figure 2.9 shows a modified version of the 4-bit reversible ripple-carry adder introduced in [1]. The adder takes two 4-bit inputs $A = (a_3, a_2, a_1, a_0)$ and $B = (b_3, b_2, b_1, b_0)$, a carry-in bit $c_o$ and an extra bit $z$ as inputs. The $c_0$ and*

*$A = (a_3, a_2, a_1, a_0)$ values are restored in the outputs. The input $B = (b_3, b_2, b_1, b_0)$ is transformed into the sum of $A$ and $B$, $S = (s_3, s_2, s_1, s_0)$, and input $z$ changes to $z \oplus s_4$. One of the interesting properties of the reversibility is that once a circuit is designed, the inverse function can be easily implemented by using the inverse circuit. For example, the inverse of the circuit of Figure 2.9 is a subtractor and running the circuit backwards (using the inverse gates) yields $B$ as the result of subtracting $A$ from $S$* i.e. *$B = S - A$.*

**Reversible and Quantum Cost Functions:**

Various cost metrics have been considered in the literature in order to evaluate quantum circuits such as the number of gates, circuit depth, concurrency, gate distribution on qubits, etc. [52, 69] among which the two most popular cost metrics are quantum cost and the number of qubits.

The actual cost of a physical implementation of a quantum gate depends highly on the target technology. However, in design automation of reversible and quantum circuits, 1-qubit and 2-qubit quantum gates that are usually directly implemented are considered to have unit quantum cost and are referred to as *elementary quantum gates*. Each elementary quantum gate relates to one or more physical operations in the target quantum technology. This approximation is suitable in the sense that it gives a good measure for complexity of larger reversible gates and is technology independent; hence, it can be used in technology independent synthesis and optimization approaches.

**Definition 2.9. Quantum cost** *of a circuit is the number of gates obtained as a result of technology mapping that circuit into a cascade of elementary quantum gates. The gates in the NCV library,* i.e. *NOT, CNOT, $V$ and $V^+$ gates, are examples of elementary quantum gates.*

Due to current technology limitations, the number of qubits that can be placed in a quantum system is restricted. In a quantum circuit, qubits are in an entangled quantum state meaning that it is impossible to separate the contributions of the states of the individual qubits from the state of the whole system [23]. The number of quantum bits that can be placed together in an entangled quantum system is restricted. The highest number of entangled qubits that have been physically implemented is 14 as reported in [70]. Consequently, the number of qubits in a quantum circuit is an important cost metric which can be varied based on the number of ancillaries used in a circuit.

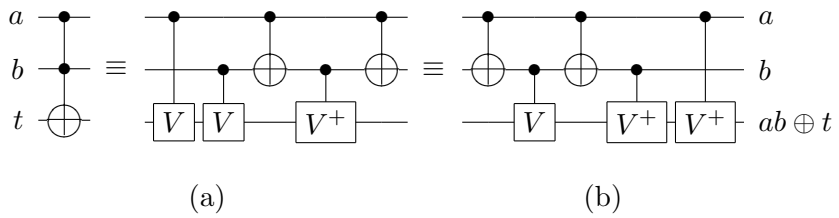



Figure 2.10: NCV realizations of a 2-control MCT gate.

**Mapping and optimization:**

A complex reversible or quantum gate can be realized by a cascade of elementary quantum gates. Ancillary lines are useful in mapping reversible gates in that using ancillaries may help to reduce the size of the resulting quantum circuit [5, 71].

**Example 2.9.** *Figure 2.10 (a) shows an NCV circuit resulting from mapping a 2-control MCT gate to the NCV library. As demonstrated, the quantum cost of this Toffoli gate is five. The inverse realization (Property 2.3) of this gate is shown in Figure 2.10 (b). Alternate realizations can also be found by interchanging the $V$ and $V^+$ gates in the circuits of Figure 2.10 (a) and (b).*

Optimization of reversible and quantum circuits is generally concerned with finding a cascade of gates realizing the identity function and removing them from the circuit. For example, MCT gates are self-inverse and two consecutive identical MCT gates yield the identity mapping and thus they cancel each other. $V$ and $V^+$ gates with the same target and the same control are the inverses of each other and two consecutive such gates can be removed from the circuit.

**Example 2.10.** *Consider the Peres gate shown in Figure 2.6. The equivalent MCT cascade for this gate is shown in Figure 2.11 (a). This circuit consists of a Toffoli gate T(A,B;C) followed by a CNOT gate T(A;B). Using the quantum realization of the Toffoli gate shown in Figure 2.10 (a), the Peres gate is mapped to a cascade of six NCV gates as shown in Figure 2.11 (b). As stated above, the CNOT gates are self-inverse and two consecutive identical CNOT gates yield the identity mapping; hence, the last two gates in the cascade of Figure 2.11 (b) are redundant and can be removed from the circuit which yields the circuit of Figure 2.11 (c), the quantum realization of a Peres gate with cost 4.*

**Example 2.11.** *Using the same approach as the previous example, the equivalent NCV circuit for the inverse Peres (TR) gate is obtained with cost 4 as shown in*

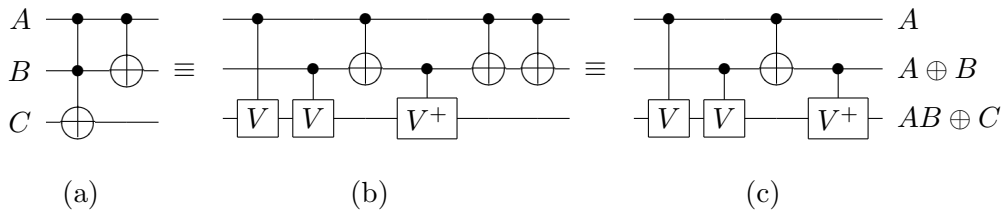



Figure 2.11: Peres gate (a) MCT realization, (b) after mapping to quantum gates, (c) optimized quantum realization.

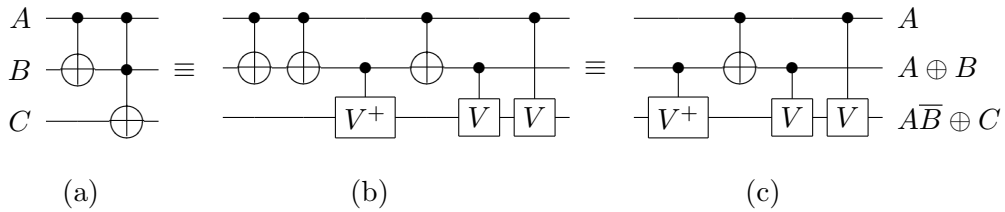



Figure 2.12: TR gate (b) after mapping to quantum gates, (c) optimized quantum realization.

*Figure 2.12. Note that the reverse realization of Figure 2.10 (a) is used to map the 2-control MCT gate.*

The above examples illustrate an important property in MPMCT circuits. Note that the circuit in Figure 2.12 (c) is the reverse of the circuit in Figure 2.11 (c) and realizes its inverse function. Due to the reverse realization property of MPMCT gates (Property 2.4), reversing the order of quantum gates in a realization of an MPMCT circuit yields a realization for the inverse circuit.

## 2.3 Decision Diagrams

Decision diagrams are graphical representations for functions, sets, and relations. They can also be used to represent matrices and circuits. They provide an efficient and compact data-structure that can be used for large problems. Different types of decision diagrams have been introduced in the literature. Here, we briefly introduce two types of decision diagrams that are used in this dissertation: *Reduced Ordered Binary Decision Diagrams (ROBDDs)* [48] and *Quantum Multiple-valued Decision Diagrams (QMDDs)* [50, 72]. Another type of decision diagrams called *Decision Diagrams for Matrix Functions (DDMFs)* [73] is discussed in detail in Chapter 4.

### 2.3.1 Reduced Ordered Binary Decision Diagrams (ROBDDs)

Every Boolean function $f : \mathbb{B}^n \to \mathbb{B}$ can be represented by a Binary Decision Diagram. The original idea was first introduced by Lee [74] and further refined and studied in many articles including [48, 75, 76].

**Definition 2.10.** *A* **Binary Decision Diagram (BDD)** *over Boolean variables $X$ is a directed acyclic graph $G = (V, E)$ with the following properties:*

1. *Each node $v \in V$ is either a terminal or a nonterminal.*
2. *Each terminal node $v \in V$ is labeled by a value $t \in T = \{0, 1\}$ and has no outgoing edge.*
3. *Each nonterminal node $v \in V$ is labeled by a Boolean variable $x_i \in X$.*
4. *The Shanon decomposition [77], $f = \overline{x_i} f_{x_i=0} + x_i f_{x_i=1}$, holds for each nonterminal node which leads to two outgoing edges $e \in E$ whose successors are denoted by low(v) (for $f_{x_i=0}$) and high(v) (for$f_{x_i=1}$) respectively.*

The size of a BDD is defined as the number of nonterminal nodes. A BDD is called *free* if no variable appears more than once on any single path from its root to a terminal node. A BDD is *ordered* if the variables on every path from its root to a terminal vertex adhere to the same order. In *reduced* BDDs isomorphic subgraphs are shared and redundant nodes are removed.

**Definition 2.11.** *A BDD that is both reduced and ordered is called a* **Reduced Ordered Binary Decision Diagram (ROBDD)**.

The ROBDD representation of a function is unique for a fixed variable ordering [48]. However, the size of an ROBDD highly depends on the order of the variables.

**Example 2.12.** *Figure 2.13 shows an ROBDD representation of the function $f(x, y, z) = x + yz$ in which the order of the variables is $(x, y, z)$. A dashed line out of a vertex $v$ shows its low edge ($v = 0$) and a solid line out of $v$ shows its high edge ($v = 1$).*

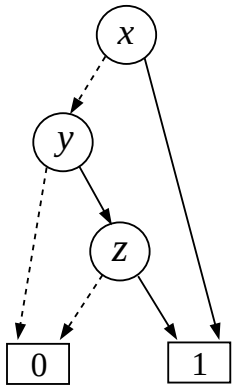



Figure 2.13: ROBDD representation of the function $f(x, y, z) = x + yz$.

### 2.3.2 Quantum Multiple-valued Decision Diagrams (QMDDs)

Recalling from the previous section, reversible and quantum operations are defined by $p^n \times p^n$ unitary matrices with complex values in a $p$-valued logic where $n$ is the size of a gate or the number of inputs of a circuit. Quantum Multiple-valued Decision Diagrams are designed to effectively represent the complex-valued matrices assigned to quantum cascades and to facilitate the manipulation and operations on those matrices.

A QMDD structure is based on partitioning a matrix $M$ with dimension $p^n \times p^n$ into a $p \times p$ grid matrices $M_i$ with dimension $p^{n-1} \times p^{n-1}$ as shown in Equation 2.18.

$$M = \begin{pmatrix} M_0 & M_1 & \cdots & M_{p-1} \\ M_p & M_{p+1} & \cdots & M_{2p-2} \\ \vdots & \vdots & \ddots & \vdots \\ M_{p^2-p} & M_{p^2-p+1} & \cdots & M_{p^2-1} \end{pmatrix} \tag{2.18}$$

A variable is assigned to the above partitioning for which each node has $p^2$ outgoing edges to the corresponding $p^2$ matrices. This partitioning can be recursively performed on each submatrix until the resulting submatrices are single elements. Assigning variables to the matrices at each level results in a decision diagram with shared common structures and no redundant vertices which is defined as follows.

**Definition 2.12.** *A* **Quantum Multiple-valued Decision Diagram (QMDD)** *is a directed acyclic graph with the following properties:*

1. *There is a single* **terminal vertex** *with associated value 1 which has no outgoing*

*edges.*

2. *There are a number of* **nonterminal vertices** *each labeled by a $p^2-$valued selection variable. Each nonterminal vertex has $p^2$ outgoing edges named $e_0, e_1, \ldots,$ $e_{p^2-1}$.*

3. *One vertex is the* **start vertex** *and has a single incoming edge that itself has no source vertex.*

4. *Every edge in the QMDD (including the one leading to the start vertex) has an associated complex* **weight**.

5. *The selection variables are ordered. Assume with no loss of generality that the order of the variables beginning at the start vertex is $\{x_0, x_1, \ldots, x_{n-1}\}$, the QMDD satisfies the following two rules:*

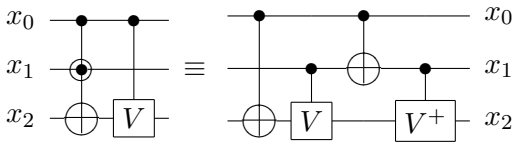
   - *Each selection variable appears at most once on each path from the start vertex to the terminal vertex.*
   - *An edge from a nonterminal vertex labeled $x_i$ points to a nonterminal vertex labeled $x_j$, $j > i$ or to the terminal vertex. Hence $x_0$ is closest to the terminal vertex and $x_k$ for the highest $k$ present labels the start vertex.*

6. *No nonterminal vertex is* **redundant** i.e. *all its outgoing edges point to a common vertex with the same weight.*

7. *Each nonterminal node is* **normalized** *such that $e_j$, $0 \leq j \leq p^2 - 1$, has weight 1 and $\forall e_i$, $i < j$ have weight 0. Note, such an $e_j$ must exist or the vertex would be redundant (all weights 0).*

8. *Nonterminal vertices are* **unique** i.e. *no two of them have the same label and the same set of outgoing edges with equal weights.*

Evaluating a QMDD for a particular input/output assignment in a function is equivalent to finding the corresponding element in the matrix of the function. The evaluation involves following the path from the start vertex to the terminal vertex and multiplying the weights of the edges as they are being traversed. Columns of the matrix correspond to input patterns and rows correspond to possible output patterns *i.e.* if $M_{ij} = 1$ then the input pattern in column $j$ is mapped to the output pattern in Row $i$ in the truth table of the function.




$$\left(\begin{array}{cccc|cccc} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ \hline 0&0&0&0&0&0&\frac{1-i}{2}&\frac{1+i}{2}\\ 0&0&0&0&0&0&\frac{1+i}{2}&\frac{1-i}{2}\\ 0&0&0&0&\frac{1+i}{2}&\frac{1-i}{2}&0&0\\ 0&0&0&0&\frac{1-i}{2}&\frac{1+i}{2}&0&0 \end{array}\right)$$

(a) (b)

Figure 2.14: (a) The circuit, (b) Matrix representation.

As in all decision diagrams, there is a canonical representation for QMDDs which is obtained by a normalization procedure described in [72].

**Example 2.13.** *Figure 2.14 (a) shows a 3-input circuit composed of a Peres gate followed by a controlled-V gate. The corresponding matrix for this circuit is depicted in Figure2.14 (b). The matrix is* $2^3 \times 2^3$ *since there are three qubits and the logic has two basis states* $|0\rangle$ *and* $|1\rangle$*. The QMDD structure for this circuit is given in Figure 2.15. Each vertex has four outgoing edges labeled* 0 *to* 3 *from left to right. Each edge is assigned a complex weight and corresponds to one of the partitions in the original matrix. For example, the partitioning for the first variable,* $x_0$*, is shown in the matrix of Figure 2.14. The first edge out of the* $x_0$ *vertex corresponds to the upper left partition. The second and third edges correspond to the upper right and lower left partitions respectively. These edges point to* 0 *matrices that are shown by* 0 *stubs. The fourth edge out of the* $x_0$ *vertex points to the lower right partition. Each partition is further partitioned accordingly for the rest of the variables. Elements of the matrix are obtained by multiplying the complex weights on each path from the root to the terminal nodes.*

## 2.4 Summary

The required background for the discussions in the subsequent chapters of this dissertation was reviewed in this chapter. The chapter mainly introduced the basic concepts of quantum mechanics and the required background on reversible and quantum circuits. Some decision diagram structures that are used in the approaches discussed in this dissertation were also introduced.

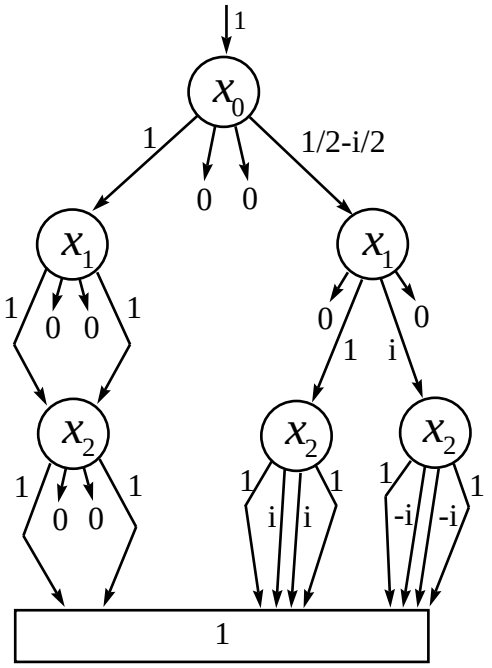



Figure 2.15: QMDD structure of Example 2.13.

# Chapter 3

# Optimization

Reversible synthesis methods are automated tools that transform a given reversible function to a reversible or a quantum circuit. Finding the optimal reversible or quantum circuit for a given function is an intractable problem [45] and usually the circuits resulting from synthesis are not optimal. Hence, as in classical logic design, post-synthesis optimizations are required to achieve low cost reversible/quantum circuits. The optimizations are sometimes technology specific and are suited to the target technology. However, technology independent optimizations can be applied at any time based on technology independent metrics such as the number of elementary quantum gates, the number of qubits, *etc.* The results of this part of the dissertation have been published in [78–82].

The first part of this chapter gives a brief summary of the existing optimization methods. Section 2 is devoted to the new constraints proposed in this dissertation for moving reversible or quantum gates in a circuit. The proposed optimization procedure is described in section 3 and section 4 concludes the discussion.

## 3.1 Previous Work

Optimization of reversible and quantum circuits is generally concerned with finding a cascade of gates realizing the identity function and removing it from the circuit, or finding subcircuits that can be realized by fewer gates and replacing them with their optimized counterparts. This section provides a literature review for the existing optimization approaches. These methods are classified based on their functionality in the following subsections.

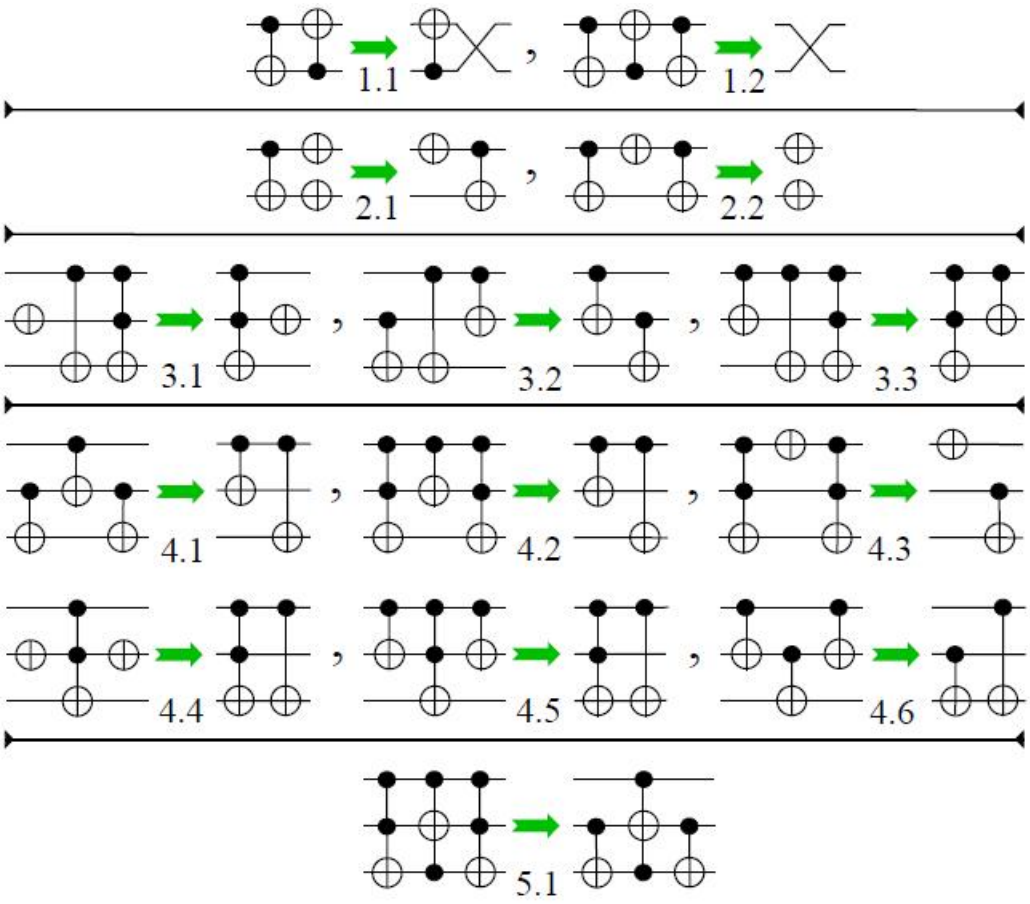



Figure 3.1: Reversible reduction rules [2].

### 3.1.1 Template Matching

A common approach to reversible and quantum circuit simplification is *template matching*. In template matching methods, first, a set of templates is defined showing the possible reductions for a number of cascades. Then the circuit is examined to find such templates. Sometimes the gates inside a circuit are reordered to find the desired templates. Each template that is found is replaced by a functionally equivalent smaller cascade.

Templates were initially defined as reduction rules *e.g.* [2]. A set of templates of this form for reversible circuits is shown in Figure 3.1 [2]. However, according to the current definition, a template of size $m$ is an optimal cascade of $m$ gates that realizes the identity function. Template matching methods usually define all templates up to a certain size for a given gate library. For example, [83] has enumerated all Fredkin-Toffoli templates with less than six gates. All Toffoli gate templates of size seven

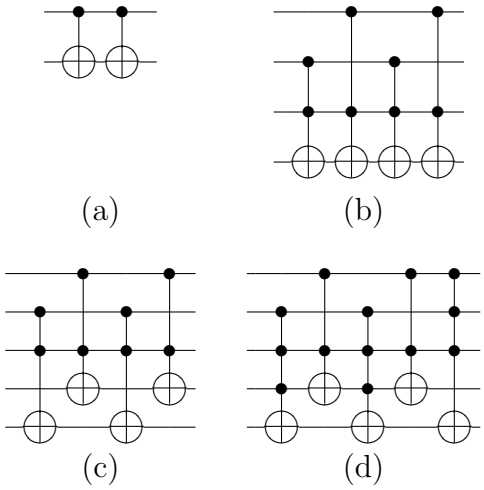



Figure 3.2: Sample reversible templates [3].

and less can be found in [2]. A circuit is examined to find more than half of the gates in a template in either forward or backward direction and the found cascade is substituted with the inverse cascade (Property 2.1) of the rest of the gates in the corresponding template. For example, for a template $G_0G_1 \dots G_{m-1}$ the forward and backward applications are as follows [83]:

- Forward application: A sequence of gates in the circuit that matches the cascade $G_iG_{(i+1) \bmod m} \dots G_{(i+k-1) \bmod m}$ in the template is replaced with the sequence $G^{-1}_{(i-1) \bmod m}G^{-1}_{(i-2) \bmod m} \dots G_0^{-1}G^{-1}_{m-1} \dots G^{-1}_{(i+k) \bmod m}$, where $\frac{m}{2} \leq k \leq m$.
- Backward application: A sequence of gates in the circuit that matches the cascade $G_iG_{(i-1) \bmod m} \dots G_{(i-k+1) \bmod m}$ in the template is replaced with the sequence $G^{-1}_{(i+1) \bmod m}G^{-1}_{(i+2) \bmod m} \dots G^{-1}_{m-1}G_0^{-1} \dots G^{-1}_{(i-k) \bmod m}$, where $\frac{m}{2} \leq k \leq m$.

Examples of reversible template matching optimizers include [2, 3, 40, 83]. Figure 3.2 shows some reversible templates. Quantum templates can be found in [4, 39, 84]. Some quantum templates are depicted in Figure 3.3.

### 3.1.2 Using ancillaries

Depending on the target technology and the application of the resulting circuit, sometimes extra ancillary lines are more effective in reducing the quantum cost [85, 86].

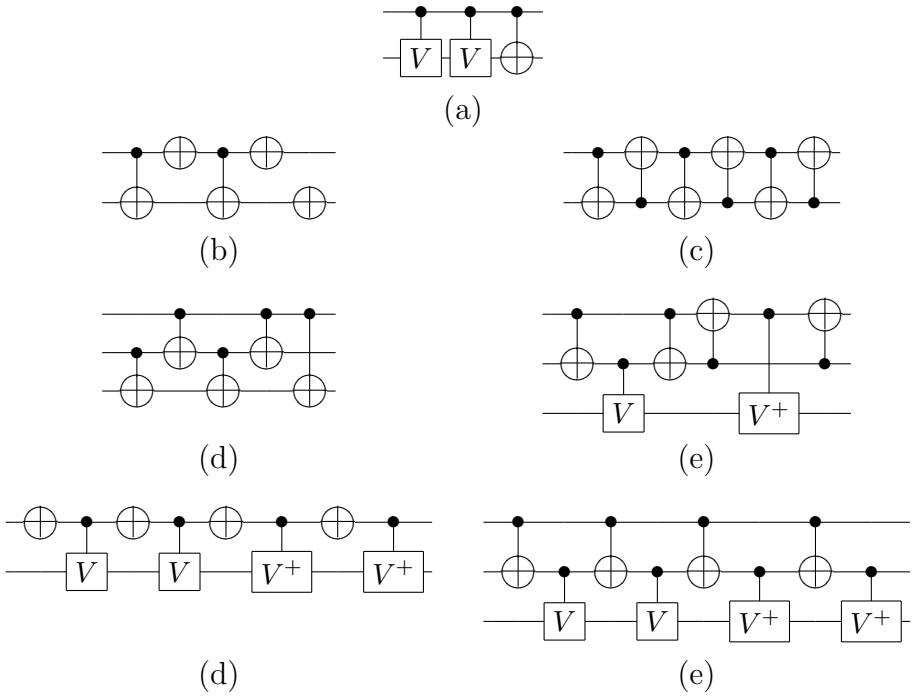



Figure 3.3: Sample quantum templates [4].

In [85], a set of common controls across a cascade of gates is found and the corresponding controls are factored (extracted from the circuit). Then, a pair of gates having the same set of controls are added at each end of the cascade whose targets are placed on an extra helper line along with a single control for each factored gate. This approach reduces the number of controls and hence the complexity of reversible gates by adding one or more extra ancillaries.

The optimization algorithm of [86] uses extra ancillary lines and extra quantum gates to reduce the quantum cost of a reversible circuit. The reversible gate operations are performed by less expensive gates on the added ancillary lines. The approach uses Fredkin gates to move the values from the main lines to the ancillary lines, and quantum gates are added to separate the Boolean and quantum domains.

While ancillaries are often used to simplify reversible and quantum circuits, on the other hand, some efforts have been made to reduce the number of ancillary lines in reversible circuits *e.g.* [87]. This is due to the limitation in the number of qubits in quantum systems as described in the previous chapter.

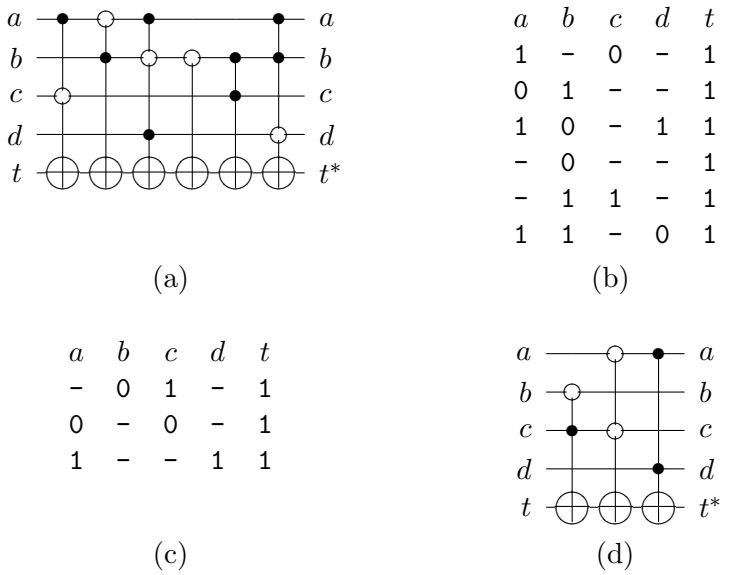



Figure 3.4: Circuit optimization based on ESOP minimization

### 3.1.3 ESOP-based Optimization

A *literal* is a Boolean variable in the negative or positive polarity. A product term composed of Boolean AND of literals is called a *cube*. An *Exclusive-Sum-Of-Products (ESOP)* is a representation form of a function that consists of the Exclusive-OR of one or more cubes [88]. There are many approaches to ESOP minimization in the literature [88–91]. Using ESOP minimization, it is possible to optimize a reversible circuit.

The general idea is illustrated in Figure 3.4 (a). The six MPMCT gates have the same target $t$. This circuit can be expressed as an ESOP as shown in the cube notation in Figure 3.4 (b). Note that each positive (negative) control leads to a positive (negative) literal in the respective cube. Each line that does not contain a control connection is represented by a don't-care in the cube. Using an ESOP minimization approach *e.g.* EXORCISM-4 [88], it is possible to reduce the ESOP as shown in Figure 3.4 (c). The circuit of Figure 3.4 (a) is thus optimized to the circuit shown in Figure 3.4 (d).

### 3.1.4 Miscellaneous

In [92], a fault model named the crosspoint fault model is used to identify redundant controls in reversible circuits. This paper shows that undetectable multiple disap-

pearance faults can be useful to simplify reversible circuits.

In [93], the optimization problem is addressed by defining a set of rules for removing NOT gates and optimizing subcircuits with common-target gates the application of which leads to quantum cost reduction. This approach minimizes the number of controls in a cascade using rules that are defined based on the Karnaugh map [94] representation of the circuit.

## 3.2 The New Moving Rule for Gate Rearrangement

Most often, optimization approaches use gate rearrangement to find possible reductions by matching subcircuits to templates or predefined rules. So far in the literature, gate rearrangement has usually been performed based on the *conventional moving rule* [28] that is described in the following property.

**Property 3.1. Conventional Moving Rule***: Two adjacent gates can be interchanged iff the target for each gate is not a control for the other gate* i.e. *in a reversible circuit, gate* $T(C_1, t_1)$ *can be interchanged with gate* $T(C_2, t_2)$ *iff* $C_1 \cap t_2 = \emptyset$ *and* $C_2 \cap t_1 = \emptyset$.

This moving rule preserves the functional behavior of a circuit while moving gates across it. Note that this assumes matrices describing the target operations for two gates that are interchanged and have a common target either commute or there is no condition under which both gates are applied. This assumption was not specified in [28] as all target matrices considered in that work commute. Note that all target matrices considered in this dissertation commute.

**Example 3.1.** *Consider the circuit of Figure 3.5 (a). According to the conventional moving rule, the first gate* $V(b; t)$ *cannot pass over the next gate* $T(a; b)$ *because its control is on the same line as the target of the next gate.*

The conventional moving rule is very restrictive and allows very small movement domains for gates. In this section, we show that breaking the conventional moving rule does not always change the functionality of a circuit. More precisely, it is possible to move a gate to certain places in a circuit provided that some conditions are met. We here present a new, more general approach to moving gates in cascades of reversible and quantum gates. The following definitions are used in the new moving rule.

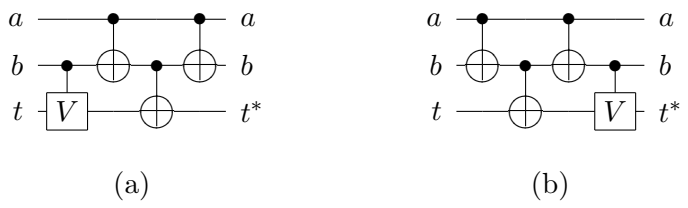



Figure 3.5: (a) The initial circuit. (b) The result of applying the new moving rule to the $V$ gate.

**Definition 3.1.** *A line is* **invariant** *with respect to a cascade of gates if it has the same function at each end of the cascade. Note that a line that is invariant with respect to a cascade can have different functionality inside the cascade.*

**Definition 3.2.** *A line is a* **non-controlling line** *across a cascade of gates if there is no control connection on that line within the cascade.*

Our new moving rule is shown in the following property which is a generalization of Property 3.1.

**Property 3.2. Generalized Moving Rule***: A gate can be moved from one end of a cascade of gates to the other end if its controls are on lines that are invariant with respect to the cascade and its target is on a non-controlling line.*

*Proof.* Since the functionality at both ends of an invariant line is the same, the cascade acts as an identity function on that line and a control connection can be passed over it. However, it is not possible to move a target over any control connection since it changes the value on the control connection. Hence a target can only be moved on a non-controlling line. □

Note that for a cascade consisting of a single gate, our generalized moving rule is equivalent to the conventional moving rule. Using our new rule, we can move a gate across a cascade of gates within a circuit as long as the conditions of Property 3.2 hold. The requirements for the commutativity of the target matrices is the same as for the conventional moving rule. Since all target matrices for gates considered in this dissertation commute, incorporating these requirements is left for future work using other quantum gate libraries.

**Example 3.2.** *In the circuit of Figure 3.5 (a), the second gate $T(a;b)$ changes the value of line $b$ to $a \oplus b$ and the fourth gate restores it to $b$. Hence, using the new*

*generalized moving rule, the first gate $V(b;t)$ can pass over all the gates to the right end of the cascade without changing the functionality as shown in Figure 3.5 (b).*

The above example shows how the new moving rule increases the movement domain of gates which can significantly contribute to finding more reductions in optimization procedures. In the next section, we have introduced an optimization procedure that employs the new moving rule to find redundant gates inside a circuit. However, note that the new moving rule is applicable to all the optimization approaches described in Section 3.1.

## 3.3 The New optimization procedure

The following procedure optimizes a given circuit using our generalized moving rule for gate rearrangement. In this procedure, the circuit is considered as a cascade of $G_1, G_2, ..., G_n$ gates where $n$ is the number of gates in the circuit.

**Procedure 3.1. Gate Reduction Procedure (GRP)**

For $p = 1$ to $n$ apply the following steps:

1. Keep track of the functional changes on each circuit line after the gate $G_p$.
2. If functions on the circuit lines after $G_p$ are all identity (*i.e.* are equal to the functions on the primary inputs side), remove the cascade of gates $G_1 \dots G_p$ from the circuit, set $p = 1$ and go to 1.
3. For $q = 1$ to $p - 1$, if the functions on the circuit lines after $G_q$ match the functions on the corresponding lines after $G_p$, remove the cascade of gates $G_{q+1} \dots G_p$ from the circuit, set $p = q + 1$ and go to 1.
4. $ReductionList = G_p$.
5. k=0.
6. For $q = p - 1$ to 1, if $G_p$ can be made adjacent to $G_q$ using the generalized moving rule (Property 3.2) and the data from step1, and the pair $G_qG_p$ can be reduced:
   (a) Remove $G_q$ from the circuit and add it to the reduction list *i.e* $ReductionList = G_q \dots G_p$.

(b) k=q.

7. If $size(ReductionList) > 1$:

   (a) Remove $G_p$ from the circuit.

   (b) Reduce the *ReductionList* to an optimal cascade.

   (c) Add the reduced *ReductionList* to the place of the leftmost gate in the original *ReductionList*, k.

   (d) Set $p = k + size(ReductionList)$ and go to 1.

The above procedure applies the optimization from one end of a circuit (input side) to the other end (output side) by considering one gate at a time and trying to find possible reductions that may occur in the circuit before the selected gate. For each gate $G_p$, the functions on the circuit lines after the gate are stored (this part will be discussed in detail in the following section and the next chapter). Then the procedure applies two major tests. First, it tries to find cascades of gates in the circuit that realize the identity function and can be removed from the circuit. This test is done by comparing the functions on circuit lines after the gate $G_p$ with the functions on the corresponding lines at each spot on the circuit from primary inputs to $G_{p-1}$. Steps 2 and 3 of the procedure perform this part.

In the next part, the procedure moves the selected gate backwards through the circuit while considering more than two gates at a time to find possible reductions. As a gate ($G_p$) is moved across the circuit, a list is made that contains gates that can be made adjacent to $G_p$ and together with $G_p$ can be reduced to have less quantum cost. Then, the gates in this list are removed from the circuit and an optimized equivalent cascade is inserted in the position of the left-most removed gate in the circuit. The optimized counterpart may be empty which indicates that the corresponding set of gates realizes the identity function.

The Gate Reduction Procedure (GRP) (Procedure 3.1) uses the new generalized moving rule to move gates backwards through the circuit and find possible reductions. Using the new moving rule implies identifying the invariant lines between the gate cascades which requires keeping track of the functions on line segments. This is the first step (step 1) of the Gate Reduction Procedure and is in fact a difficult problem. In the following section, we show our first solution to the problem of keeping track of the functionalities. In the next chapter, a more comprehensive approach to this problem will be discussed.

### 3.3.1 Line Labeling Procedure

In order to identify gate cascades for which a line is invariant, we use the following procedure to label the line segments in a circuit. This labeling is such that if two segments of a circuit line have the same label, those segments realize the same function. However, this procedure does not, in general, find all the line segments that have the same functionality.

**Procedure 3.2. Line Labeling Procedure (LLP)**

1. To begin, all circuit input lines are labeled 0 and an empty stack $S_i$ is created for each circuit line. Each stack $S_i$ keeps a record of the gates with target line $i$.

2. For each gate $G$ from the inputs towards the outputs of the circuit, apply the following steps where $t$ identifies the target of $G$:

   (a) Clear the *increase* flag.

   (b) If there are $k$ gates $H_1, ..., H_k$ at the top of $S_t$ that realize the identity function by adding $G$, *i.e.* $H_1...H_kG = I$, and their control lines are labeled the same, then the output side of the target for $G$ is assigned the label on the input side of the target for $H_1$. Otherwise, the *increase* flag is set.

   (c) If there are $m$ gates $H_1, ..., H_m$ at the top of $S_t$ ($m \geq 2$) which together realize the identity function, *i.e.* $H_1...H_m = I$, and their control lines are labeled the same, then the $m$ gates involved are pulled from $S_t$ and the procedure goes back to step (a).

   (d) If the *increase* flag is not set go to (f).

   (e) If there is a gate $H$ in the circuit which is the same as (inverse of) $G$ and each control line is labeled the same at $G$ and $H$ and the target line is labeled the same at the input (output) side of $H$ and input side of $G$, then the output side of the target for $G$ is assigned the label on the output (input) side of $H$. Otherwise, the output side label of $t$ for $G$ is set to the maximum label used on that line thus far plus 1.

   (f) Gate $G$ is pushed onto stack $S_t$.

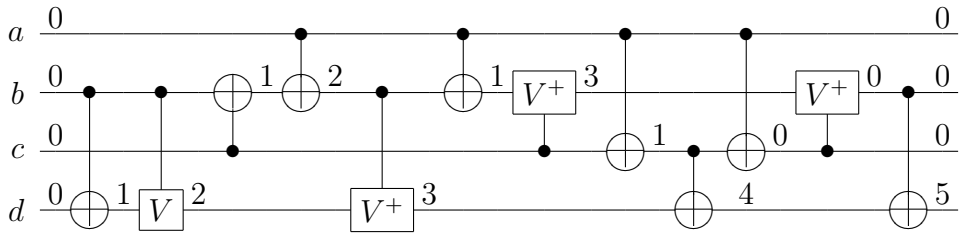



Figure 3.6: Example of circuit labeling

In the above procedure, labels are increased for targets which change the functionality, and are restored to a prior value when a set of gates realizing the identity operation is found. Overlapping and nested cascades that realize the identity function are found by this procedure.

**Example 3.3.** *Consider the circuit shown in Figure 3.6. The initial labeling is* $(a, b, c, d) = (0, 0, 0, 0)$. *For the first gate* $T(b; d)$, *the target line* $d$ *is given the label 1 (changes are shown to the right side of each gate in the Figure) and the gate is pushed onto stack* $S_d$. *The resulting labeling is* $(0, 0, 0, 1)$. *Next, the gate* $V(b; d)$ *is compared with the gate at the top of* $S_d$, $V(b; d)$, *and since they are not the inverse of each other, the label of* $d$ *after this gate is set to 2 and V(b;d) is pushed onto stack* $S_d$. *In a similar manner, gates* $T(c; b)$ *and* $T(a; b)$ *are pushed onto stack* $S_b$ *and the labeling becomes* $(0, 2, 0, 2)$. *For gate* $V^+(b; d)$ *the labeling becomes* $(0, 2, 0, 3)$ *and the gate is pushed onto stack* $S_d$.

*The next gate is* $T(a; b)$ *which is the inverse of the gate at the top of* $S_b$. *The control labels match and the output target label for the left occurrence matches the input target label for the right occurrence. Hence, the output target label for the right gate is set to 1, the input target label for the left occurrence. The gate is then pushed onto the stack* $S_b$.

*The procedure continues pushing the left* $V^+(c; b)$ *onto* $S_b$, $T(a; c)$ *onto* $S_c$, $T(c; d)$ *onto* $S_d$, *and* $T(a; c)$ *onto stack* $S_c$, *while updating the labeling as shown. Since the last two gates on* $S_c$ *are inverse of each other, they are popped from* $S_c$ *and* $V^+(c; b)$ *is pushed onto* $S_b$. *Next, it is found that the three gates at the top of* $S_b$ *give the identity and the labels satisfy the requirements. The three gates are popped from* $S_b$ *and the labeling becomes* $(0, 0, 0, 4)$. *Finally, the gate* $T(b; d)$ *changes the labeling to* $(0, 0, 0, 5)$.

*Applying Property 3.2, either occurrence of* $T(b; d)$ *can be moved over the cascade of gates in between to be next to the other one and the two gates cancel. Note that*

*this reduction is not possible using the conventional moving rule (Property 3.1).*

The procedure described above can be used in Step 1 of our optimization procedure (GRP) and labels can be used to compare functions in Steps 2 and 3. More precisely, the first step of the Line Labeling Procedure is performed once at the beginning of the optimization. Then, each time the optimization procedure reaches its Step 1, Step 2 of the Line Labeling Procedure is performed to identify the labels after a gate $G_p$. Thus, gate reductions will be identified as the circuit is labeled, and the labeling restarts as necessary. Combining labeling and gate reduction in this way is considerably more efficient than applying them as separate processes.

The following sections consider the application of the above methods to optimizing circuits composed of MPMCT and NCV gates.

### 3.3.2 Optimization of MPMCT Circuits

This section provides the application of the optimization procedures introduced in the previous section in a very common type of reversible circuits called MPMCT circuits. The following property shows the reduction rules that can be applied to MPMCT gates.

**Property 3.3.** *The MPMCT optimization procedure applies a number of* **reduction rules***:*

1. $T(C; x_t)T(C; x_t) = I$

2. $T(C; x_t)T(C \cup x_i; x_t) = T(C \cup \overline{x_i}; x_t)$

3. $T(C \cup x_i; x_t)T(C \cup x_j; x_t) = T(x_i; x_j)T(C \cup x_j; x_t)T(x_i; x_j)$

4. $T(C \cup \overline{x_i}; x_t)T(C \cup \overline{x_j}; x_t) = T(x_i; x_j)T(C \cup x_j; x_t)T(x_i; x_j)$

5. $T(C \cup x_i; x_t)T(C \cup \overline{x_j}; x_t) = T(x_i; x_j)T(C \cup \overline{x_j}; x_t)T(x_i; x_j)$

The first rule comes from the fact that MPMCT gates are self-inverse. The second rule is for two adjacent MPMCT gates that are different only in one control connection. Figure 3.7(a) and Figure 3.7(b) illustrate the possible reductions in this case. Reduction rules 3 to 5 are illustrated in Figure 3.7(c) and Figure 3.7(d). Note that

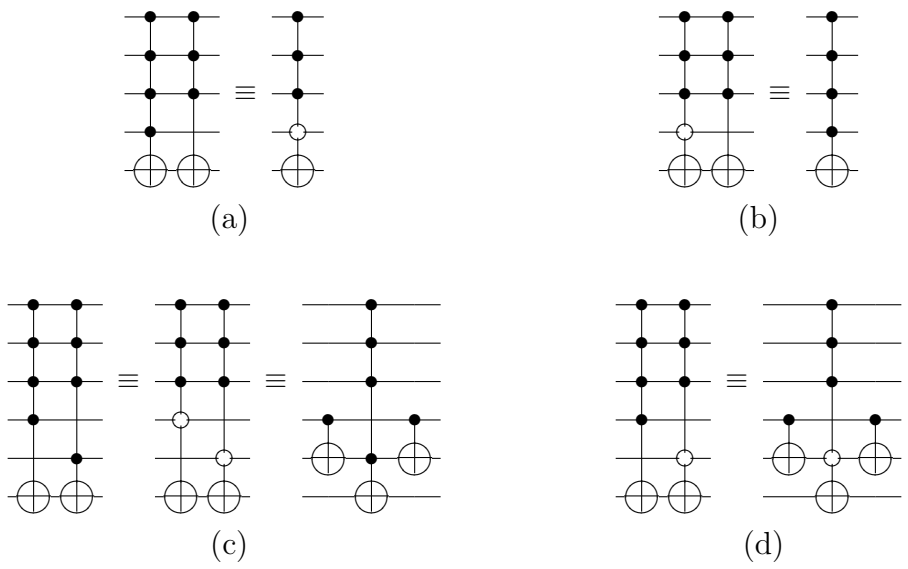



Figure 3.7: MPMCT Reduction Rules

Note: In the illustrations above, one can change all controls on any of the top three lines to negative.

the circuit on the righthand side of rules 3 to 5 has more gates than the left cascade, but it has less quantum cost since two of the gates (two CNOTs) are elementary gates and there is only one complex gate among them.

To optimize an MPMCT circuit, the above rules are employed in Step 4 of the Gate Reduction Procedure. The procedure starts from one end of the circuit and labels one MPMCT gate at a time using the circuit Line Labeling Procedure. Then, it moves that gate back through the circuit as far as possible to find the best reduction. The gate may either be canceled using Rule 1 or may be reduced to a less expensive cascade using Rules 2 to 5. After a reduction is applied, the optimization restarts from the position of the earliest gate in the substituted cascade.

**Example 3.4.** *Consider the MPMCT circuit of Figure 3.8 (a). All circuit lines are first initialized to 0. The optimization procedure then labels the first gate* $T(e; a)$*. Note that only changes on labels are shown in the figure for simplicity. There is no gate before this gate, so the optimization procedure proceeds to the next gate* $T(a, b, \overline{c}; e)$ *and labels the lines after that. Since the target of this gate is on the control of its previous gate, it cannot be moved backwards and the procedure continues accordingly until it reaches the sixth gate* $T(a, b, \overline{d}; e)$*. After labeling the line segments after this gate, the gate is moved backwards through the circuit. The gate* $T(a, b, \overline{d}; e)$ *can be*

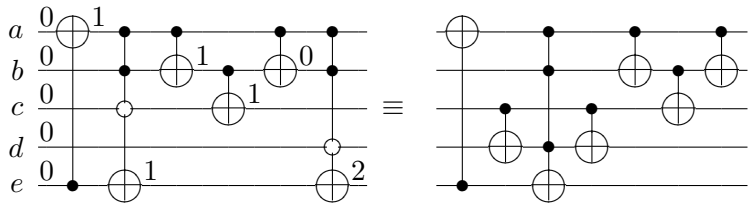



Figure 3.8: Example of MPMCT optimization.

*placed after the second gate $T(a, b, \overline{c}; e)$ since its controls have the same labels as the lines after the second gate and its target does not pass over a control connection as it is moved. The two MPMCT gates are added to the ReductionList and further reduced to $T(c; d)T(a, b, d; e)T(c; d)$ according to the rule 4 in property 3.3. The reduced cascade is then replaced with the pair of MPMCT gates and the optimization procedure restarts from the next gate. The optimized circuit is shown in Figure 3.8 (b). Considering the REVLIB cost model [9], the MPMCT optimization reduces the cost of the circuit of Figure 3.8 (a) from 32 to 20. Note that this reduction was not possible using the old moving rule (Property 3.1).*

### 3.3.3 Optimization of Quantum Circuits

In this section, we discuss the application of the Gate Reduction Procedure and Line Labeling Procedure in optimizing a class of quantum circuits consisting of gates realizing $2^k$ roots of NOT, $k > 1$. The general approach is described while explaining details for quantum gates up to the fourth root of the NOT gate. This library is called NCVW and contains NOT, CNOT, $V$, $V^+$, a fourth root of NOT, $W$, and its inverse, $W^+$. This library is studied in details in Chapter 5.

**Property 3.4.** *The following reductions are possible in the NCVW library:*

1. $NOT.NOT = I$
2. $CNOT.CNOT = I$
3. $VV^+ = V^+V = I$
4. $WW^+ = W^+W = I$
5. $VV = V^+V^+ = NOT$

Table 3.1: Weights of the NCVW gates

| **Gate** | **Weight** |
|---|---|
| $W$ | 1 |
| $W^+$ | -1 |
| $V$ | 2 |
| $V^+$ | -2 |
| $CNOT$ | 4 |

6. $WW = V$
7. $W^+W^+ = V^+$
8. $V.NOT = V^+$
9. $V^+.NOT = V$
10. $WV^+ = W^+$
11. $W^+V = W$

We use a simple, yet interesting approach to find the above cancelations and to simplify gate sequences. In this approach, a weight is assigned to each gate in the library. Suppose the highest root of $NOT$ in our library is $R_k$ in which $k$ is a power of 2 ($k = 2^r$). In this case, for any gate $R_i$, $R_i^i = N$ and $i = \{1, 2, ..., k\}$, in the library, the assigned weight would be $k/i$ as indicated in Table 3.1 for the NCVW library. Then, in order to simplify a sequence of adjacent gates with the same control and target, the weights of the gates are added and the result is divided by $2^{r+1}$. The residue of this division is used as an index to find the replacement sequence that is shown in Table 3.2 for the NCVW library. This table is built by assigning values 0 to $2^{r+1} - 1$ to the sub-circuits corresponding to the possible rotations counter clockwise.

The optimization procedure works as follows. As a gate $G$ is labeled by the Line Labeling Procedure, it is moved back through the circuit to each of the places that have the same labels on its control lines. Gate $G$ cannot be moved past a point where there is a control on the target line of $G$. As a gate is moved, a list is made that contains gates that can be adjacent to it and have the same target, control and label on the control. Then, the gates in the list are removed from the circuit and using the above approach, the list is optimized and the reduced cascade is placed in the position of the leftmost gate in the original list in the circuit.

Table 3.2: Reduced sequences of the NCVW gates

| Index | Sequence |
|---|---|
| 0 | $I$ |
| 1 | $W$ |
| 2 | $V$ |
| 3 | $VW$ |
| 4 | $CNOT$ |
| 5 | $V^+W^+$ |
| 6 | $V^+$ |
| 7 | $W^+$ |

**Example 3.5.** *Consider the circuit of Figure 3.6. After the optimization procedure labels the second gate from left $V(b;d)$, it adds the first two gates to the reduction list. The sum of the corresponding weights from Table 3.1 mod 8* i.e. $(4+2)$ *mod 8 gives the index of the equivalent optimal subcircuit in Table 3.2. So the pair of gates is substituted by a $V^+(b;d)$ gate. The resulting circuit is shown in Figure 3.9 (a). The procedure then proceeds by labeling each gate at a time and trying to move it backwards until it reaches the tenth gate from the left $V^+(c;b)$. This gate is moved backwards and as it can be made adjacent to the sixth gate $V^+(c;b)$, a reduction list of $V^+(c;b)V^+(c;b)$ is created. The weights are added $(-2)+(-2)=-4$ and the result is divided by 8. The residue of the division 4 is an index to Table 3.2 that gives the optimal subcircuit for this list which is a CNOT. Finally the last gate $T(b;d)$ is moved back and added to a reduction list with the first gate $V^+(b;d)$. The pair is then reduced to a $V(b;d)$ gate as shown in Figure 3.9 (b). Note that every time a cascade is substituted, the labeling restarts from the substituted cascade.*

## 3.4 Experimental Results

Prototype implementations of the procedures described in this chapter have been implemented using Python 2.6.5. We have utilized Python dictionaries to implement step 2.($e$) of the circuit line labeling procedure (Procedure 3.2). We keep a dictionary of all previously seen gates with different labels on their controls. The amortized worst case time of finding an element in a dictionary in python is $O(n)$ where $n$ is the number of elements in the dictionary. Hence, the time complexity of the circuit line labeling procedure (Procedure 3.2) is $O(n^2)$ where $n$ is the number of gates in

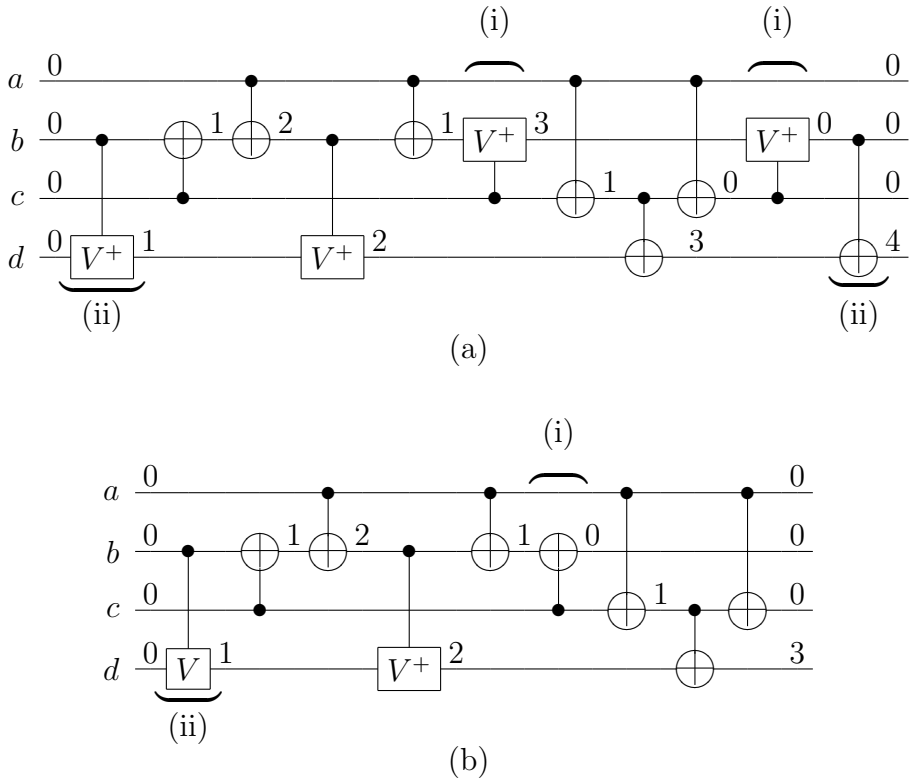



Figure 3.9: Example of NCV Optimization

the circuit. The time complexity of the gate reduction procedure (Procedure 3.1) is also $O(n^2)$.

The experiments were run on a system with a 3.2 GHz i5-650 CPU and 3.0 GB RAM. The evaluation was performed by applying the proposed optimization methods on three different test suites.

Our first test suite consists of 156 benchmark circuits taken from the REVLIB website [9]. Our prototype program uses REVLIB [9] *real* circuit format. The results are summarized in Table 3.3. Due to space limitations, the table includes only those circuits for which our approach resulted in a quantum gate cost reduction of 10% or more. For each circuit, the table gives the REVLIB circuit name and file identification number in the first column. The best known quantum costs given in REVLIB are reported in the second column. The REVLIB quantum costs are based on a library consisting of quantum gates implementing roots of the NOT gate. This library generates exponential quantum costs for individual reversible gates with size $n$ in an $n$-line circuit. To avoid the exponential realizations and higher order roots

of the NOT gate, and to keep with the work in the literature (most of the literature work use NCV quantum costs), an ancillary line is added to circuits that have $n$ lines and at least one reversible gate with size $n$ [71]. These circuits are identified by a * after the name.

The column labeled *Initial Cost* shows the NCV costs of our initial reversible circuits taken from REVLIB. These circuits are obtained by substituting the REVLIB NCV realizations for each MPMCT gate. Negative controls in MPMCT gates are handled by adding NOT gates at both ends of the corresponding lines in the realizations. The third column differs from the second column for the following reasons. First, since we only consider quantum gates NOT, CNOT, controlled-$V$ and controlled-$V^+$, the cost of a 3-control MCT gate is 14 [71] and not 13 as used in REVLIB. For the same reason, the cost of a 5-control MCT gate with one available ancillary is 26 instead of 29 as in REVLIB, and an 8-control MCT gate is realized with 104 quantum gates whereas its cost is 100 in REVLIB. For most circuits, our initial cost is lower than the REVLIB cost; however, this is not always the case.

The column *MCT Opt.* gives NCV quantum costs after our MCT optimization described in Section 3.3.2. As it is indicated in Property 3.3, during the optimization sometimes MCT gates with positive controls are replaced with negative control MCT gates. This is beneficial if negative controls have the same cost as positive controls. In the future chapters we show how this can be achieved. However, to keep with the REVLIB cost model in this chapter, we add two NOT gates for each negative control that sometimes makes the result of the MPMCT optimization worse than the original circuits. Note that using better cost functions, these circuits are realizable with lower quantum costs.

The column *Map to NCV* gives the quantum cost of the circuits after mapping them to NCV circuits using REVLIB decomposition structures [84]. Note that the quantum costs are the same as the previous column as mapping is performed by substituting individual reversible gates with their quantum counterparts.

The column *NCV Opt.* shows the final quantum costs (number of quantum gates) for the circuits after applying our quantum optimization procedure described in Section 3.3.3. The *%Imp. wrt Initial Cost* shows the percentage improvement for the quantum cost of the final circuits with respect to the quantum costs of the initial circuits. Note that this percentage is higher when comparing to the costs given in REVLIB; however, for the sake of showing the effectiveness of our optimization methods, the results are compared to the initial circuits.

The CPU times given in the last column are the total elapsed time for the MCT and the NCV optimization procedures. These times were determined using the Python time function. They show the relative performance of the program for various circuits but should not be given too much weight as our implementation has not been optimized and uses many heuristics that could be further developed.

For the circuits listed in Table 3.3, the cumulative cost reduction is 18.03% and the average improvement is 20.79%. The cumulative cost reduction for all the 156 benchmark circuits is 16.29%. The difference comes from the fact that there are a number of small circuits in the test suite that are already highly optimized. The average improvement for all the circuits in the test suite is 4.8%. As expected, the results in Table 3.3 show that the cost reduction resulting from applying our optimization procedures is very circuit dependent. The initial benchmark circuits have already gone through different optimizations and are quite optimized. However, by changing the optimization constraints such as the gate moving rule, we showed that more cancelations can be found. All circuits generated in our experiments have been verified using the QMDD-based verification approach described in [95].

Our second test suite consists of the same benchmarks that have been run with the added optimization of factoring the MCT gates with a single added helper line as described in [85]. This experiment shows how our methods can be combined with other optimization techniques to yield better circuits. The results are shown in Table 3.4 for the circuits listed in Table 3.3. Note that for this experiment, a single helper line has been added to every circuit. In this case the cumulative cost reduction for the circuits in Table 3.4 is 21.44% compared to the initial costs and the improvement is 27.04% on average.

Our third test suit is a selection of benchmark functions with 10 to 15 lines as considered in [44]. These functions are shown in Table 3.5. For each function, we considered three synthesis methods: TBS [40], RMS [29], and QMDD-based [44]. TBS and RMS produce circuits with only MCT gates. QMDD-based synthesis produces circuits with MPMCT gates. We applied our optimization methods to the circuits resulting from each synthesis method. The results are summarized in Table 3.5.

For each function and synthesis method we show the quantum gate count (GC) for three cases:

**Case (a)** The MPMCT gates in the circuit produced by the synthesis method are individually mapped to quantum gates using the approach described for the first test suite that employs the REVLIB quantum cost function for the NCV library

Table 3.3: Optimization results on REVLIB benchmark circuits.

| | **REVLIB Cost** | **Initial Cost** | **MCT Opt.** | **Map to NCV** | **NCV Opt.** | **%Imp. wrt initial Cost** | **Total Time (s)** |
|---|---|---|---|---|---|---|---|
| sym9_148 | 4368 | 4452 | 798 | 798 | 798 | 82.08 | 0.467 |
| sym6_145 | 777 | 777 | 297 | 297 | 283 | 63.58 | 0.048 |
| 4gt4-v0_73* | 89 | 89 | 57 | 57 | 55 | 38.20 | 0.004 |
| 4mod5-v0_18 | 25 | 25 | 19 | 19 | 17 | 32.00 | 0.019 |
| 4mod5-v0_21 | 19 | 25 | 19 | 19 | 17 | 32.00 | 0.001 |
| toffoli_double_4 | 10 | 10 | 7 | 7 | 7 | 30.00 | 0.001 |
| 4gt12-v0_86* | 58 | 56 | 44 | 44 | 42 | 25.00 | 0.017 |
| rd53_133 | 128 | 128 | 104 | 104 | 98 | 23.44 | 0.017 |
| rd53_134 | 120 | 128 | 104 | 104 | 98 | 23.44 | 0.001 |
| ham15_107 | 1831 | 1856 | 1434 | 1434 | 1426 | 23.17 | 0.163 |
| 4gt12-v0_87* | 54 | 52 | 40 | 40 | 40 | 23.08 | 0.001 |
| 4gt4-v0_72* | 54 | 52 | 40 | 40 | 40 | 23.08 | 0.001 |
| urf4 | 160020 | 160020 | 136534 | 136534 | 127692 | 20.20 | 10572.489 |
| mod5d2_70 | 16 | 16 | 13 | 13 | 13 | 18.75 | 0.000 |
| 4gt13_90 | 34 | 35 | 32 | 32 | 29 | 17.14 | 0.002 |
| rd32-v0_66 | 12 | 12 | 12 | 12 | 10 | 16.67 | 0.000 |
| rd32-v0_67 | 8 | 12 | 12 | 12 | 10 | 16.67 | 0.000 |
| rd73_140 | 76 | 76 | 76 | 76 | 64 | 15.79 | 0.001 |
| rd73_141 | 64 | 76 | 76 | 76 | 64 | 15.79 | 0.017 |
| sym9_146 | 108 | 108 | 108 | 108 | 91 | 15.74 | 0.033 |
| sym9_147 | 94 | 108 | 108 | 108 | 91 | 15.74 | 0.016 |
| 4mod5-v0_19 | 13 | 13 | 11 | 11 | 11 | 15.38 | 0.001 |
| mod5mils_65 | 13 | 13 | 11 | 11 | 11 | 15.38 | 0.000 |
| mod5mils_71 | 13 | 13 | 11 | 11 | 11 | 15.38 | 0.000 |
| rd32-v1_68 | 13 | 13 | 13 | 13 | 11 | 15.38 | 0.000 |
| rd32-v1_69 | 9 | 13 | 13 | 13 | 11 | 15.38 | 0.000 |
| alu-v2_31 | 101 | 107 | 91 | 91 | 91 | 14.95 | 0.002 |
| hwb4_49* | 65 | 67 | 60 | 60 | 57 | 14.93 | 0.002 |
| hwb4_50* | 63 | 67 | 60 | 60 | 57 | 14.93 | 0.001 |
| plus63mod8192_164* | 45025 | 37102 | 32018 | 32018 | 31606 | 14.81 | 61.255 |
| one-two-three-v1_99 | 36 | 37 | 34 | 34 | 32 | 13.51 | 0.000 |
| rd84_142 | 112 | 112 | 112 | 112 | 97 | 13.39 | 0.017 |
| rd84_143 | 98 | 112 | 112 | 112 | 97 | 13.39 | 0.017 |
| 4gt11_82 | 16 | 16 | 16 | 16 | 14 | 12.50 | 0.017 |
| sys6-v0_111 | 72 | 72 | 72 | 72 | 63 | 12.50 | 0.016 |
| sys6-v0_144 | 62 | 72 | 72 | 72 | 63 | 12.50 | 0.002 |
| plus63mod4096_163* | 32539 | 25534 | 22532 | 22532 | 22406 | 12.25 | 11.776 |
| plus127mod8192_162* | 73357 | 65456 | 58336 | 58336 | 57754 | 11.77 | 145.439 |
| miller_11 | 17 | 17 | 17 | 17 | 15 | 11.76 | 0.001 |
| 4gt5_76 | 29 | 30 | 27 | 27 | 27 | 10.00 | 0.002 |
| **Total** | **319618** | **296979** | **253552** | **253552** | **243419** | ** | **10791.844** |

* An additional line is added for mapping to the NCV library.

** Cumulative Improvement: 18.03%, Average Improvement: 20.79%

Table 3.4: Optimization of REVLIB benchmark circuits with an additional helper line.

| | **REVLIB Cost** | **Initial Cost** | **MCT Opt.** | **Map to NCV** | **NCV Opt.** | **%Imp. wrt initial Cost** | **Total Time (s)** |
|---|---|---|---|---|---|---|---|
| sym9_148 | 4368 | 4452 | 703 | 703 | 699 | 84.30 | 783.299 |
| sym6_145 | 777 | 777 | 244 | 244 | 239 | 69.24 | 308.241 |
| alu-v2_31 | 101 | 107 | 54 | 54 | 49 | 54.21 | 103.206 |
| 4gt4-v0_73 | 89 | 89 | 45 | 45 | 43 | 51.69 | 94.685 |
| ham15_107 | 1831 | 1856 | 1015 | 1015 | 1005 | 45.85 | 1050.851 |
| rd53_133 | 128 | 128 | 86 | 86 | 80 | 37.50 | 117.500 |
| rd53_134 | 120 | 128 | 86 | 86 | 80 | 37.50 | 117.500 |
| hwb4_49 | 65 | 67 | 51 | 51 | 42 | 37.31 | 79.313 |
| hwb4_50 | 63 | 67 | 51 | 51 | 42 | 37.31 | 79.313 |
| 4mod5-v0_18 | 25 | 25 | 19 | 19 | 17 | 32.00 | 49.000 |
| 4mod5-v0_21 | 19 | 25 | 19 | 19 | 17 | 32.00 | 49.000 |
| 4mod5-v0_19 | 13 | 13 | 10 | 10 | 9 | 30.77 | 39.769 |
| mod5mils_65 | 13 | 13 | 10 | 10 | 9 | 30.77 | 39.769 |
| mod5mils_71 | 13 | 13 | 10 | 10 | 9 | 30.77 | 39.769 |
| toffoli_double_4 | 10 | 10 | 7 | 7 | 7 | 30.00 | 37.000 |
| 4gt12-v0_86 | 58 | 56 | 42 | 42 | 40 | 28.57 | 68.571 |
| 4gt12-v0_87 | 54 | 52 | 38 | 38 | 38 | 26.92 | 64.923 |
| 4gt4-v0_72 | 54 | 52 | 38 | 38 | 38 | 26.92 | 64.923 |
| 4gt13_90 | 34 | 35 | 29 | 29 | 26 | 25.71 | 51.714 |
| plus63mod8192_164 | 45025 | 37102 | 28170 | 28170 | 27902 | 24.80 | 27926.797 |
| plus63mod4096_163 | 32539 | 25534 | 19557 | 19557 | 19499 | 23.64 | 19522.635 |
| plus127mod8192_162 | 73357 | 65456 | 51771 | 51771 | 51413 | 21.45 | 51434.454 |
| mod5d2_70 | 16 | 16 | 13 | 13 | 13 | 18.75 | 31.750 |
| urf4 | 160020 | 160020 | 140471 | 140471 | 131238 | 17.99 | 131255.987 |
| rd32-v0_66 | 12 | 12 | 12 | 12 | 10 | 16.67 | 26.667 |
| rd32-v0_67 | 8 | 12 | 12 | 12 | 10 | 16.67 | 26.667 |
| rd73_140 | 76 | 76 | 76 | 76 | 64 | 15.79 | 79.789 |
| rd73_141 | 64 | 76 | 76 | 76 | 64 | 15.79 | 79.789 |
| sym9_146 | 108 | 108 | 108 | 108 | 91 | 15.74 | 106.741 |
| sym9_147 | 94 | 108 | 108 | 108 | 91 | 15.74 | 106.741 |
| rd32-v1_68 | 13 | 13 | 13 | 13 | 11 | 15.38 | 26.385 |
| rd32-v1_69 | 9 | 13 | 13 | 13 | 11 | 15.38 | 26.385 |
| one-two-three-v1_99 | 36 | 37 | 34 | 34 | 32 | 13.51 | 45.514 |
| 4gt5_76 | 29 | 30 | 27 | 27 | 26 | 13.33 | 39.333 |
| 4gt11_82 | 16 | 16 | 16 | 16 | 14 | 12.50 | 26.500 |
| rd84_142 | 112 | 112 | 112 | 112 | 98 | 12.50 | 110.500 |
| rd84_143 | 98 | 112 | 112 | 112 | 98 | 12.50 | 110.500 |
| miller_11 | 17 | 17 | 17 | 17 | 15 | 11.76 | 26.765 |
| sys6-v0_111 | 72 | 72 | 72 | 72 | 64 | 11.11 | 75.111 |
| sys6-v0_144 | 62 | 72 | 72 | 72 | 64 | 11.11 | 75.111 |
| **Total** | **319618** | **296979** | **243419** | **243419** | **233317** | ** | **234398.468** |

** Cumulative Improvement: 21.44%, Average Improvement: 27.04%

(containing NOT, CNOT, $V$, and $V^+$ gates). The number of quantum gates required for each MPMCT gate depends on the number of controls and the number of ancillary lines available.

**Case (b)** In this case, the MPMCT gate optimizations presented in Section 3.3.2 are applied to the circuit produced by the synthesis method and the quantum gate count is then determined as in Case (a).

**Case (c)** In this case, the optimized circuit from Case (b) is mapped to the NCV library. The mapped circuit is then optimized using our quantum gate optimization described in Section 3.3.3. The gate count is the number of gates following that optimization.

**Time (s)** This column shows the total optimization time in seconds. The times were determined using the Python time function.

Gate count is, of course, only a rough estimate of the complexity of the circuit. A more accurate quantum cost model would be highly technology dependent. Note that we assume there is always at least one ancillary line available at every gate. Hence, if a circuit contains a gate that uses every line as a control or the target, we assume one extra line is added to the circuit. This is required to ensure a circuit can be built from NCV gates. As a result, higher order roots-of-NOT gates are not used in cost calculations. For each circuit, only one extra line needs to be added as ancillaries can be reused. These circuits are identified by an * after their name. As for the previous results, all circuits generated in Table 3.5 have been verified using the QMDD-based verification approach described in [95].

The results clearly show the advances of the proposed approaches. Using the MPMCT optimization presented in Section 3.3.2, average reductions in the gate count by approximately 13.2%, 25.2%, and 17.7% can be achieved for the circuits resulting from the TBS [40], RMS [29], and QMDD-based [44] synthesis methods respectively (see columns labeled with Case (b)). This can be further improved to 16.5%, 27%, and 24.8% if the methods from Section 3.3.3 are applied (see columns labeled with Case (c)). Thus, we can conclude that the optimization techniques described in this chapter are viable approaches to producing more efficient quantum circuit realizations for reversible functions than can be found using earlier techniques.

## 3.5 Summary

Post synthesis optimizations are commonly used in reversible and quantum circuit design automation. In this chapter, we moved the optimization techniques forward by relaxing the constraints that are applied for gate movements. In particular, we introduced a more general moving rule that allows larger movement domains for gates and results in finding more reductions. The experimental results show that the techniques presented in this chapter can result in considerable cost savings for NCV realization of MPMCT gate circuits.

The current implementation of the Gate Reduction Procedure (Procedure 3.1) that uses the Line Labeling Procedure (Procedure 3.2) as its first step does not guarantee to find all places in a circuit where a gate can be moved to and hence does not guarantee to find all possible reductions. However, in the next chapter, we will show that using a functional approach, it is possible to find all such cancelations. Note that besides being simple and straightforward, the Line Labeling procedure is fast and gives a good balance of final circuit cost versus computational cost.

The techniques presented can be applied to other reversible and quantum gate libraries. We are working on extending the methods to rotation and various other quantum gates [13]. In the approach discussed here, gates are only moved when they can be canceled or combined with others. However, we have found examples where moving a gate, or gates, allows for a reduction involving other gates. We are exploring this and looking for heuristics to guide when the movement of a gate will be advantageous.

The methods presented use heuristic greedy techniques. Our future work will consider alternatives. Finally, as shown in our second set of experimental results, the techniques presented in this chapter can be combined with other optimization approaches to reduce quantum implementation costs even further.

Table 3.5: Optimization results for a selected set of MPMCT circuits obtained by three different synthesis methods.

| Function | n | TBS [40] | | | | RMS [29] | | | | QMDD [44] | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Case (a) GC | Case (b) GC(%) | Case (c) GC(%) | Time (s) | Case (a) GC | Case (b) GC(%) | Case (c) GC(%) | Time (s) | Case (a) GC | Case (b) GC(%) | Case (c) GC(%) | Time (s) |
| max46_240 | 10 | 6091 | 5571(8.5) | 5050(17.1) | 12.075 | 4566 | 4072(10.8) | 3720(18.5) | 5.722 | 4566 | 4072(10.8) | 3720(18.5) | 6.646 |
| rd73_252 | 10 | 5547 | 4952(10.7) | 4666(15.9) | 3.164 | 1079 | 542(49.8) | 540(50.0) | 0.098 | 1116 | 1071(4.0) | 965(13.5) | 0.384 |
| sqn_258 | 10 | 6484 | 5645(12.9) | 5252(19.0) | 9.630 | 1770 | 1339(24.4) | 1314(25.8) | 0.301 | 1712 | 1448(15.4) | 1270(25.8) | 1.094 |
| sym9_193* | 10 | 6019 | 5224(13.2) | 4854(19.4) | 10.172 | 5857 | 5515(5.8) | 5121(12.6) | 10.206 | 5857 | 5515(5.8) | 5121(12.6) | 10.106 |
| dc1_220* | 11 | 10467 | 9728(7.1) | 9446(9.8) | 10.921 | 155 | 143(7.7) | 143(7.7) | 0.025 | 305 | 305(0.0) | 265(13.1) | 0.063 |
| wim_266 | 11 | 24054 | 21487(10.7) | 20876(13.2) | 59.986 | 181 | 171(5.5) | 171(5.5) | 0.023 | 224 | 207(7.6) | 183(18.3) | 0.049 |
| z4_268 | 11 | 1093 | 719(34.2) | 717(34.4) | 0.079 | 473 | 162(65.8) | 157(66.8) | 0.028 | 669 | 522(22.0) | 494(26.2) | 0.089 |
| cm152a_212 | 12 | 289 | 286(1.0) | 263(9.0) | 0.048 | 235 | 235(0.0) | 219(6.8) | 0.033 | 235 | 235(0.0) | 219(6.8) | 0.064 |
| cycle10_2_110 | 12 | 1212 | 1212(0.0) | 1176(3.0) | 0.235 | 1212 | 1212(0.0) | 1176(3.0) | 0.267 | 2134 | 1390(34.9) | 1242(41.8) | 0.773 |
| plus63mod4096_163* | 12 | 1010 | 1010(0.0) | 926(8.3) | 0.454 | 2640 | 2640(0.0) | 2598(1.6) | 0.363 | 1040 | 1040(0.0) | 928(10.8) | 0.595 |
| rd84_253 | 12 | 11253 | 8761(22.1) | 8485(24.6) | 6.351 | 2203 | 1114(49.4) | 1102(50.0) | 0.290 | 2273 | 2111(7.1) | 1953(14.1) | 1.449 |
| sqrt8_260* | 12 | 44334 | 37567(15.3) | 37367(15.7) | 29.403 | 1556 | 1011(35.0) | 953(38.8) | 0.305 | 677 | 621(8.3) | 529(21.9) | 0.269 |
| adr4_197 | 13 | 5231 | 3326(36.4) | 3299(36.9) | 1.364 | 639 | 219(65.7) | 217(66.0) | 0.029 | 635 | 179(71.8) | 177(72.1) | 0.065 |
| dist_223 | 13 | 40279 | 35169(12.7) | 34897(13.4) | 32.854 | 6439 | 5746(10.8) | 5729(11.0) | 1.792 | 5726 | 5150(10.1) | 4953(13.5) | 4.199 |
| plus127mod8192_162* | 13 | 1246 | 1246(0.0) | 1142(8.3) | 0.502 | 3717 | 3717(0.0) | 3637(2.2) | 0.912 | 1288 | 1288(0.0) | 1144(11.2) | 0.939 |
| plus63mod8192_164* | 13 | 1296 | 1296(0.0) | 1192(8.0) | 0.688 | 3582 | 3582(0.0) | 3502(2.2) | 0.913 | 1326 | 1326(0.0) | 1194(10.0) | 1.002 |
| radd_250 | 13 | 15160 | 11340(25.2) | 11286(25.6) | 3.828 | 632 | 174(72.5) | 174(72.5) | 0.028 | 680 | 256(62.4) | 254(62.6) | 0.029 |
| root_255* | 13 | 44569 | 37530(15.8) | 37134(16.7) | 49.300 | 5683 | 4011(29.4) | 3995(29.7) | 1.509 | 3259 | 2777(14.8) | 2636(19.1) | 1.944 |
| squar5_261 | 13 | 1872 | 1821(2.7) | 1812(3.2) | 0.204 | 260 | 234(10.0) | 233(10.4) | 0.019 | 361 | 344(4.7) | 339(6.1) | 0.050 |
| clip_206 | 14 | 90624 | 77118(14.9) | 75902(16.2) | 292.211 | 7757 | 4525(41.7) | 4507(41.9) | 1.544 | 5863 | 3698(36.9) | 3549(39.5) | 2.585 |
| cm42a_207* | 14 | 121688 | 100614(17.3) | 98332(19.2) | 854.385 | 225 | 181(19.6) | 180(20.0) | 0.039 | 324 | 324(0.0) | 288(11.1) | 0.110 |
| cm85a_209 | 14 | 23298 | 14980(35.7) | 14864(36.2) | 10.742 | 5113 | 3730(27.0) | 3701(27.6) | 0.554 | 2464 | 624(74.7) | 586(76.2) | 0.173 |
| sao2_257 | 14 | 90417 | 76797(15.1) | 76045(15.9) | 211.830 | 24382 | 13249(45.7) | 13053(46.5) | 23.335 | 6096 | 5632(7.6) | 5256(13.8) | 8.145 |
| co14_215 | 15 | 5490 | 5220(4.9) | 4997(9.0) | 6.133 | 3820 | 1930(49.5) | 1702(55.4) | 2.052 | 3820 | 1930(49.5) | 1702(55.4) | 2.008 |
| dc2_222 | 15 | 36371 | 31596(13.1) | 31271(14.0) | 36.637 | 2484 | 2222(10.5) | 2220(10.6) | 0.304 | 1775 | 1601(9.8) | 1461(17.7) | 1.006 |
| misex1_241 | 15 | 81924 | 69785(14.8) | 68064(16.9) | 456.408 | 831 | 678(18.4) | 676(18.7) | 0.140 | 824 | 799(3.0) | 725(12.0) | 0.380 |
| | | | **(13.2)** | **(16.5)** | | | **(25.2)** | **(27.0)** | | | **(17.7)** | **(24.8)** | |

* An additional line is added for mapping to the NCV library.
** The percentages are the improvements relative to case (a).

# Chapter 4

# Circuit Optimization: A Functional Approach

Reversible synthesis tools often generate circuits that are not optimal and can be improved by post synthesis optimizations [82, 86, 93]. Optimization methods usually use gate rearrangement to find possible reductions which is often performed based on gate moving rules. The conventional moving rule for reversible and quantum gates was proposed by Maslov [28]. A generalized gate moving rule (Property 3.2) was introduced in the previous chapter that allows larger movement domains for gates in reversible circuits. An optimization method (Procedure 3.1) was also proposed that uses the generalize moving rule to optimize reversible and quantum circuits.

To implement the new gate moving rule, a heuristic method (Procedure 3.2) was introduced in the previous chapter that employs labels to keep track of functional changes on qubit states between gates in a reversible circuit. Although this approach is fast, effective and straightforward, it cannot find all the possible reductions in a circuit.

In this chapter, we discuss an exact method that utilizes a functional description instead of labels to represent functions on circuit line segments. The functional representation that we use is a decision diagram structure called Decision Diagram for a Matrix Function (DDMF) [73, 96]. The limitation of this representation is that it is only applicable to Boolean reversible circuits and a certain class of quantum circuits called Semi-Classical Quantum Circuits(SCQC) [73, 96]. The experimental results show that the proposed approach can find more reductions than the Line Labeling Procedure (Procedure 3.2) that was introduced in the previous chapter. However,

due to the limitation of the new approach, it is applicable to Boolean reversible circuits but not all quantum circuits. The results of this part of the dissertation were published in [97, 98].

## 4.1 Basic Optimization Approach

Most optimization methods use gate rearrangement to find possible reductions in reversible and quantum circuits [2, 4, 93]. So far, the gate rearrangements have been performed under the limitations imposed by the conventional moving rule [28]. In the previous chapter, gate rearrangement constraints were modified in Property 3.2 to allow gate movements beyond the conventional moving rule limits.

In the previous chapter, an optimization approach called Gate Reduction Procedure GRP (Procedure 3.1) was proposed that uses the generalized moving rule at the Step 6 to find possible cancelations. Using the generalized moving rule requires knowing the functionality at each line segment in a circuit. The functions on the line segments after each gate are stored at Step 1 of the procedure. In Section 3.3.1, we showed that one way to store functions at different line segments in a circuit is to represent them by labels. Therefore, a procedure called the Line Labeling Procedure (LLP) (Procedure 3.2) was designed to assign labels to circuit line segments. This labeling is such that if two segments on a circuit line have the same label, those segments realize the same function. The procedure uses stacks to keep track of identity structures.

Although the same label on a circuit line identify the same function, there might be segments on a line with the same function but different labels. It means that the Line Labeling Procedure does not guarantee to find all identical functions. Hence, if we use the LLP (Procedure 3.2) as the first step of the Gate Reduction Procedure (Procedure 3.1), it is not guaranteed to find all possible reductions and there are cases where identities cannot be found using this method. In the following, two examples are shown where the identity structures cannot be found by using labels as in the Line Labeling Procedure.

**Example 4.1.** *Consider the circuits in Figure 4.1. Two consecutive identical MCT gates are shown in Figure 4.1 (a) that realize the identity according to Property 2.2. In Figure 4.1 (b), the first MCT gate is replaced by its five gate realization from the NCV library (see Figure 2.10) and since controls can have any order in MCT*

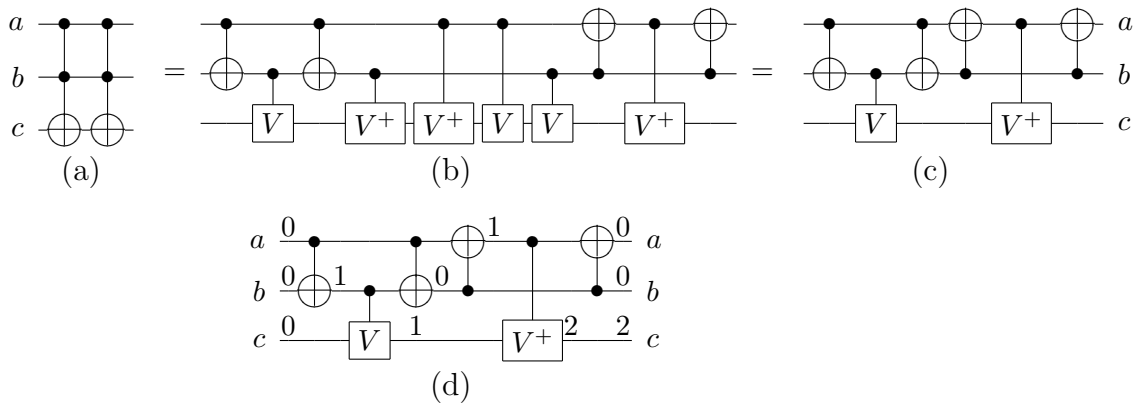



Figure 4.1: (a) Two MCT gates realizing the identity (b) After substituting the NCV realizations (c) After removing the four redundant gates in the middle (d) After applying the Line Labeling procedure.

*gates, the second MCT gate is replaced by the inverse realization (Property 2.3) of the same MCT with a different order of controls. In the resulting circuit,* $V^+(a;c)$ *from the first realization can be canceled with* $V(a;c)$ *from the second realization and similarly* $V^+(b;c)$ *from the first realization can be canceled with* $V(b;c)$ *from the second realization which yields the circuit shown in Figure 4.1 (c). Since the circuit in Figure 4.1 (c) is equivalent to the circuit in Figure 4.1 (a), it realizes the identity. Figure 4.1 (d) shows the result of applying the Line Labeling Procedure (Procedure 1 in [79]) to the circuit of Figure 4.1 (c). Note that only changes on the line labels are shown. As shown, the labels on line c are not the same at the two ends of the circuit although they represent the same function.*

**Example 4.2.** *Consider the two consecutive Swap gates shown in Figure 4.2 (a) realizing the identity function. If we substitute the first Swap gate with its NCV realization in Figure 2.8 and the second Swap gate with an alternate realization, the circuit in Figure 4.2 (b) will be achieved. The circuit in Figure 4.2 (b) implements the identity function, but as shown in Figure 4.2 (c), applying the Line Labeling Procedure*

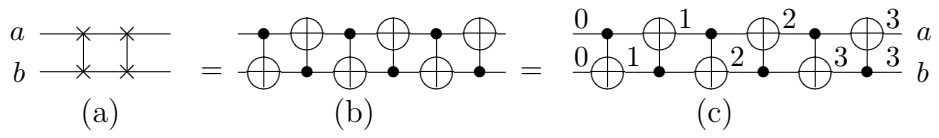



Figure 4.2: (a) Two Swap gates realizing the identity (b) After mapping to NCV gates (c) After applying the Line Labeling procedure.

*(Procedure 1 in [79]) leads to different labels at the two ends of the circuit while the functionalities of the two lines a and b are the same at both the input and the output side of the circuit.*

The above examples show that the GRP optimization method that uses the Line Labeling Procedure to find the identities cannot find all of the possible reductions in a circuit. In the next section, we discuss an alternative approach that uses functional representations instead of labels and allows more effective implementation of the generalized moving rule.

## 4.2 DDMF-based Optimization Method

It was shown in the previous section that the optimization approach of Procedure 3.1 that uses the Line Labeling Procedure as its first step cannot find all possible reductions in a circuit because labels do not uniquely represent the functionality of line segments. The solution to address this problem is to use a full representation of the functions at each line segment in the circuit. In this section, an alternative to the basic optimization method is presented that uses a decision diagram structure called Decision Diagram for a Matrix Function (DDMFs) [73] for this purpose. DDMFs are compact structures and they are well suited to represent the functionality at each line segment in a circuit.

### 4.2.1 Decision Diagram for a Matrix Function (DDMF)

Decision Diagrams for Matrix Functions are specifically designed to represent the matrix associated with each line segment in a subset of quantum circuits called *Semi-Classical Quantum Circuits (SCQC)*. Before defining the Decision Diagram for a Matrix Function, we first introduce some concepts and definitions that are required for describing DDMFs.

**Definition 4.1.** *A* **Semi-Classical Quantum Circuit (SCQC)** *is a quantum circuit in which if all the initial input quantum states of the circuit are* $|1\rangle$ *or* $|0\rangle$ *(classical values), the quantum states at all gate controls in the circuit are* $|1\rangle$ *or* $|0\rangle$ *at the time that the gate is being operated [73].*

The above definition means that entanglement does not occur in Semi-Classical Quantum Circuits as long as their inputs are initialized to classical (Boolean) values.

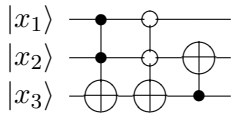



Figure 4.3: Example of an SCQC reversible circuit. .

Recalling from Chapter 2, the qubits of a quantum circuit are in an entangled quantum state if it is impossible to separate the contributions of the states of the individual qubits from the state of the whole system [23].

In general, reversible circuits that consist of reversible gates with Boolean operations are all SCQC. For instance, all circuits of Mixed-Polarity Multiple-Control Toffoli (MPMCT) gates are SCQC.

**Example 4.3.** *Consider the circuit of Figure 4.3 that consists of two MPMCT gates* $T(x_1, x_2; x_3)$ *and* $T(\overline{x_1}, \overline{x_2}; x_3)$ *followed by the CNOT gate* $T(x_3; x_2)$*. This circuit transforms the state of the third qubit* $|x_3\rangle$ *into* $|x_3 \oplus f(x_1, x_2)\rangle$ *where* $f(x_1, x_2) = x_1.x_2 + \overline{x_1}.\overline{x_2}$*. Hence, the circuit calculates a Boolean function on the third line and the value on the control of the third gate is always* $|0\rangle$ *or* $|1\rangle$ i.e. *the circuit is SCQC.*

A reversible circuit that is SCQC may be realized by a non-SCQC quantum circuit depending on the mapping procedure and the quantum cost model that is used.

**Example 4.4.** *Now consider the quantum circuit shown in Figure 4.4. This circuit is composed of the five gate realization of a 2-control MCT gate as in Figure 2.10 (a) followed by a CNOT gate* $T(x_3; x_2)$*. Hence, the third qubit* $|x_3\rangle$ *is transformed into* $|x_3 \oplus f(x_1, x_2)\rangle$ *where* $f(x_1, x_2) = x_1.x_2$*. Thus the circuit is SCQC, but it is not trivial to check the condition for the quantum state of the control qubit of the last gate* $T(x_3; x_2)$*. We later show that by using the concepts that are explained in this section whether a circuit is an SCQC is easily checked.*

**Example 4.5.** *Figure 4.5 shows an example of a non-SCQC circuit. If the second qubit* $|x_2\rangle$ *is* $|1\rangle$ *then the value of the first qubit* $|x_1\rangle$ *at the control connection of the second gate* $T(x_2; x_1)$ *will be* $V|x_1\rangle$ *in which* $V$ *corresponds to the matrix of Equation 2.11 in Chapter 2. Hence, the first qubit has a quantum value at the time when the second gate is being operated and this causes entanglement.*

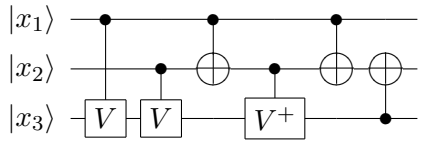



Figure 4.4: Example of an SCQC quantum circuit. .

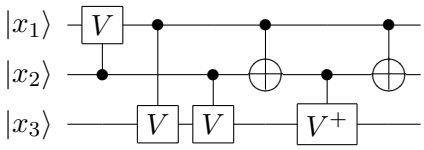



Figure 4.5: Example of a non-SCQC quantum circuit. .

Later in the experimental results section we show that non-entangled NCV (NOT, CNOT, $V$, $V^+$, controlled-$V$, and controlled-$V^+$) circuits are achieved by using non-entangled quantum realizations of MPMCT gates in mapping MPMCT circuits to quantum circuits. Removing the entanglement adds extra quantum costs that are sometimes comparable to the improvements achieved by the new method.

**Definition 4.2.** *A* **quantum function** $f : \mathbb{B}^n \rightarrow \mathbb{Q}$ *with respect to n Boolean variables* $x_1, x_2, \ldots, x_n$ *is a mapping from each of the* $2^n$ *variable assignments to a qubit state* $|Q\rangle$ *and is denoted by* $qf(x_1, x_2, \ldots, x_n)$ *[73].*

**Example 4.6.** *See the third bit after the first gate in the circuit of Figure 4.4 again. If the initial state of* $|x_3\rangle$ *is* $|0\rangle$*, the resulting state of the third qubit after the first gate can be seen as a quantum function* $qf_1(x_1, x_2)$ *as shown in the second column of Table 4.1. For example, assigning* $x_1 = 1$ *and* $x_2 = 0$ *leads to the state of* $|x_3\rangle$ *after the first gate transform to* $V|0\rangle$*. Hence,* $qf_1(1, 0) = V|0\rangle$ *as shown in Table 4.1.*

A Boolean function can be considered as a special case of quantum functions as defined above in which the output states are restricted to $|0\rangle$ and $|1\rangle$ corresponding to the Boolean values 0 and 1 respectively.

**Example 4.7.** *The third column of Table 4.1 shows the quantum function of the third qubit after the two MPMCT gates in the circuit of Figure 4.3 given that the initial state of* $|x_3\rangle$ *is* $|0\rangle$*.*

Table 4.1: Truth table for quantum functions and matrix functions of Examples 4.6 to 4.8.

| $x_1, x_2$ | $qf_1$ | $qf_2$ | $mf_1$ | $mf_2$ | $CM(NOT)$ |
|---|---|---|---|---|---|
| 0, 0 | $\|0\rangle$ | $\|1\rangle$ | $I$ | $NOT$ | $NOT$ |
| 0, 1 | $\|0\rangle$ | $\|0\rangle$ | $I$ | $I$ | $NOT$ |
| 1, 0 | $V\|0\rangle$ | $\|0\rangle$ | $V$ | $I$ | $NOT$ |
| 1, 1 | $V\|0\rangle$ | $\|1\rangle$ | $V$ | $NOT$ | $NOT$ |

**Definition 4.3.** *A* **matrix function** *with $n$ Boolean variables $x_1, \ldots, x_n$ is a mapping from $\{0,1\}^n$ to $2 \times 2$ unitary matrices and is denoted by $mf(x_1, \ldots, x_n)$ [73].*

The value of a quantum function $qf(x_1, \ldots, x_n)$ can always be expressed by a matrix function as $mf(x_1, \ldots, x_n)|0\rangle$.

**Example 4.8.** *The fourth and the fifth columns of the Table 4.1 show the matrix functions $mf_1$ and $mf_2$ associated with the quantum functions $qf_1$ and $qf_2$ respectively. A matrix function whose output values are only $\boldsymbol{I}$ or $\boldsymbol{NOT}$ is treated as a classical Boolean function by considering that the $\boldsymbol{NOT}$ and $\boldsymbol{I}$ matrices in outputs of the matrix function correspond to the 1 and 0 values in the Boolean function respectively.*

**Definition 4.4.** *A matrix function $mf(x_1, \ldots, x_n)$ is called* **a constant matrix function** *if $mf(x_1, \ldots, x_n)$ is the same matrix $\boldsymbol{M}$ for all assignments to $x_1, \ldots, x_n$ and is denoted by $CM(\boldsymbol{M})$ [73].*

**Example 4.9.** *The constant matrix function $CM(\boldsymbol{NOT})$ with respect to the variables $x_1$ and $x_2$ is shown in the last column of Table 4.1.*

The following two operators $\oplus$ and $*$ are defined on matrix functions [73]. These operators are used to construct DDMFs for quantum circuits, which will be described later in this chapter.

**Definition 4.5.** *Let $mf_1$ and $mf_2$ be matrix functions with respect to $x_1, \ldots, x_n$. The $mf_1 \oplus mf_2$ is defined as*

$$mf_1 \oplus mf_2 = mf_1(x_1, \ldots, x_n) \cdot mf_2(x_1, \ldots, x_n) \tag{4.1}$$

*where $\cdot$ means normal matrix multiplication [73].*

Table 4.2: Example of operators $\oplus$ and $*$.

| $x_1, x_2$ | $mf_1$ | $mf_2$ | $mf_1 \oplus mf_2$ | $mf_3$ | $f$ | $f * mf_3$ |
|---|---|---|---|---|---|---|
| 0, 0 | $R(\frac{\pi}{2})$ | $R(\frac{\pi}{2})$ | $R(\pi)$ | $R(\frac{\pi}{2})$ | 1 | $R(\frac{\pi}{2})$ |
| 0, 1 | $I$ | $I$ | $I$ | $I$ | 0 | $I$ |
| 1, 0 | $I$ | $R(\frac{\pi}{4})$ | $R(\frac{\pi}{4})$ | $R(\pi)$ | 1 | $R(\pi)$ |
| 1, 1 | $R(\frac{\pi}{2})$ | $R(\frac{\pi}{4})$ | $R(\frac{3\pi}{4})$ | $R(\pi)$ | 0 | $I$ |

**Definition 4.6.** *Let $mf_3$ be a matrix function with respect to $x_1, \ldots, x_n$ and $f$ be a Boolean function with respect to $x_1, \ldots, x_n$. The $f * mf_3$ is defined as below [73].*

$$f * mf_3 = \begin{cases} mf_3(x_1, \ldots, x_n) & \text{if } f(x_1, \ldots, x_n) = 1 \\ I & \text{if } f(x_1, \ldots, x_n) = 0 \end{cases} \tag{4.2}$$

**Example 4.10.** *Consider the $mf_1$ and $mf_2$ functions in the second and third columns of Table 4.2. The $\boldsymbol{R}(\theta)$ matrices are rotations about the $\hat{x}-$axis on the Bloch sphere by $\theta$ degrees (see Figure 2.1).*

$$\boldsymbol{R}(\theta) = \begin{pmatrix} cos(\theta/2) & -isin(\theta/2) \\ -isin(\theta/2) & cos(\theta/2) \end{pmatrix} \tag{4.3}$$

*The corresponding $mf_1 \oplus mf_2$ is shown in the fourth column. In addition, for the $mf_3$ and the $f$ functions shown in the fifth and the sixth columns, the corresponding $f * mf_3$ operation is shown in the last column. Note that if both $mf_1$ and $mf_2$ are Boolean functions, the operator $\oplus$ acts as the $EXOR$ of the two functions. Also, if $mf_3$ is a Boolean function, the operator $*$ corresponds to the $AND$ of the two Boolean functions.*

A matrix function can be expressed by an edge-valued decision diagram structure which is described as follows.

**Definition 4.7.** *A* **Decision Diagram for a Matrix Function (DDMF)** *is a directed acyclic graph with three types of nodes:*

1. *A single terminal node corresponding to the identity matrix $\boldsymbol{I}$,*
2. *a root node with an incoming edge having a weighted matrix $\boldsymbol{M}$, and*
3. *a set of non-terminal (internal) nodes [73].*

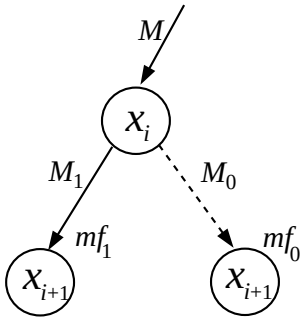



Figure 4.6: An internal DDMF node.

In DDMFs, each internal and the root node are associated with a Boolean variable $x_i$, and have two outgoing edges which are called the 1-edge (solid line) leading to its 1-child node and the 0-edge (dashed line) leading to its 0-child node. DDMFs are ordered *i.e.* variables appear in the same order on each path from the root node to a terminal node. Every edge has an associated matrix. The matrix function represented by a node is defined recursively by the following three rules [73, 96].

1. The matrix function represented by the terminal node is the constant matrix function $CM(\boldsymbol{I})$.

2. The matrix function represented by an internal node (or the root node) whose associated variable is $x_i$ is defined as

$$mf = x_i * (CM(\boldsymbol{M}_1) \oplus mf_1) \oplus \overline{x_i} * (CM(\boldsymbol{M}_0) \oplus mf_0) \tag{4.4}$$

   where $mf_1$ and $mf_0$ are the matrix functions represented by the 1-child node and the 0-child node respectively, and $\boldsymbol{M}_1$ and $\boldsymbol{M}_0$ are the matrices of the 1-edge and the 0-edge, respectively.

3. The root node has one incoming edge that has a matrix $\boldsymbol{M}$. The matrix function represented by the whole DDMF is $CM(\boldsymbol{M}) \oplus mf$, where $mf$ is the matrix function represented by the root node. (See an illustration of this structure in Figure 4.6)

Similar to conventional binary decision diagrams (BDD), a canonical form can be defined for DDMFs.

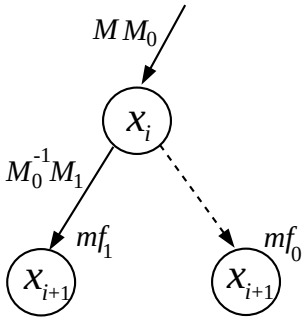



Figure 4.7: A normalized internal DDMF node.

**Definition 4.8.** *A DDMF is* **canonical** *if the following conditions hold [73]:*

1. *All the matrices on 0-edges are* $\boldsymbol{I}$.
2. *There are no redundant nodes* i.e. *no node has both its 0-edge and 1-edge pointing to the same node with* $\boldsymbol{I}$ *as the 1-edge matrix.*
3. *Common subgraphs are shared* i.e. *there are no two identical subgraphs.*

Any DDMF can be normalized to its canonical form by applying the following transformations on its edges from the terminal nodes to the root node. Suppose the matrices on the incoming edge, 0-edge and 1-edge of a node be $\boldsymbol{M}$, $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$, respectively. Then, if $\boldsymbol{M}_0$ is not $\boldsymbol{I}$, we modify these three matrixes as follows:

1. The matrix on the incoming edge is changed to be $\boldsymbol{M}\boldsymbol{M}_0$.
2. The matrix on the 1-edge is changed to be $\boldsymbol{M}_0^{-1}\boldsymbol{M}_1$.
3. The matrix on the 0-edge is changed to be $\boldsymbol{I}$.

It is easily verified that the above transformations do not change the matrix function represented by the DDMF. See the illustration in Figure 4.7 where the matrix on the 0-edge of the node $x_i$ is converted to $\boldsymbol{I}$. The $\boldsymbol{I}$ matrices on 0-edges are not shown for simplicity.

**Definition 4.9.** *Let* $DDMF_{mf_1}$, $DDMF_{mf_2}$ *and* $DDMF_{mf_3}$ *be DDMFs that represent matrix functions* $mf_1$, $mf_2$ *and* $mf_3$ *respectively. Then* $DDMF_{mf_1} \oplus DDMF_{mf_2}$ *is defined as a DDMF that represents the matrix function* $mf_l \oplus mf_2$. *Let also* $DDMF_f$ *be a DDMF that represents a Boolean function* $f$. *Then* $DDMF_f * DDMF_{mf_3}$ *is defined as a DDMF that represents the matrix function* $f * mf_3$.

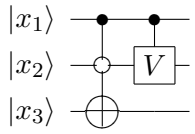



Figure 4.8: Example of an SCQC circuit. .

### 4.2.2 DDMFs for Reversible and Quantum Circuits

We use DDMFs to represent matrix functions on all line segments in a circuit. To begin, for each primary input $x_i$ a DDMF representing that variable is built. Then, gates are considered one at a time from the primary inputs towards the outputs, and DDMFs for the functions corresponding to the circuit lines on the output side of the gate are constructed from DDMFs on the input side as follows.

Let $D_i^j$ be the DDMF for the $i-$th line after the $j-$th gate and $F(D)$ be the matrix function represented by a DDMF $D$. Then the DDMFs after the $j-$th gate are built as follows:

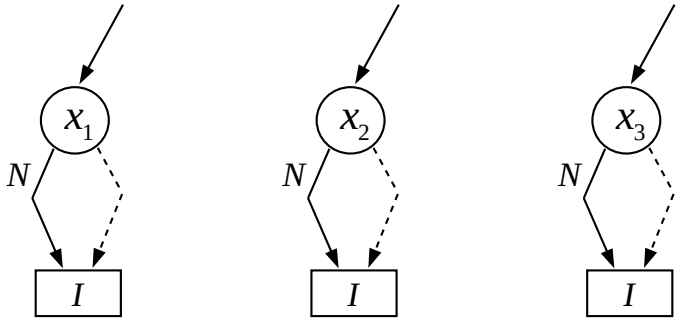
1. If the $i-$th line is not the target bit of the $j-$th gate, assign $D_i^j = D_i^{j-1}$.

2. If the $i-$th line is the target of the $j-$th gate, assign $D_i^j = D_i^{j-1} \oplus D_{gate}$ where $D_{gate}$ is constructed by the following steps:

   (a) Assume the $j-$th gate has positive controls $p_1, p_2, \ldots, p_k$ and negative controls $n_1, n_2, \ldots, n_l$. The matrix function for the controls ($g$) is constructed by $g = F(D_{p_1}^{j-1}) \cdot F(D_{p_2}^{j-1}) \ldots F(D_{p_k}^{j-1}) \cdot \overline{F(D_{n_1}^{j-1})} \cdot \overline{F(D_{n_2}^{j-1})} \ldots \overline{F(D_{n_l}^{j-1})}$. Because of the restriction of SCQCs, all the matrix functions for the controls are essentially classical Boolean functions and the above expression is simply the logical AND of those functions.

   (b) The $D_{gate}$ is constructed by $D_{gate} = (DDMF \text{ for } g) * (DDMF \text{ for } CM(U))$, where $U$ is a unitary matrix associated with the $j-$th gate.

**Example 4.11.** *Consider the circuit in Figure 4.8. First, a DDMF corresponding to an identity function is constructed for each input as shown in Figure 4.9. The matrix function for the controls of the first gate (g) is obtained from* $g = F(D_{x_1}^0).\overline{F(D_{x_2}^0)}$ *in which the* $F(D_{x_1}^0)$ *and* $\overline{F(D_{x_2}^0)}$ *correspond to the matrix functions of the inputs* $x_1$ *and*




(a) (b) (c)

Figure 4.9: DDMF structures for the inputs in the circuit of Figure 4.8 (a) $D^0_{x_1}$ (b) $D^0_{x_2}$ and (c) $D^0_{x_3}$.

Table 4.3: Matrix functions for $D^0_{x_1}$ and $D^0_{x_2}$.

| $x_1, x_2$ | $F(D^0_{x_1})$ | $F(D^0_{x_2})$ | $g = F(D^0_{x_1}).\overline{F(D^0_{x_2})}$ |
|---|---|---|---|
| 0, 0 | $I$ | $I$ | $I$ |
| 0, 1 | $I$ | $NOT$ | $I$ |
| 1, 0 | $NOT$ | $I$ | $NOT$ |
| 1, 1 | $NOT$ | $NOT$ | $I$ |

*$x_2$ respectively. These functions are shown in Table 4.3. Note that since the values on the controls are always classical Boolean values, the "." operation is in fact the logical $AND$. The DDMF of the first gate is built from the DDMF of the g and the DDMF of the CM(**NOT**) as the gate inverts the target* i.e. $D_{gate} = (DDMF\ for\ g) * (DDMF\ for\ CM(\mathbf{NOT}))$. *The DDMFs for the g and $CM(\mathbf{NOT})$ are shown in Figure 4.10 (a) and (b) respectively. The DDMF of the third qubit $|x_3\rangle$ after the first gate is shown in Figure 4.10 (c). The DDMFs of the first two qubits $|x_1\rangle$ and $|x_2\rangle$ after the first gate are the same as the DDMFs shown in Figure 4.9 (a) and (b). Similarly, the DDMFs after the second gate $V(x_1; x_2)$ can be built from the DDMFs after the first gate. Figure 4.11 shows the DDMFs on the output side of the circuit in Figure 4.8.*

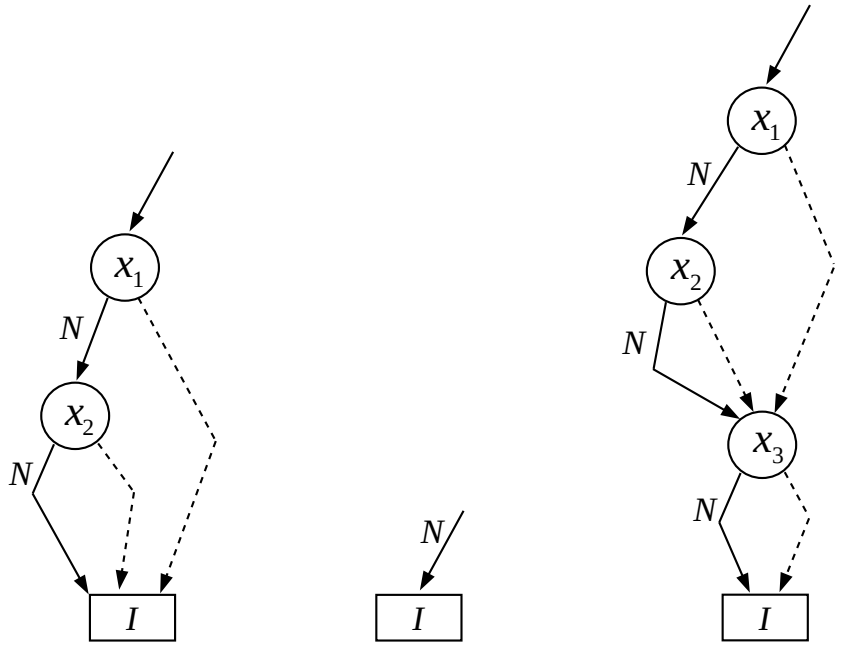



Figure 4.10: DDMF structures for (a) $g$ (b) $CM(\boldsymbol{NOT})$ and (c) $D^1_{x_3}$ for the circuit of Figure 4.8.

### 4.2.3 The Functional Optimization Method

In the previous chapter, an optimization method titled the Gate Reduction Procedure (Procedure 3.1) was introduced that requires knowing the functional changes after each gate being processed in a circuit. We showed that the basic method that uses labels to represent functions does not find all functional equivalences. The best solution for this problem is to use a functional description such as DDMFs. We chose DDMFs because they are well suited to represent functions on the circuit line segments.

The DDMF-based optimization first builds DDMFs for each input of a given circuit. Then, it follows the steps of the Gate Reduction Procedure (Procedure 3.1) and each time Step 1 of the procedure is performed, the DDMFs after the selected gate are constructed from the DDMFs for the circuit lines before the gate. Comparison of the functionalities on the line segments (as in Steps 2,3 and 6) are performed by comparing the DDMF structures on the corresponding lines. Because DDMFs are in the canonical form, the comparison is easily performed in constant time by comparing the pointers to the root nodes. Hence, DDMFs are stored for all the line segments in

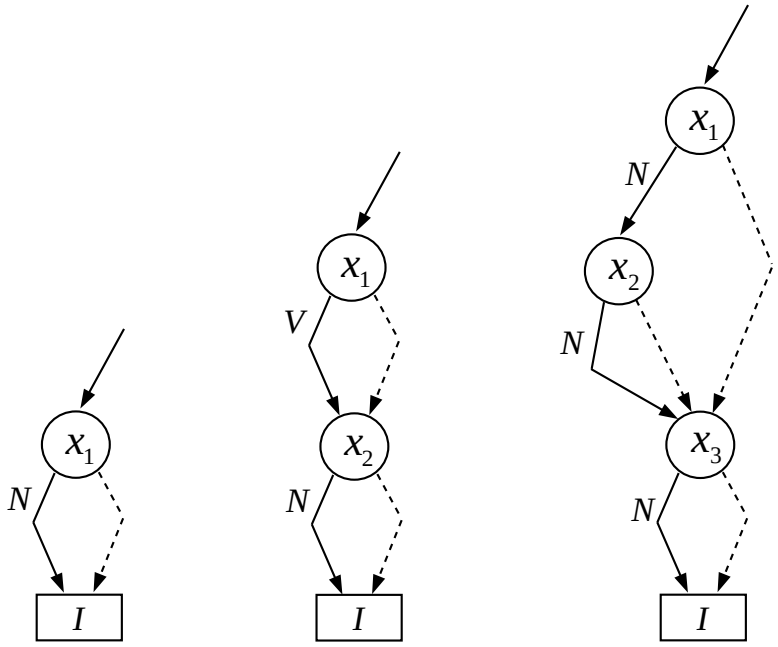



Figure 4.11: DDMF structures for the outputs in the circuit of Figure 4.8 (a) $D^2_{x_1}$ (b) $D^2_{x_2}$ and (c) $D^2_{x_3}$.

a circuit.

Using DDMFs, the identity blocks in the examples of Figures 4.1 and 4.2 can be found and removed from circuits as the second and third steps of the optimization procedure are being performed.

## 4.3 Experimental Results

The basic optimization procedure has been implemented using Python 2.6.5 and the results of applying it on different benchmark circuits were discussed in the previous chapter. Prototype implementation of the DDMF-based optimization procedure has been implemented using the C++ programming language. We use the DDMF package [73] in which the DDMF operations such as comparing two DDMFs or operations in Definitions 4.5 and 4.6 are implemented efficiently. The experiments were run on a system with a 3.2 GHz i5-650 CPU and 3.0 GB RAM. The programs use REVLIB [9] *real* circuit format. All circuits generated in our experiments have been verified using

the QMDD-based verification approach described in [95].

To compare the basic optimization method with the DDMF-based optimization method, we applied the DDMF-based method to the 156 benchmark circuits of the first test suite in the experimental results of the previous chapter. The results are summarized in Table 4.4 for the circuits in which the improvement was over 10%.

For each circuit, the second column shows the quantum cost reported in the REVLIB website [9] followed by the quantum cost of the initial circuit that we consider. The initial quantum costs are slightly different from the REVLIB costs as they are targeted for the NCV quantum library whereas the REVLIB costs are based on a different quantum library. Note that the reversible gates other than MPMCT gates such as Fredkin [64] or Peres [63] gates are all accommodated by replacing them by equivalent MPMCT gate sequences. The fourth column shows the quantum costs of the resulting circuits after applying the DDMF-based optimization on the MPMCT reversible circuits. The next column shows the costs after mapping to quantum NCV circuits and the sixth column corresponds to the quantum costs after applying the DDMF-based optimization on the quantum circuits.

The only benchmark circuits that could be optimized using the DDMF-based method were small circuits with quantum costs up to 112 as reported in Table 4.4 because the DDMF-based method only applies to non-entangled circuits and the larger benchmark circuits contained more complex reversible gates that had entangled realizations in the NCV library used for the mapping. The cumulative cost reduction for the circuits reported in Table 4.4 is 15.35%. The cumulative cost reduction for all the non-entangled circuits in the test suite is 7.36% which is quite a bit lower because the remaining non-entangled circuits are small and are already highly optimized. For the same reason, the difference between the two methods (basic and DDMF-based methods) is not visible in this table. The average cost reduction for the circuits in Table 4.4 is 16.58% and the average improvement among all the circuits in the test suite is 5.56%. Note that the execution times are rather higher than the basic method despite the fact that the basic method is written in Python which is supposed to run slower than C++ code.

In the results of Table 4.4, we used the REVLIB cost model for mapping MPMCT circuits to NCV circuits that generates entanglement for large circuits. However, to show the benefits of the DDMF-based optimization, we need large SCQC quantum circuits which requires a technology mapper that considers this restriction in mapping reversible circuits to quantum circuits. In the next chapter, we will discuss different

Table 4.4: Optimization results of using the DDMF-based method on REVLIB benchmark circuits.

| | **REVLIB Cost** | **Initial Cost** | **MCT Opt.** | **Map to NCV** | **NCV Opt.** | **%Imp. wrt initial Cost** | **Total Time (s)** |
|---|---|---|---|---|---|---|---|
| 4mod5-v0_18 | 25 | 25 | 19 | 19 | 17 | 32.00 | 0.032 |
| 4mod5-v0_21 | 19 | 25 | 19 | 19 | 17 | 32.00 | 0.032 |
| toffoli_double_4 | 10 | 10 | 7 | 7 | 7 | 30.00 | 0.000 |
| mod5d2_70 | 16 | 16 | 13 | 13 | 13 | 18.75 | 0.016 |
| 4gt13_90 | 34 | 35 | 32 | 32 | 29 | 17.14 | 0.031 |
| rd32-v0_66 | 12 | 12 | 12 | 12 | 10 | 16.67 | 0.032 |
| rd32-v0_67 | 8 | 12 | 12 | 12 | 10 | 16.67 | 0.032 |
| rd73_140 | 76 | 76 | 76 | 76 | 64 | 15.79 | 0.062 |
| rd73_141 | 64 | 76 | 76 | 76 | 64 | 15.79 | 0.047 |
| sym9_146 | 108 | 108 | 108 | 108 | 91 | 15.74 | 0.078 |
| sym9_147 | 94 | 108 | 108 | 108 | 91 | 15.74 | 0.078 |
| 4mod5-v0_19 | 13 | 13 | 11 | 11 | 11 | 15.38 | 0.032 |
| mod5mils_65 | 13 | 13 | 11 | 11 | 11 | 15.38 | 0.016 |
| mod5mils_71 | 13 | 13 | 11 | 11 | 11 | 15.38 | 0.032 |
| rd32-v1_68 | 13 | 13 | 13 | 13 | 11 | 15.38 | 0 |
| rd32-v1_69 | 9 | 13 | 13 | 13 | 11 | 15.38 | 0.032 |
| alu-v2_31 | 101 | 107 | 91 | 91 | 91 | 14.95 | 0.032 |
| hwb4_49* | 65 | 67 | 60 | 60 | 57 | 14.93 | 0.032 |
| hwb4_50* | 63 | 67 | 60 | 60 | 57 | 14.93 | 0.016 |
| one-two-three-v1_99 | 36 | 37 | 34 | 34 | 32 | 13.51 | 0.032 |
| rd84_142 | 112 | 112 | 112 | 112 | 97 | 13.39 | 0.074 |
| rd84_143 | 98 | 112 | 112 | 112 | 97 | 13.39 | 0.078 |
| 4gt11_82 | 16 | 16 | 16 | 16 | 14 | 12.50 | 0.032 |
| sys6-v0_111 | 72 | 72 | 72 | 72 | 63 | 12.50 | 0.047 |
| sys6-v0_144 | 62 | 72 | 72 | 72 | 63 | 12.50 | 0.062 |
| miller_11 | 17 | 17 | 17 | 17 | 15 | 11.76 | 0.000 |
| 4gt5_76 | 29 | 30 | 27 | 27 | 27 | 10.00 | 0.032 |
| **Total** | **1198** | **1277** | **1214** | **1214** | **1081** | ** | **0.989** |

* An additional line is added for mapping to the NCV library.

** Cumulative Improvement: 15.35%, Average Improvement: 16.58%

quantum cost functions and a state-of-the-art MPMCT to NCV mapper [99]. We will here use this mapper and a modified version of it to generate SCQC circuits without going into the details of its implementation. For a detailed description of this cost model see the next chapter.

Our second test suite consists of 22 MPMCT circuits that are produced by the QMDD-based synthesis method in [44, 100] and optimized by the ESOP-based optimization EXORCISM-4 [88] as reported in [82] plus the three unstructured reversible functions (urf1, urf2 and urf5) taken from the REVLIB website [9]. To compare the basic and the DDMF-based optimization procedures, both optimization methods were applied to the MPMCT circuits in the test suites. The results are mapped to NCV circuits and the optimization procedures were again applied to the NCV circuits. For mapping MPMCT to NCV circuits, we used the mapping procedure introduced

in [99] with an extension to generate non-entangled NCV circuits to comply with the limitation of using DDMFs (*i.e.* being applicable on the SCQCs only). To this end, the target of an MPMCT gate is never used as an ancillary as it is mapped to an NCV circuit. The resulting realizations are more expensive than the entangled realizations reported in [99]. Tables 4.5 and 4.6 show the non-entangled NCV costs of MPMCT gates with 1 and $n-3$ ancillaries respectively.

We here consider three different approaches: (1) the basic optimization method on entangled circuits, (2) the basic optimization method on non-entangled circuits and (3) the DDMF-based optimization method on non-entangled circuits. The results of applying these methodologies are summarized in Table 4.7. The first column of the table gives the REVLIB circuit name and file identification number. The column labeled *Initial* shows the non-Entangled NCV cost of the initial MPMCT circuits. The next 12 columns give the result of applying the basic optimization on entangled circuits, basic optimization on non-entangled circuits and DDMF-based optimization on non-entangled circuits. For each method, the results are reported in four columns:

(a) The NCV cost of the circuits resulting from applying the corresponding optimization method on the initial MPMCT circuits.

(b) The NCV cost of resulting circuits after mapping MPMCT circuits to NCV circuits using [99].

(c) The NCV costs after applying the corresponding optimization method on the circuits of column (b).

(d) The total elapsed time in seconds for steps (a)-(c).

The last two columns of Table 4.7 show the difference between the three approaches in terms of the quantum cost. $\Delta_1$ is the difference between the ninth column (Basic Non-Entangled (c)) and the thirteenth column (DDMF (c)). In fact, $\Delta_1$ gives the improvement that can be achieved by using the DDMF-based optimization method rather than the basic optimization method for non-entangled circuits. $\Delta_2$ gives the difference between the fifth column (Basic Entangled (c)) and the thirteenth column (DDMF (c)). It compares the two options of using the basic method on entangled circuits and the DDMF-based method on non-entangled circuits.

The results are interesting in that they show that using a functional approach does not in general add significant improvement to what can be achieved by the basic

Table 4.5: NCV cost of MPMCT gates of size $n = 4 \ldots 16$ with 1 ancillary.

| Controls | Number of Negative Controls | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 3 | 14 | 14 | 16 | 18 | | | | | | | | | | | | |
| 4 | 20 | 20 | 20 | 22 | 24 | | | | | | | | | | | |
| 5 | 42 | 42 | 42 | 44 | 46 | 48 | | | | | | | | | | |
| 6 | 54 | 54 | 54 | 54 | 56 | 58 | 60 | | | | | | | | | |
| 7 | 72 | 72 | 72 | 72 | 76 | 78 | 80 | 82 | | | | | | | | |
| 8 | 84 | 84 | 84 | 84 | 84 | 86 | 88 | 90 | 92 | | | | | | | |
| 9 | 108 | 108 | 108 | 108 | 108 | 110 | 112 | 114 | 116 | 118 | | | | | | |
| 10 | 132 | 132 | 132 | 132 | 132 | 132 | 134 | 136 | 138 | 140 | 142 | | | | | |
| 11 | 156 | 156 | 156 | 156 | 156 | 156 | 158 | 160 | 162 | 164 | 166 | 168 | | | | |
| 12 | 180 | 180 | 180 | 180 | 180 | 180 | 180 | 182 | 184 | 186 | 188 | 190 | 192 | | | |
| 13 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 206 | 208 | 210 | 214 | 216 | 218 | 220 | | |
| 14 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 230 | 232 | 234 | 238 | 240 | 242 | 244 | |
| 15 | 252 | 252 | 252 | 252 | 252 | 252 | 252 | 252 | 254 | 256 | 258 | 260 | 264 | 268 | 270 | 272 |

Table 4.6: NCV cost of MPMCT gates of size $n = 4 \ldots 16$ with $n - 3$ ancillary lines.

| Controls | Number of Negative Controls | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 3 | 14 | 14 | 16 | 18 | | | | | | | | | | | | |
| 4 | 20 | 20 | 20 | 22 | 24 | | | | | | | | | | | |
| 5 | 32 | 32 | 32 | 34 | 36 | 38 | | | | | | | | | | |
| 6 | 44 | 44 | 44 | 44 | 46 | 48 | 50 | | | | | | | | | |
| 7 | 56 | 56 | 56 | 56 | 58 | 60 | 62 | 64 | | | | | | | | |
| 8 | 68 | 68 | 68 | 68 | 68 | 70 | 72 | 74 | 76 | | | | | | | |
| 9 | 80 | 80 | 80 | 80 | 80 | 82 | 84 | 86 | 88 | 90 | | | | | | |
| 10 | 92 | 92 | 92 | 92 | 92 | 92 | 94 | 96 | 98 | 100 | 102 | | | | | |
| 11 | 104 | 104 | 104 | 104 | 104 | 104 | 106 | 108 | 110 | 112 | 114 | 116 | | | | |
| 12 | 116 | 116 | 116 | 116 | 116 | 116 | 116 | 118 | 120 | 122 | 124 | 126 | 128 | | | |
| 13 | 128 | 128 | 128 | 128 | 128 | 128 | 128 | 130 | 132 | 134 | 136 | 138 | 140 | 142 | | |
| 14 | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 142 | 144 | 146 | 148 | 150 | 152 | 154 | |
| 15 | 152 | 152 | 152 | 152 | 152 | 152 | 152 | 152 | 154 | 156 | 158 | 160 | 162 | 164 | 166 | 168 |

method. However, in some cases (*e.g.* the clip_206 circuit) the DDMF-based method outperforms the basic entangled method despite using the expensive non-entangled cost function. The CPU time of the DDMF-based method is significantly more than the basic method as expected.

Table 4.7: The results of applying the basic and DDMF-based optimization methods on MPMCT circuits.

| | | **Basic Entangled** | | | | **Basic Non-Entangled** | | | | **DDMF** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Circuits | Initial | (a) | (b) | (c) | (d) | (a) | (b) | (c) | (d) | (a) | (b) | (c) | (d) | $\Delta_1$ | $\Delta_2$ |
| adr4_197 | 535 | 161 | 161 | 161 | 1.351 | 161 | 161 | 161 | 1.056 | 161 | 161 | 158 | 0.923 | 3 | 3 |
| clip_206 | 4061 | 2640 | 2485 | 2465 | 12.368 | 2640 | 2485 | 2465 | 11.983 | 2640 | 2485 | 2449 | 16.922 | 16 | 16 |
| cm152a_212 | 165 | 165 | 143 | 139 | 0.330 | 165 | 143 | 139 | 0.283 | 165 | 143 | 139 | 0.454 | 0 | 0 |
| cm42a_207 | 226 | 226 | 200 | 194 | 0.565 | 226 | 200 | 194 | 0.534 | 226 | 200 | 194 | 0.439 | 0 | 0 |
| cm85a_209 | 1976 | 502 | 486 | 482 | 1.441 | 502 | 486 | 482 | 1.620 | 502 | 486 | 475 | 1.517 | 7 | 7 |
| cycle10_2_110 | 1398 | 852 | 850 | 818 | 0.711 | 916 | 914 | 880 | 0.744 | 916 | 914 | 877 | 1.045 | 3 | -59 |
| dc1_220 | 239 | 239 | 206 | 200 | 0.706 | 239 | 206 | 200 | 0.629 | 239 | 206 | 200 | 0.798 | 0 | 0 |
| dc2_222 | 1412 | 1274 | 1130 | 1116 | 5.274 | 1274 | 1130 | 1116 | 5.369 | 1274 | 1130 | 1111 | 5.328 | 5 | 5 |
| max46_240 | 2874 | 2182 | 1994 | 1987 | 5.236 | 2568 | 2198 | 2192 | 6.397 | 2568 | 2198 | 2182 | 7.437 | 10 | -195 |
| misex1_241 | 638 | 625 | 539 | 531 | 2.365 | 625 | 539 | 531 | 2.224 | 625 | 539 | 526 | 2.109 | 5 | 5 |
| plus63mod4096 | 749 | 665 | 651 | 638 | 1.252 | 749 | 705 | 701 | 1.299 | 749 | 705 | 699 | 1.298 | 2 | -61 |
| plus63mod8192 | 955 | 849 | 809 | 794 | 1.252 | 955 | 884 | 866 | 1.502 | 955 | 884 | 864 | 1.642 | 2 | -70 |
| radd_250 | 536 | 226 | 208 | 208 | 0.965 | 226 | 208 | 208 | 0.965 | 226 | 208 | 204 | 1.047 | 4 | 4 |
| rd73_252 | 823 | 788 | 692 | 674 | 6.962 | 788 | 692 | 674 | 6.671 | 788 | 692 | 661 | 6.859 | 13 | 13 |
| rd84_253 | 1693 | 1577 | 1440 | 1417 | 9.364 | 1577 | 1440 | 1417 | 9.294 | 1577 | 1440 | 1405 | 10.436 | 12 | 12 |
| root_255 | 2248 | 1866 | 1719 | 1689 | 5.814 | 1866 | 1719 | 1689 | 5.533 | 1866 | 1719 | 1677 | 6.702 | 12 | 12 |
| sqn_258 | 1128 | 940 | 876 | 862 | 2.107 | 940 | 876 | 862 | 2.262 | 940 | 876 | 858 | 2.626 | 4 | 4 |
| sqrt8_260 | 500 | 456 | 404 | 391 | 1.348 | 456 | 404 | 391 | 1.270 | 456 | 404 | 389 | 1.391 | 2 | 2 |
| squar5_261 | 301 | 292 | 263 | 263 | 1.115 | 292 | 263 | 263 | 1.269 | 292 | 263 | 263 | 2.058 | 0 | 0 |
| sym9_193 | 3859 | 3159 | 2928 | 2904 | 18.998 | 3645 | 3191 | 3156 | 16.177 | 3645 | 3191 | 3141 | 22.765 | 15 | -237 |
| wim_266 | 186 | 171 | 159 | 157 | 0.426 | 171 | 159 | 157 | 0.300 | 171 | 159 | 157 | 0.281 | 0 | 0 |
| z4_268 | 521 | 412 | 406 | 400 | 2.433 | 412 | 406 | 400 | 2.227 | 412 | 404 | 390 | 2.359 | 10 | 10 |
| urf1_278 | 16130 | 15687 | 15585 | 15525 | 195.042 | 15791 | 15689 | 15629 | 196.025 | 15787 | 15685 | 15469 | 33703.485 | 160 | 56 |
| urf2_277 | 6711 | 6525 | 6475 | 6445 | 48.147 | 6555 | 6505 | 6475 | 46.992 | 6555 | 6505 | 6473 | 1031.906 | 2 | -28 |
| urf5_280 | 13613 | 13295 | 13213 | 13165 | 112.336 | 13439 | 13357 | 13309 | 111.624 | 13435 | 13353 | 13212 | 13020.905 | 97 | -47 |

## 4.4 Summary

We introduced an optimization approach that uses a functional description to implement the generalized moving rule as opposed to the labels in [79]. The results presented show that using a functional approach does not result in considerable cost savings while it is computationally expensive. However, the trade-offs between the techniques presented can be considered depending on the target circuits. For example, for small SCQCs, the DDMF-based approach is definitely the best choice while for larger circuits, the basic method seems to be more practical.

Our experiments to date have been for MPMCT circuits mapped to NCV circuits using a particular approach to the mapping of each MPMCT gate. It may be that the DDMF method will yield more improvement for alternative mapping approaches or for SCQC found by other approaches. This is a consideration for ongoing research.

# Chapter 5

# Quantum Gate Libraries

Implementation of a quantum computer in any technology must satisfy a number of essential criteria such as a scalable physical system with well characterized qubits, the ability to initialize the state of the qubits, relatively long coherence times (much longer than the gate operation time), a qubit-specific measurement capability and a universal set of quantum gates [101].

Unlike classical computers in which transistors are basic elements to implement gates of different kinds, quantum gates are implemented in different ways depending on the underlying technology which involves technology dependant complications. The technology dependant limitations may limit computer scientists from defining arbitrary quantum gates. However, there are reversible and quantum gates that are widely used in reversible and quantum circuit design and are proven to be implementable *e.g.* Cirac-Zoller implementation of the CNOT gate in Ion trap technology [102], optical implementation of a 2-control MCT gate [103], and optical implementation of a Hadamard gate [104].

Complex reversible and quantum gates (gates with a large number of qubits) are not usually directly implemented and first have to be mapped to cascades of elementary quantum gates. Elementary quantum gates involve one or two qubits and are assumed to have unit quantum cost and quantum gate libraries are defined based on these gates. In this dissertation, a universal set of elementary one or two-qubit quantum gates by which any reversible computation can be implemented is referred to as a quantum gate library. Gate level synthesis and optimizations consider the quantum cost of reversible gates as the number of elementary quantum gates used to realize them.

In this chapter, we first review an existing quantum gate library and the cor-

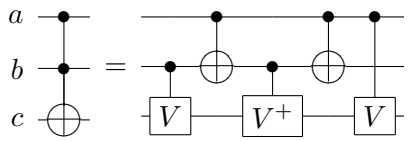



Figure 5.1: NCV realizations of $T(a, b; c)$ [5].

responding reversible to quantum mapping structures. Then, we show how we can improve it to produce lower quantum costs for reversible gates (MPMCT gates in particular). Furthermore, we explore new quantum gate libraries that can improve the quantum costs of reversible gates. For each library, we also discuss possible reversible to quantum circuit mapping structures. The effects of using the new mapping methods on the quantum cost of reversible circuits will be studied in the next chapter. This part of the dissertation was published in [82, 99, 105–110].

## 5.1 NCV Library

The most common quantum gate library, which is widely used for mapping and cost calculation of reversible gates particularly MPMCT gates, is the NCV library that is composed of NOT, CNOT, controlled-$V$ and controlled-$V^+$ gates with positive controls.

### 5.1.1 Previous Work

Barenco *et al.* [5] provided the first comprehensive study of the realization of MCT gates in terms of elementary quantum gates. The five-gate realization of a 2-control MCT gate (Toffoli gate) using elementary quantum gates was first introduced in Lemma 6.1 of [5] as shown in Figure 5.1. The authors of [5] further used the circuit of Figure 5.1 to explore the realization of larger MCT gates. Below, we will summarize the main ideas and structures.

Consider an MCT gate $G$ with the size $m + 1$ that has $m$ positive controls in an $n$-line network. According to Lemma 7.2 in [5], if $n \geq 5$ and $m \in \{3 \ldots \lceil \frac{n}{2} \rceil\}$, then $G$ can be mapped to a cascade consisting of $4(m - 2)$ 2-control MCT gates that is of the form shown in Figure 5.2 for $m = 5$ and $n = 9$. In this structure, $m - 2$ lines in addition to the gate lines are used as ancillaries (lines 6 to 8 in example of Figure 5.2). Note that in this construction and in the ones following, the gate operation is performed correctly regardless of the initial state of the ancillary lines (i.e. they do

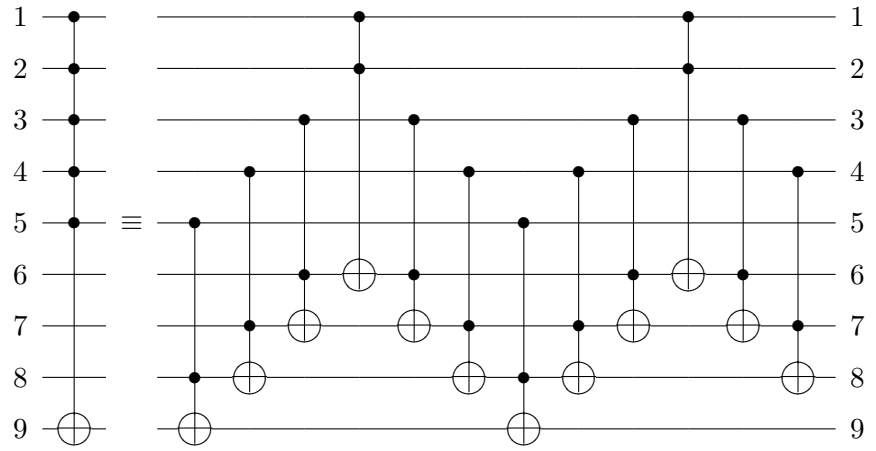



Figure 5.2: Illustration of Lemma 7.2 of [5] for $n = 9$ and $m = 5$.

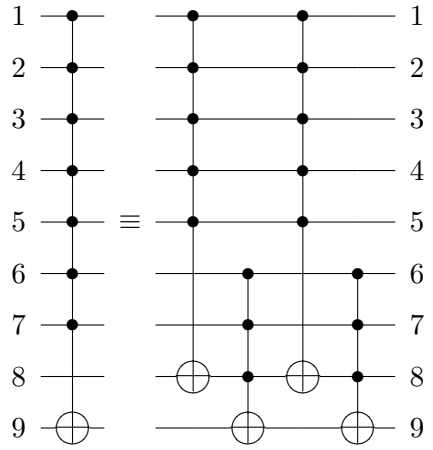



Figure 5.3: Illustration of Lemma 7.3 of [5] for $n = 9$ and $m = 3$.

not have to be "cleared" to 0 first), and after the gate operation their initial values are restored.

The 2-control MCT gates in the realization of Figure 5.2 are then mapped to the standard five-gate realization, Figure 5.1, until an NCV circuit is achieved. Note that the structure of Lemma 7.2 requires at least $m - 2$ ancillary lines be available in the circuit. For the case where the available ancillaries are less than $m-2$, Barenco *et al.* suggest another decomposition structure in Lemma 7.3 of [5]. According to Lemma 7.3, a gate $G$ with size $n - 1$ in an $n$-line circuit where $n \geq 5$ can be simulated by a network consisting of two gates with the size $m+1$ and two gates with the size $n-m$ where $m \in \{2, \ldots n - 3\}$ as shown in Figure 5.3.

Each MCT gate in the righthand side of Figure 5.3 is subsequently mapped using the structures in Figures 5.2 or 5.3 depending on the available ancillaries. Hence, an

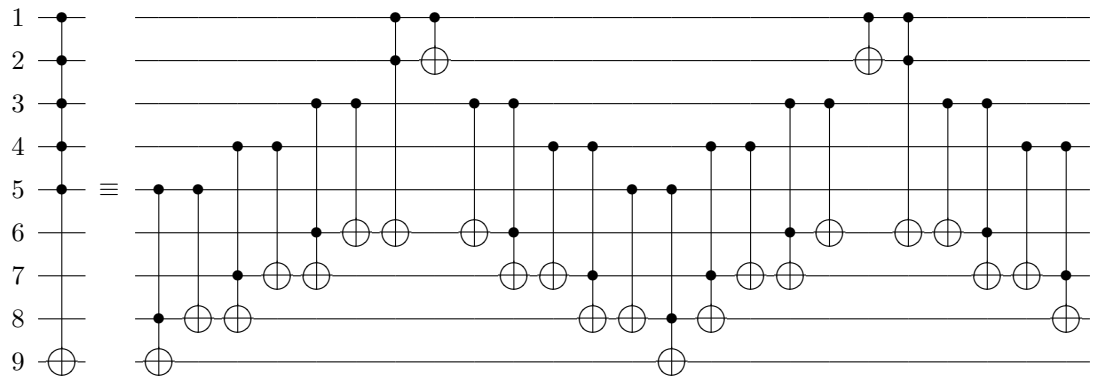



Figure 5.4: Illustration of Dueck and Maslov [6] modification of Figure 5.2.

MCT gate of size $n-1$ in an $n$-line circuit can be mapped to an NCV circuit. It is simply found that the choice of $m$ in Lemma 7.3 significantly affects the size of the resulting circuit [5]. However, as described in Corollary 7.4, there is an efficient value of $m = \lceil \frac{n}{2} \rceil$ which yields the smallest possible NCV realization. Choosing $m = \lceil \frac{n}{2} \rceil$ ensures that sufficient ancillaries are always available to map subsequent MCT gates to NCV cascades using the structure in Lemma 7.2. Proofs can be found in [5].

Note that the structure of Figure 5.3 requires at least one ancillary line *i.e.* this structure cannot be used for mapping gates of size $n$ in $n$-line circuits unless extra ancillaries are added to the circuit. In the next section we will show how it is possible to realize MCT gates without ancillaries using other quantum gate libraries.

As described above, Barenco's decomposition yields NCV realizations of size $20n-60$ for an MCT gate with size $n$ when $n-3$ ancillary lines are available. If less than $n-3$ ancillaries are available, the number of NCV gates to realize an MCT gate of size $n$ will be $40n-200$.

Dueck and Maslov [6] later modified the Barenco's fundamental decomposition rule, Figure 5.2, as shown in Figure 5.4. Dueck and Maslov's insight was to add the CNOT gates as shown in Figure 5.4 to convert the 12 2-control MCT gates to Peres and inverse Peres (TR) gates. The CNOT gates are inserted in such a way that their effects cancel in pairs so the overall circuit functionality is not changed. In chapter 2, we showed that Peres and inverse Peres gates can be realized by cascades of four NCV gates which are less expensive than five-gate realizations of 2-control MCT gates. Hence, the quantum cost of the circuit in Figure 5.4 is $12 \times 4 = 48$, 80% of the original Barenco's cost.

Maslov *et al.* [84] subsequently presented techniques whereby the number of ele-

mentary gates can be further reduced, e.g. to 38 elementary gates for the example shown above. That approach uses heuristic quantum template application to perform optimizations on the circuit and can be computationally expensive. However, using the optimizations, Maslov *et al.* reduced the cost of MCT gates of size $n$ ($n > 3$) with $n-3$ ancillaries to $12n-34$. In the same manner, the cost of an MCT gate of size $n$ with 1 ancillary line was reduced to $24n-88$ (for $n > 5$) [84].

In [71], Miller suggested a new decomposition method for mapping MCT gates to quantum gate cascades that improved the quantum cost of individual MCT gates. A key contribution in [71] was the coverage of the intermediate cases where the number of available ancillaries are between 1 and $n-3$ lines. The mapping method considers the general case of decomposing a $c$-control MCT gate, $c \geq 2$, using $p$, $1 \leq p \leq c-2$, ancillary lines $\{a_0, a_1, \ldots, a_{p-1}\}$. The central idea is to partition the control lines into $p+1$ disjoint sets $C_i$, $0 \leq i \leq p$, which together include all control lines. This is a direct extension of the Barenco *et al.* decomposition for one ancillary line and, as will be shown, also includes their decomposition for $c-2$ ancillary lines.

The partition required for $c$ controls and $p$ ancillary lines can be defined by an ordered set of $p+1$ integers $\{\alpha_0, \alpha_1, \ldots, \alpha_p\}$ where the interpretation is that each set $C_i, 0 \leq i \leq p$, contains $\alpha_i$ controls and the controls are assigned to the sets in order.

Miller's general decomposition rule is given by

$$G_0 G_1 .. G_{p-1} G_p G_{p-1}^R .. G_1^R G_0^R G_1 .. G_{p-1} G_p^R G_{p-1}^R .. G_1^R \tag{5.1}$$

where $G_0$ denotes the MCT gate $T(C_0 \cup a_0; t)$, $G_i, 1 \leq i \leq p-1$, denotes the MCT gate $T(C_i \cup a_i; a_{i-1})$, and $G_p$ denotes the MCT gate $T(C_p; a_{p-1})$. $G_i^R$ denotes the inverse implementation of $G_i$ (Property 2.3).

Once an MCT gate is decomposed using Equation 5.1, each of the MCT gates in the resulting cascade with more than two controls is recursively decomposed by applying Equation 5.1. The maximum number of ancillary lines will be available for each, so the partition $\{1, \ldots, 1, 2\}$ always applies. Note that each unique $G_i$ only needs to be decomposed once. Subsequent occurrences of $G_i$ and $G_i^R$ simply involve copying the appropriate gate cascade (using reverse order for $G_i^R$).

The choice of the initial partitioning of the given MCT gate is key to the procedure. This is done in [71] as follows for a gate with $c$ controls and $p$ ancillary lines:

1. Assign $\alpha_i, 1 \leq i \leq p-1$ to 1.

2. Assign $\alpha_0$ and $\alpha_p$ such that $\alpha_0 + \alpha_p = c - p + 1$ and $\alpha_p - \alpha_0 = 1$ or 2.

Item 1 assigns $\alpha_1$ to $\alpha_{p-1}$ to yield Toffoli gates which occur in pairs positioned to allow substitution of Peres and inverse-Peres gates as introduced by Maslov and Dueck and possible redundant quantum gate removal. Item 2 distributes the remaining control inputs to the gates associated with $\alpha_0$ and $\alpha_p$. There is a single pair of gates for each so it is best to use the larger control sets here and it is best to balance the number of inputs to go towards smaller gates quickly. This also ensures there will be the maximum ancillary lines required when these gates are subsequently decomposed. The gates associated with $\alpha_0$ are assigned 1 or 2 fewer control inputs since they have an ancillary line as a control.

### 5.1.2 Improving NCV cost of MCT Gates

An earlier version of our Gate Reduction Procedure (Procedure 3.1) that utilizes the Line Labeling Procedure (Procedure 3.2) was used in [105] to improve the results of the mapping approach introduced by Miller in [71]. Applying the optimization improved the quantum costs of MCT gates and lead to the following conjecture:

**Conjecture 5.1.** *The minimum number of gates in an NCV realization of an MCT gate with c controls and p ancillary lines is given by $N(c, p)$, where*

$$N(c,p) = \begin{cases} 12c - 22 & \text{if } p = c - 2 \\ 12c - 20 & \text{if } p = c - 3 \\ 12c - 12 & \text{if } p = c - 4 \\ 24c - 12p - 64 & \text{if } p \leq c - 5 \end{cases} \tag{5.2}$$

The NCV gate counts for all the MCT gates up to 15 controls with 1 to the number of controls minus 2 ancillary lines are shown in Table 5.1. Note that the count for 3 controls and 1 ancillary line is 14. A 13 gate circuit is given in [5], but that circuit contains the fourth root of NOT and its inverse. The second column shows the gate counts for the work in [4] for the case of one ancillary line. Those results were found using heuristic synthesis and optimization and are not as good as the results for the systematic procedure presented here. The results are also improved in comparison to those given by the earlier version of the procedure [71].

Table 5.1: Number of NCV gates required for MCT gates for $c = 3 \ldots 15$ control lines and $1 \ldots c-2$ ancillary lines

| | [4] | **Number of Ancillary Lines** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Controls | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 3 | 15 | 14 | | | | | | | | | | | | |
| 4 | 37 | 28 | 26 | | | | | | | | | | | |
| 5 | 54 | 48 | 40 | 38 | | | | | | | | | | |
| 6 | 80 | 68 | 60 | 52 | 50 | | | | | | | | | |
| 7 | 100 | 92 | 80 | 72 | 64 | 62 | | | | | | | | |
| 8 | 128 | 116 | 104 | 92 | 84 | 76 | 74 | | | | | | | |
| 9 | 152 | 140 | 128 | 116 | 104 | 96 | 88 | 86 | | | | | | |
| 10 | 176 | 164 | 152 | 140 | 128 | 116 | 108 | 100 | 98 | | | | | |
| 11 | — | 188 | 176 | 164 | 152 | 140 | 128 | 120 | 112 | 110 | | | | |
| 12 | — | 212 | 200 | 188 | 176 | 164 | 152 | 140 | 132 | 124 | 122 | | | |
| 13 | — | 236 | 224 | 212 | 200 | 188 | 176 | 164 | 152 | 144 | 136 | 134 | | |
| 14 | — | 260 | 248 | 236 | 224 | 212 | 200 | 188 | 176 | 164 | 156 | 148 | 146 | |
| 15 | — | 284 | 272 | 260 | 248 | 236 | 224 | 212 | 200 | 188 | 176 | 168 | 160 | 158 |

The top entry for each number of ancillary lines equals the conjectured number of gates ($c \times 12 - 22$) when the maximum $c-2$ ancillary lines needed are available [84]. The increment by 12 can be seen to apply down each diagonal. Conjecture 5.1 can be readily verified from the table.

**Example 5.1.** *Consider realizing a 3-control MCT gate, $T(c_2, c_1, c_0; t)$, using the Barenco structure given one ancillary line $a$ is available as shown in Figure 5.5. This can be mapped to NCV gates using the mapping method of [71] as follows:*

1. *Expand the leftmost gate, $T(c_0, a; t)$ using the Toffoli gate realization in Figure 5.1.*

2. *Expand the next gate, $T(c_2, c_1, a)$ using the Toffoli gate realization in Figure 5.1.*

3. *Expand the next gate, $T(c_0, a; t)$ using the inverse realization of the Toffoli gate (Property 2.3).*

4. *Expand the last gate, $T(c_2, c_1, a)$ using the inverse realization of the Toffoli gate (Property 2.3).*

*The resulting circuit has 20 gates. But, a pair of gates from the first and third Toffoli gates cancel and two pairs of gates from the second and fourth Toffoli gates cancel. Thus, the 14 gate circuit as shown in Figure 5.6 results.*

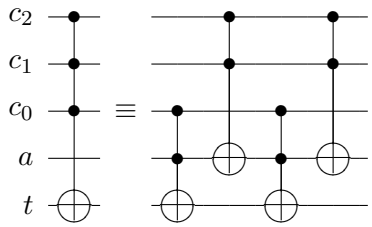



Figure 5.5: A decomposition of $T(c_2, c_1, c_0; t)$ with one ancillary line $a$.

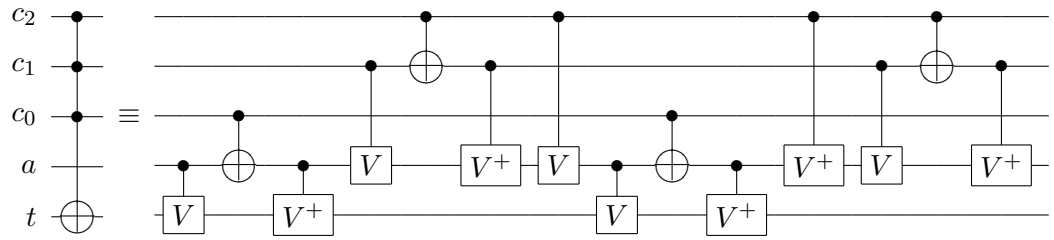



Figure 5.6: NCV circuit for $T(\{c_2, c_1, c_0\}; t)$ with ancillary line $a$.

The circuit just presented suggests a way to realize an MCT gate with more control lines as an NCV circuit. For example, consider the circuit in Figure 5.6. We can insert a fourth control line (labeled $x$) by changing the two $T(c_0; a)$ gates into Toffoli gates incorporating the new control. By expanding these gates in the manner outlined above, we obtain the circuit in Figure 5.7 which has a cost of 20 [106].

The idea of replacing the two CNOT gates in Figure 5.6 as shown above can be extended to more controls and combined with the general form of the decomposition illustrated in Figure 5.5 [106]. This leads to a new decomposition structure given by the following equation where $C = C_0 \cup C_1$ and $C_0 \cap C_1 = \phi$:

$$\begin{aligned} T(C;t) &= V(a_0;t)T_0(C_0;a_0)V^+(a_0;t)T_1(C_1;a_0) \\ &\quad V(a_0;t)T_2(C_0;a_0)V^+(a_0;t)T_3(C_1;a_0) \end{aligned} \tag{5.3}$$

An example of this decomposition for 8 controls and 3 ancillary lines is illustrated in Figure 5.8.

Figure 5.1 is the optimal realization for a Toffoli gate. We also believe Figure 5.6 is the optimal NCV realization for a 3-control MCT gate with 1 ancillary line. In particular, a SAT-based search has not found a simpler circuit [111]. These two circuits form the basis for our decomposition method which builds a catalog of NCV

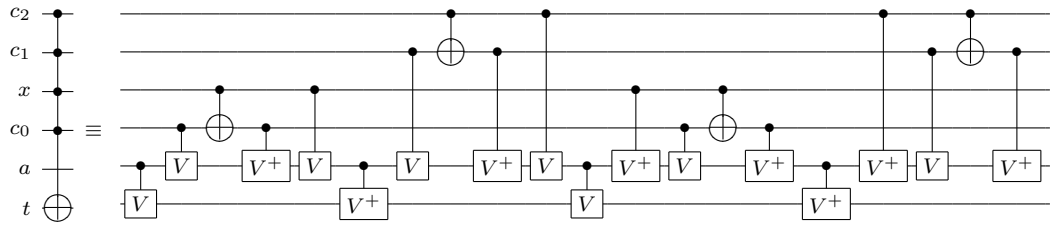



Figure 5.7: $T(c_2, c_1, x, c_0; t)$ with one ancillary $a$.

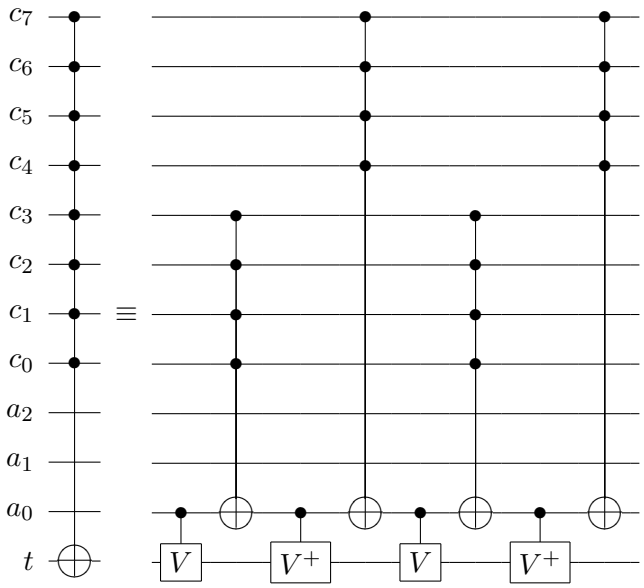



Figure 5.8: Illustration of the decomposition in Equation 5.3.

realizations of MCT gates for successively higher numbers of controls.

**Procedure 5.1.** *To determine a circuit with $p$ controls and $q$ ancillary lines $1 \leq q \leq p-2$, our method proceeds as follows:*

1. *The lines are ordered as shown in Figure 5.8,* i.e. *target line followed by the ancillary lines followed by the control lines.*

2. *The set of controls $C$ ($|C| = p$) is partitioned as $C_0C_1$ in all possible ways where $|C_0| >= 1$, $|C_1| >= 1$, and the order of the controls from $C$ is preserved across $C_0$ and $C_1$. Note that there are $|C| - 3$ such partitionings. It is not necessary to try all permutations since reordering of the controls of an MCT gate has no effect on its operation.*

3. *For each partitioning of $C$, Equation 5.3 is applied. Gate $T_0$ is replaced by the optimized circuit from the catalog. The number of ancillary lines available is*

*$q + |C_1|$. $T_2$ is replaced with the same circuit reversed with the $V$ and $V^+$ gates interchanged. Likewise, gate $T_1$ is replaced by the optimized circuit from the catalog. The number of ancillary lines available is $q + |C_0|$. $T_3$ is replaced with the same circuit reversed with the $V$ and $V^+$ gates interchanged.*

4. *The NCV circuit for each partitioning is simplified using the NCV optimization procedure described in Section 3.3.3.*

5. *As the possible partitionings are tried, record is kept of which one leads to the circuit with the fewest NCV gates. That circuit becomes the catalog entry for $p$ controls and $q$ ancillary lines.*

A critical factor in the above procedure is how the ancillary lines are assigned when replacing each $T_i$ gate with a catalog circuit. For $T_0$ and $T_2$, $t$ is used as the first ancillary, followed by the ancillary lines except $a_0$ in order followed by controls from $C_1$ as needed. For $T_1$ and $T_3$, the ancillary lines except $a_0$ are used in order followed by controls from $C_0$ as needed. Experimentation has shown this leads to the best circuits, but so far we have no proof that it is an optimal approach.

Note that the structure of Figure 5.8 requires only one ancillary line. However, it utilizes the available ancillaries to reduce the number of quantum gates required for realizing reversible gates. Table 5.2 shows the results of using the new decomposition structure for calculating the NCV cost of MCT gates with up to 15 controls. The first column of this table shows the number of controls in MCT gates. The next seven columns show the state-of-the-art NCV costs presented in [106] for MCT gates with different number of ancillaries ranging from 0 to 6. The last two columns of Table 5.2 show the NCV costs reported in [84] for MCT gates with $\{1 \dots n-4\}$ and $n-3$ ancillaries respectively. The results are not compared to REVLIB quantum costs as they are based on a different library containing roots of the NOT gate.

The table is restricted to 15 controls for space reasons; however, the method applies to any number of controls. Note that in each row, allowing further ancillary lines does not reduce the size of the circuit. For example, the smallest circuit for an MCT gate with 15 controls is achieved using only 6 ancillary lines while previously, 13 ancillaries were required [84].

The left part of Table 5.2 gives the state-of-the-art NCV costs for individual MCT gates. It shows significant improvement compared to [84] and the prior works summarized in Table 5.1. In the next section, realizing MPMCT gates with negative controls using the NCV library will be discussed.

Table 5.2: NCV realizations of MCT gates of size $n = [1 \dots 16]$ with ancillaries up to $n - 3$.

| | **Number of Ancillary Lines** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **NCV Costs of [106]** | | | | | | | **NCV Costs of [84]** | |
| **Controls c = n − 1** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **[1 . . . n − 4]** | **n − 3** |
| **0** | 1 | | | | | | | 1 | 1 |
| **1** | 1 | | | | | | | 1 | 1 |
| **2** | 5 | | | | | | | 5 | 5 |
| **3** | — | 14 | | | | | | 14 | 14 |
| **4** | — | 20 | | | | | | 32 | 26 |
| **5** | — | 32 | | | | | | 56 | 38 |
| **6** | — | 44 | | | | | | 80 | 50 |
| **7** | — | 64 | 56 | | | | | 104 | 62 |
| **8** | — | 76 | 68 | | | | | 128 | 74 |
| **9** | — | 96 | 88 | 80 | | | | 152 | 86 |
| **10** | — | 108 | 100 | 92 | | | | 176 | 98 |
| **11** | — | 132 | 120 | 112 | 104 | | | 200 | 110 |
| **12** | — | 156 | 132 | 124 | 116 | | | 224 | 122 |
| **13** | — | 180 | 156 | 148 | 136 | 128 | | 248 | 134 |
| **14** | — | 204 | 180 | 172 | 148 | 140 | | 272 | 146 |
| **15** | — | 228 | 204 | 198 | 172 | 160 | 152 | 296 | 158 |

### 5.1.3 Negative Controls

It has been shown that a 2-control MPMCT gate can be realized using 5 NCV gates without an ancillary line if at least one of its controls is positive, while an MPMCT gate with two negative controls and no ancillary lines can be realized by six NCV gates [7], see Figure 5.9.

Realization of mixed-polarity MCT gates using the NCV library has been studied in [7, 84]. A trivial solution to this problem is to add NOT gates wherever negative controls occur and use the corresponding positive control MPMCT gate realization [13]. In [84], Maslov *et al.* proved that except for the case where all the controls are negative, it is possible to implement MPMCT gates with the same cost as the corresponding positive control MPMCT gates using Barenco's decomposition [5] and their improved method [84]. According to [84], the NCV cost of MPMCT gates of size $n$ ($n > 3$) with less than $n - 2$ negative controls and $n - 3$ ancillaries is $12n - 34$. For the case where all the controls are negative, the cost is increased by two *i.e.* $12n - 32$ (two NOT gates will be placed on an arbitrary control line at the two ends of the NCV cascade). In the same manner, the cost of an MPMCT gate of size $n$ with less

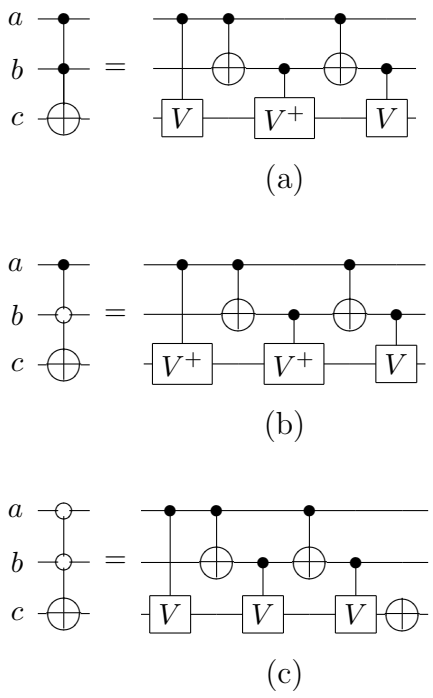



Figure 5.9: NCV realizations of (a) $T(a, b; c)$, (b) $T(a, \overline{b}; c)$, (c) $T(\overline{a}, \overline{b}; c)$ [7].

than $n - 2$ negative controls and 1 ancillary line is $24n - 88$ (for $n > 5$). The cost is $24n - 86$ for the case where all the controls are negative.

In the previous section, a new decomposition method was introduced that improved the NCV cost of positive control MPMCT gates. In this section, for mapping MPMCT gates with mixed-polarity controls to the NCV library, we use the decomposition method of the previous section and incorporate the realization of MPMCT gates that have negative controls using the procedure below.

A 2-control MPMCT gate is realized using the appropriate circuit from Figure 5.9. For realizing an MPMCT gate $T(C; t)$ with more than 2 controls and $m$ positive controls the following steps are applied. Note that for an MPMCT gate $T(C; t)$ ($|C| > 2$) to be realized using the NCV library, it is required to have at least one ancillary line [5].

**Procedure 5.2.** *For all possible ways of partitioning the set of controls $C$ into two disjoint sets $C_1$ and $C_2$ ($|C_1| > 0$ and $|C_2| > 0$) repeat the following steps: (Note that the permutation of controls in MPMCT gates is not considered in this partitioning.)*

1. *Partition the m positive controls into two parts of size $m_1$ and $m_2$ where*

$$m_1 = round(\frac{|C_1|}{|C|} \times m) \qquad (5.4)$$
$$m_2 = m - m_1$$

*such that ($m_1 > 0$ and $m_2 > 0$ if $m > 1$), and then assign the positive controls to the corresponding MPMCT gates so that $m_1$ is the number of positive controls in $C_1$ and $m_2$ is the number of positive controls in $C_2$.*

2. *Build the NCV circuit realization of $T(C;t)$ by looking up the corresponding circuits in the catalog table for the two partitions and substituting them in the following equation.*

$$\begin{aligned} T(C;t) &= V(a;t)T(C_1;a)V^+(a;t)T(C_2;a) \\ &\quad V(a;t)T_I(C_1;a)V^+(a;t)T_I(C_2;a) \end{aligned} \qquad (5.5)$$

*Note that the $T_I(C_1;a)$ and $T_I(C_2;a)$ gates are substituted by the inverse realization of the corresponding MPMCT gates for possible cancelations.*

3. *In the previous step, if $m_i = 0$, meaning that the corresponding MPMCT gate has all negative controls, replace the circuit of $T(C_i;a)$ with $T(;c_j)T(C_i';a)T(;c_j)$, which makes $c_j \in C_i'$ a positive control.*
4. *The NCV circuit resulting from substituting the partitions is simplified using the NCV optimization procedure described Section 2.3.3.*

*The minimum cost circuit found for $T(C;t)$ with m positive controls is stored in the catalog.*

**Example 5.2.** *Figure 5.10 shows the decomposition of the MPMCT gate $T(\overline{c_5}, c_4, \overline{c_3}, c_2, \overline{c_1}, c_0; t)$ as an example. As illustrated in this decomposition, the three positive controls are distributed in two partitions so that the ratio of the negative controls to positive controls stays the same in the partitions. We believe that this heuristic leads to cheaper NCV realizations. Figure 5.11 shows the corresponding NCV circuit for the decomposition structure of Figure 5.10.*

Using Procedure 5.2 a catalog of NCV circuits is constructed for MPMCT gates with different numbers of controls, negative controls, and ancillaries up to $n - 3$. These realizations can be used to map any MPMCT circuit to an NCV circuit by simply substituting the corresponding realizations from the catalog.

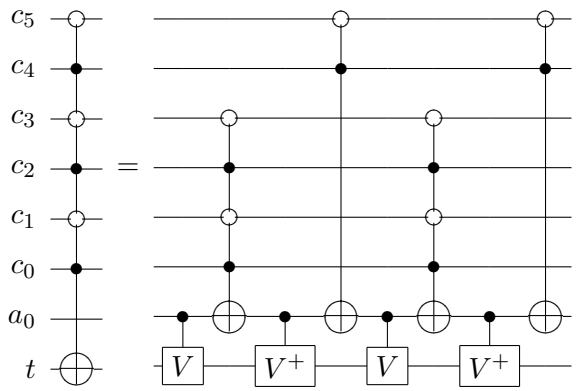



Figure 5.10: The decomposition structure for $T(\overline{c_5}, c_4, \overline{c_3}, c_2, \overline{c_1}, c_0; t)$.

### 5.1.4 Experimental Results

The catalog for the NCV library contains circuits for MPMCT gates of size $n$ ($n > 0$) with all possible cases of negative controls (*i.e.* $0 \ldots n-1$) and available ancillary lines (*i.e.* $1 \ldots n-3$). However, to compare with the work in the literature, the results reported here are only for the two extremes of available ancillary lines, 1 and $n-3$. We report the cost of MPMCT gates of size up to 16; nevertheless, it is possible to run the program for larger MPMCT gates.

The elapsed time to generate a catalog of size up to 15-control MPMCT gates is 27 minutes; however, it should not be given too much weight as our implementation has not been optimized and uses many heuristics that need further development. Moreover, once the catalog is built, there is no need to build it again and it can be used to map any MPMCT circuit to its NCV realization.

The results of our method for realizing MPMCT gates with mixed controls using the NCV library (Procedure 5.2) are compared to the state-of-the-art NCV costs for MPMCT gates with mixed controls presented in [84]. Tables 5.3 and 5.4 show the results of our method versus [84]. Table 5.3 shows the results when there is only one ancillary line available and Table 5.4 shows the results for the case when $n-3$ ancillary lines are available, where $n$ is the size of the corresponding MPMCT gate.

In both tables, the first column is the number of controls in the MPMCT gates, the next two columns show the NCV cost of MPMCT gates with some but not all negative controls and all negative controls respectively using the method in [84]. Columns 4 to 17 show the results of our method when the number of negative controls changes from 0 to $n-3$.

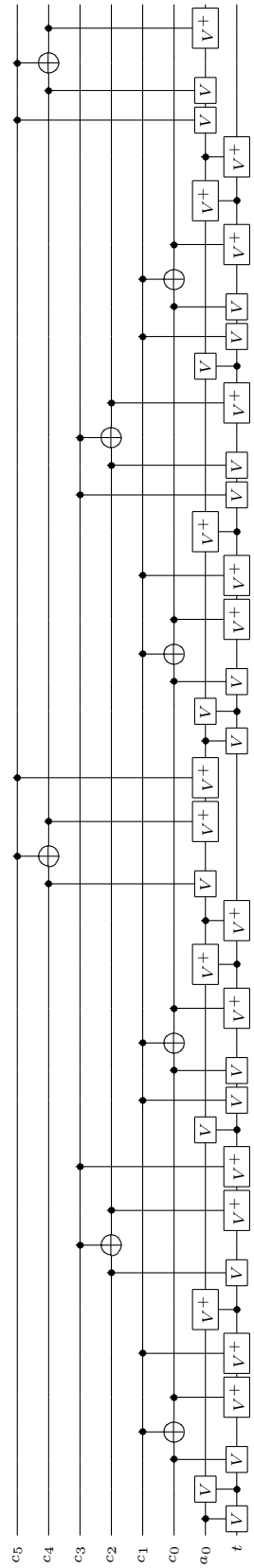



Figure 5.11: NCV realization of $T(\overline{c_5}, c_4, \overline{c_3}, c_2, \overline{c_1}, c_0; t)$.

Using the method in [84] as illustrated in the second and third columns of Table 5.3, the NCV cost of MPMCT gates of size $n$ ($n > 5$) with 0 to $c-1$ negative controls ($c = n-1$ is the number of controls) and one ancillary line is equal to $24n - 88$, and for all negative control MPMCT gates, this cost is incremented by 2. Similarly, using the method in [84], MPMCT gates of size $n$ ($n > 3$) with some but not all negative controls and $n-3$ ancillaries can be realized by $12n - 34$ NCV gates as shown in the second column of Table 5.4 and this number is incremented by 2 when all the controls are negative as shown in the third column of Table 5.4. The columns 4 to 17 of Table 5.3 and Table 5.4 show that using our method, the NCV cost of MPMCT gates with $m$ negative controls $m = \{0, \ldots, c-3\}$ stays the same as the cost of MPMCT gates with all positive controls until $m = \lfloor c/2 \rfloor$ and is incremented by 2 for each additional negative control after that.

Table 5.3 shows that when there is only one ancillary available, our method outperforms the state-of-the-art method of [84] by 31.08% on average. Since our method utilizes all available ancillaries, it yields lower costs when the number of ancillaries is more than 1 and less than $n-3$ comparing to [84] that uses the same cost as for one available ancillary. As a result, for MPMCT gates with ancillaries in the range $\{2, \ldots, n-4\}$ our method shows even more improvement. For example, if the number of ancillaries is in the range $\{\lceil c/2 \rceil - 2, \ldots, n-4\}$, the cost of MPMCT gates would be the same as when there is $n-3$ ancillaries available, which gives 43.5% improvement over [84]. To compare our method with existing methods, we here show the results our method for the two extremes (1 and $n-3$ ancillaries). however, the intermediate NCV costs can be computed by using the costs reported in [106] and following the same growth rate for negative controls.

Table 5.4 shows that our method introduces cheaper realizations for MPMCT gates of size $n$ when there is $n-3$ ancillaries available and the number of negative controls is less than or equal to $\lfloor c/2 \rfloor + 3$ where $c$ is the number of controls. However, if the number of negative controls is more than $\lfloor c/2 \rfloor + 3$, our method generates more expensive realizations than those introduced in [84].

The results show that the proposed method reduces the best known NCV cost of MPMCT gates with mixed controls by 33.73% on average. The results yield cheaper, and faster, realizations for circuits composed of MPMCT gates by simple substitution of a quantum realization for each MPMCT gate.

Table 5.3: NCV cost of MPMCT gates of size $n = 4 \ldots 16$ with one ancillary line.

| | **Number of Negative Controls** | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **[84]** | | **Our method** | | | | | | | | | | | | | | | |
| Controls $c = n - 1$ | [0 ... c-1] | c | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 3 | — | — | 14 | 14 | 16 | 18 | | | | | | | | | | | | |
| 4 | — | — | 20 | 20 | 20 | 22 | 24 | | | | | | | | | | | |
| 5 | 56 | 58 | 32 | 32 | 32 | 34 | 36 | 38 | | | | | | | | | | |
| 6 | 80 | 82 | 44 | 44 | 44 | 44 | 46 | 48 | 50 | | | | | | | | | |
| 7 | 104 | 106 | 64 | 64 | 64 | 64 | 66 | 68 | 70 | 72 | | | | | | | | |
| 8 | 128 | 130 | 76 | 76 | 76 | 76 | 76 | 78 | 80 | 82 | 84 | | | | | | | |
| 9 | 152 | 154 | 96 | 96 | 96 | 96 | 96 | 98 | 100 | 102 | 104 | 106 | | | | | | |
| 10 | 176 | 178 | 108 | 108 | 108 | 108 | 108 | 108 | 110 | 112 | 114 | 116 | 118 | | | | | |
| 11 | 200 | 202 | 132 | 132 | 132 | 132 | 132 | 132 | 134 | 136 | 138 | 140 | 142 | 144 | | | | |
| 12 | 224 | 226 | 156 | 156 | 156 | 156 | 156 | 156 | 156 | 158 | 160 | 162 | 164 | 166 | 168 | | | |
| 13 | 248 | 250 | 180 | 180 | 180 | 180 | 180 | 180 | 180 | 182 | 184 | 186 | 188 | 190 | 192 | 194 | | |
| 14 | 272 | 274 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 204 | 206 | 208 | 210 | 212 | 214 | 216 | 218 | |
| 15 | 296 | 298 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 230 | 232 | 234 | 236 | 240 | 242 | 244 | 246 |

Table 5.4: NCV cost of MPMCT gates of size $n = 4 \ldots 16$ with $n - 3$ ancillary lines.

| | **Number of Negative Controls** | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **[84]** | | **Our method** | | | | | | | | | | | | | | | |
| Controls $c = n - 1$ | [0 ... c-1] | c | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 3 | 14 | 16 | 14 | 14 | 16 | 18 | | | | | | | | | | | | |
| 4 | 26 | 28 | 20 | 20 | 20 | 22 | 24 | | | | | | | | | | | |
| 5 | 38 | 40 | 32 | 32 | 32 | 34 | 36 | 38 | | | | | | | | | | |
| 6 | 50 | 52 | 44 | 44 | 44 | 44 | 46 | 48 | 50 | | | | | | | | | |
| 7 | 62 | 64 | 56 | 56 | 56 | 56 | 58 | 60 | 62 | 64 | | | | | | | | |
| 8 | 74 | 76 | 68 | 68 | 68 | 68 | 68 | 70 | 72 | 74 | 76 | | | | | | | |
| 9 | 86 | 88 | 80 | 80 | 80 | 80 | 80 | 82 | 84 | 86 | 88 | 90 | | | | | | |
| 10 | 98 | 100 | 92 | 92 | 92 | 92 | 92 | 92 | 94 | 96 | 98 | 100 | 102 | | | | | |
| 11 | 110 | 112 | 104 | 104 | 104 | 104 | 104 | 104 | 106 | 108 | 110 | 112 | 114 | 116 | | | | |
| 12 | 122 | 124 | 116 | 116 | 116 | 116 | 116 | 116 | 116 | 118 | 120 | 122 | 124 | 126 | 128 | | | |
| 13 | 134 | 136 | 128 | 128 | 128 | 128 | 128 | 128 | 128 | 130 | 132 | 134 | 136 | 138 | 140 | 142 | | |
| 14 | 146 | 148 | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 142 | 144 | 146 | 148 | 150 | 152 | 154 | |
| 15 | 158 | 160 | 152 | 152 | 152 | 152 | 152 | 152 | 152 | 152 | 154 | 156 | 158 | 160 | 162 | 164 | 166 | 168 |

## 5.2 NCVW Library

In this section, we generalize the MPMCT to NCV decomposition structure to target a new quantum gate library, NCVW, containing the NCV gates plus quantum gates *controlled*$-W$ and *controlled*$-W^+$ which each implement a fourth-root of NOT. We also show how to extend the approach to map MCT gates with a mixture of positive and negative controls.

### 5.2.1 Basic Concepts

It was shown in Chapter 2 that the operation of a quantum gate on its target line can be defined as a $2 \times 2$ unitary matrix and a quantum gate in an $n$ line circuit is given by a $2^n \times 2^n$ unitary matrix (see [13] for details).

**Theorem 5.1.** *Consider the matrix*

$$\mathbf{R}_k = \frac{1}{2}\begin{pmatrix} 1+i^{2/k} & 1-i^{2/k} \\ 1-i^{2/k} & 1+i^{2/k} \end{pmatrix} \tag{5.6}$$

*where $k$ is a power of 2. $\mathbf{R}_k$ is the corresponding matrix for a $k$-th root of the NOT gate, i.e. $(\mathbf{R}_k)^k = \mathbf{N}$.*

*Proof.* Consider

$$\mathbf{R}_p{}^2 = \left[\frac{1}{2}\begin{pmatrix} 1+i^{2/p} & 1-i^{2/p} \\ 1-i^{2/p} & 1+i^{2/p} \end{pmatrix}\right]^2 \tag{5.7}$$

$$= \frac{1}{2}\begin{pmatrix} 1+i^{4/p} & 1-i^{4/p} \\ 1-i^{4/p} & 1+i^{4/p} \end{pmatrix} \tag{5.8}$$

The matrix in Equation 5.8 is $\mathbf{R}_{p/2}$ which is verified by setting $k = p/2$ in Equation 5.6. Since $\mathbf{R}_p{}^2 = \mathbf{R}_{p/2}$ and $\mathbf{R}_1 = \mathbf{N}$, it follows by induction that for $k$ a power of 2, $(\mathbf{R}_k)^k = \mathbf{N}$. □

**Theorem 5.2.** *Consider the matrix*

$$\mathbf{R}_k^+ = \frac{1}{2}\begin{pmatrix} 1+i^{-2/k} & 1-i^{-2/k} \\ 1-i^{-2/k} & 1+i^{-2/k} \end{pmatrix} \tag{5.9}$$

*where $k$ is a power of 2. $\mathbf{R}_k^+$ is the corresponding matrix for a $k$-th root of the NOT gate which is the inverse of $\mathbf{R}_k$, i.e. $(\mathbf{R}_k^+)^k = \mathbf{N}$.*

*Proof.* $\mathbf{R}_k^+$ is the adjoint (complex conjugate transpose) of $\mathbf{R}_k$ and also $\mathbf{R}_k\mathbf{R}_k^+ = I$. Hence, $\mathbf{R}_k^+ = \mathbf{R}_k^{-1}$ which means $\mathbf{R}_k$ and $\mathbf{R}_k^+$ are unitary matrices that realize inverse quantum gates. From Theorem 5.1, $(\mathbf{R}_k)^k = \mathbf{N}$. By taking the conjugate of both sides given that the conjugate of the product of two matrices is the product of their conjugates, $(\mathbf{R}_k^+)^k = \mathbf{N}$. □

In the construction below, we make use of $V$ and $V^+$ gates with two controls as intermediate gates.

**Definition 5.1.** *The three line* **controlled-controlled-$V$ gate**, *denoted $C^2V$, applies the transformation of Equation 2.11 to the target line whenever both of the control lines are set to one. Similarly, the three line* **controlled-controlled-$V^+$ gate**, *denoted $C^2V^+$, applies the transformation shown in Equation 2.12 to the target whenever both of its control lines are one.*

We next introduce gates realizing a fourth-root of NOT.

**Definition 5.2.** *A* **controlled-$W$ gate** *applies the transformation defined by the matrix in Equation 5.10 (a) to the target line if its control line is set to 1. A* **controlled-$W^+$ gate** *changes the target line using the transformation shown in Equation 5.10 (b) if its control line is 1. Gates $W$ and $W^+$ are referred to as* **controlled-fourth-root of NOT gates** *since $\mathbf{W} = \mathbf{R}_4$ and $\mathbf{W}^+ = \overline{\mathbf{R}}_4$, hence $\mathbf{W}^4 = (\mathbf{W}^+)^4 = \mathbf{N}$. Clearly, $\mathbf{W}$ is a square-root of $\mathbf{V}$ and $\mathbf{W}^+$ is a square-root of $\mathbf{V}^+$.*

$$\mathbf{W} = \frac{1}{2}\begin{pmatrix} 1+\sqrt{i} & 1-\sqrt{i} \\ 1-\sqrt{i} & 1+\sqrt{i} \end{pmatrix} \qquad (a) \tag{5.10}$$

$$\mathbf{W}^+ = \frac{1}{2}\begin{pmatrix} 1-i\sqrt{i} & 1+i\sqrt{i} \\ 1+i\sqrt{i} & 1-i\sqrt{i} \end{pmatrix} \qquad (b)$$

The $\mathbf{W}$ ($\mathbf{W}^+$) gate defines a counter-clockwise (clockwise) rotation about the $x$-axis by the angle $\pi/4$ radians on the Bloch sphere model of a qubit.

**Definition 5.3.** *The* **NCVW library** *is composed of the elementary quantum gates NOT, CNOT, controlled-$V$, controlled-$V^+$, controlled-$W$ and controlled-$W^+$.*

The gates in the NCVW library are considered to have unit quantum cost; thus, the quantum cost of an NCVW circuit is the number of gates in that circuit.

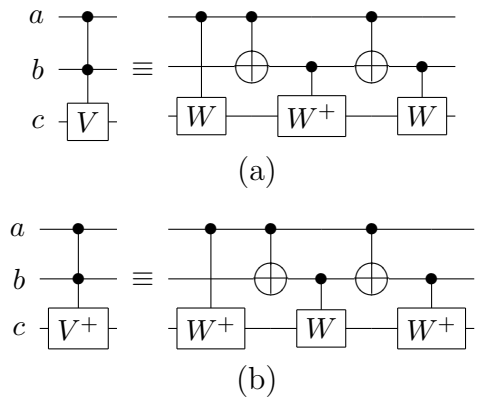



Figure 5.12: (a) NCW realization of $V(\{a, b\}; c)$. (b) NCW realization of $V^+(\{a, b\}; c)$.

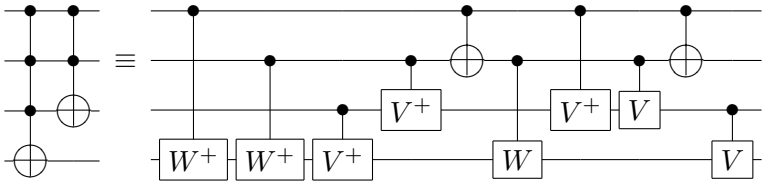



Figure 5.13: Example of NCVW realization of a pair of MCT gates [8].

Figure 5.12 shows the realizations of the $C^2V$ and $C^2V^+$ gates using controlled $W$ and controlled $W^+$ gates.

### 5.2.2 NCVW Realizations of MCT Gates

We here describe a procedure for mapping MCT gates to NCVW circuits. It is shown in the experimental results section that the procedure leads to substantial cost reductions and speedup in the realization of MCT gates. This assumes that the $W$ and $W^+$ gates can be physically realized with the same cost as other elementary quantum gates and can thus be assigned unit cost. This is reasonable since $W$ and $W^+$ gates implement rotations in a similar manner to $V$ and $V^+$ gates.

**Example 5.3.** *Figure 5.13 shows an NCVW realization for a pair of MCT gates [8]. The cost of the circuit is 10 which shows 45% improvement compared to the corresponding NCV realization that has* $14 + 5 = 19$ *gates. Note that to realize this pair in the NCV library, one ancillary line is required.*

As shown in Lemma 7.1 of Barenco *et al* [5], a 2 positive control MCT gate, a Toffoli gate [62], can be implemented using 5 NCV gates, Figure 5.14 (a). Barenco

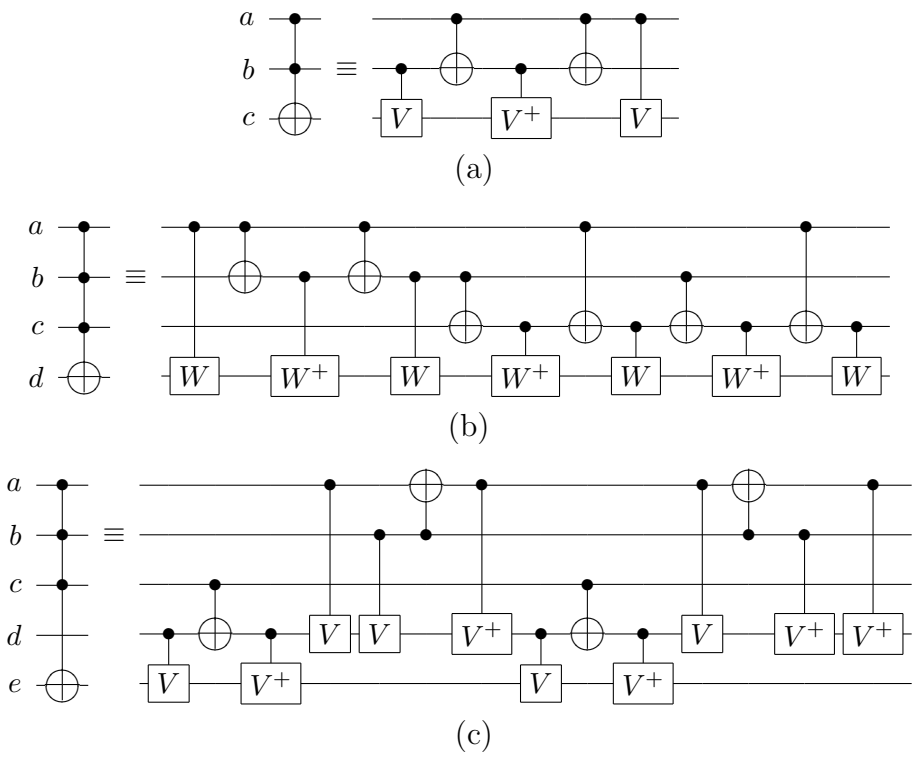



Figure 5.14: (a) NCV realization of Toffoli gate $T(\{a, b\}; c)$. (b) NCW realization of 3-control MCT gate $T(\{a, b, c\}; d)$ without ancillaries. (c) NCV realization of 3-control MCT gate $T(\{a, b, c\}; e)$ with one ancillary line, $d$.

*et al* further show that a 3 positive control MCT gate can be implemented using 13 NCW gates as shown in Figure 5.14 (b). Note that realizing a 3 positive control MCT gate using NCV gates requires an ancillary line and 14 gates [84] as shown in Figure 5.14 (c).

MCT gates with 4 or more controls can not be realized using the NCVW library without an ancillary line. To do that requires higher roots of $NOT$ gates, see [5]. However, by using at least one ancillary line, it is possible to realize any MCT gate with more than three controls using the NCVW library as we show below.

**Example 5.4.** *Consider the 20 NCV gate realization of a 4-control MCT gate that uses one ancillary line shown in Figure 5.15. If a fifth control, $c_4$, is inserted in the circuit by changing the $V(a; t)$ and $V^+(a; t)$ gates to $C^2V(\{c_4, a\}; t)$ and $C^2V^+(\{c_4, a\}; t)$ that incorporate the new control, and the circuits of Figure 5.12 is used to realize the resulting $C^2V$ and $C^2V^+$ gates, the circuit of Figure 5.16 is obtained. This circuit implements a 5-control MCT gate with cost 28.*

Note that we have used the inverse realizations for $C^2V(\{c_4, a\}; t)$ gates. The best

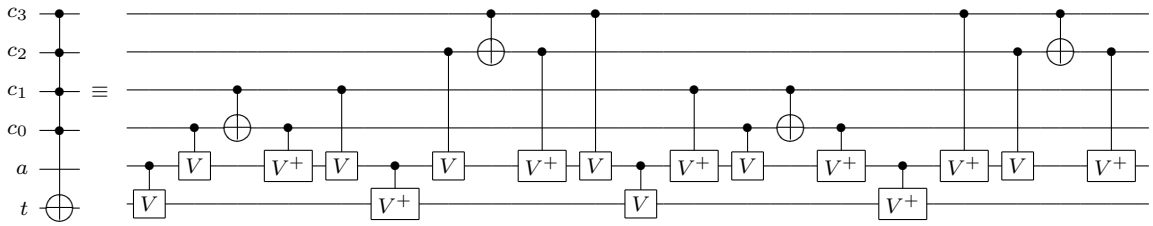



Figure 5.15: $T(\{c_3, c_2, c_1, c_0\}; t)$ with one ancillary $a$.

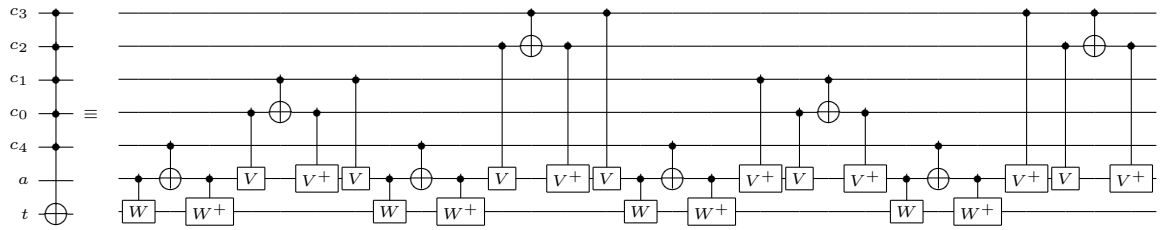



Figure 5.16: $T(\{c_3, c_2, c_1, c_0, c_4\}; t)$ with one ancillary $a$.

reported cost for the 5-control MCT gate using the NCV library is 32 [106]. This example illustrates how using smaller rotations can improve the cost of a circuit.

Figure 5.16 suggests a new structure for NCVW targeted decomposition of MCT gates. Below, we show how this new structure leads to a general form of decomposition for MCT gates using NCVW gates. We describe a procedure employing this extended decomposition to yield a table of NCVW gate realizations for MCT gates. The table shows lower costs than the best known for the NCV library [106].

### Decomposition Structure

Generalizing the structure illustrated in Figure 5.16, an MCT gate $T(C; t)$ can be decomposed into an NCVW circuit as described in the following steps. First the set of controls $C$ is partitioned into three non-empty disjoint sets $C_0$, $C_1$, and $C_2$ such that $C_0 \cup C_1 \cup C_2 = C$. Then, the structure given in Equation 5.11 is used to decompose the MCT gate into MCT gates each containing one of the disjoint sets as their controls. $a$ is an ancillary line. $T_I(C_i; t)$ represents the inverse realization (Property 2.3) of $T(C_i; t)$.

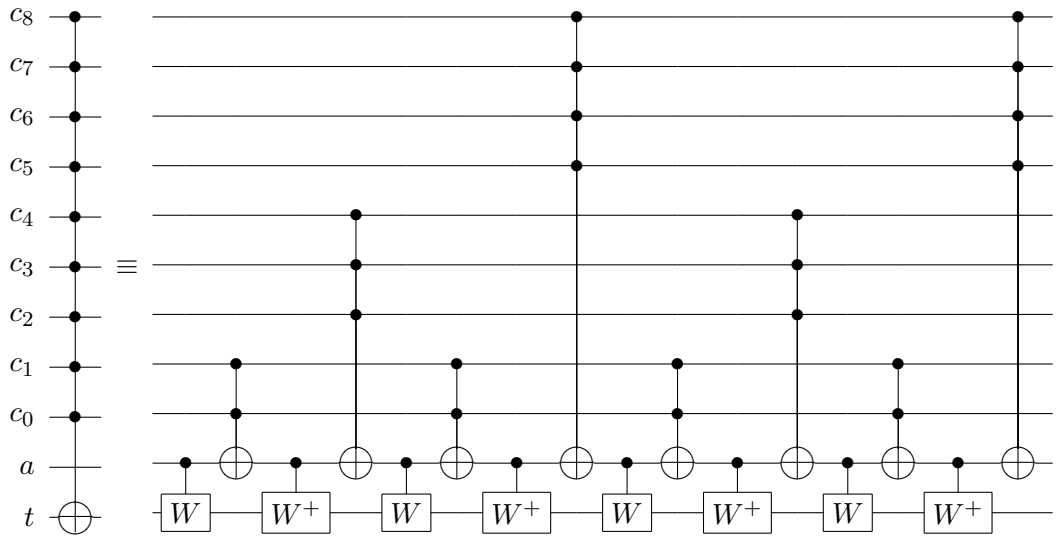



Figure 5.17: Illustration of the decomposition in Eqn. 5.11.

$$
\begin{aligned}
T(C;t) = \quad & W(a;t)T(C_0;a)W^+(a;t)T(C_1;a) \\
& W(a;t)T_I(C_0;a)W^+(a;t)T(C_2;a) \\
& W(a;t)T(C_0;a)W^+(a;t)T_I(C_1;a) \\
& W(a;t)T_I(C_0;a)W^+(a;t)T_I(C_2;a)
\end{aligned} \tag{5.11}
$$

Figure 5.17 shows one of the possible decompositions of an 8-control MCT gate using the decomposition structure of Equation 5.11. In this example, $C_0 = \{c_0, c_1\}$, $C_1 = \{c_2, c_3, c_4\}$, and $C_2 = \{c_5, c_6, c_7, c_8\}$; however, the set of controls can be partitioned in many other ways and the cost of the resulting circuit varies depending on the selected partitioning. In the following, a procedure is presented that searches for the best partitioning and finds a circuit with the least quantum cost among all possibilities.

The decomposition structure can be generalized to consider libraries containing higher roots of $NOT$. In this case, for a library that contains up to the $n$-th root of the $NOT$ gate, $R_n$ ($R_n^n = N$,$n = 2^m$, $m = 1, 2, ...$), there are $n$ $R_n$ gates and $n$ $R_n^+$ gates ($R_n R_n^+ = I$) on the target line and the set of controls is partitioned into $m + 1$ disjoint sets following the same structure.

### Decomposition Procedure

The decomposition procedure finds NCVW gate realizations for MCT gates. Two tables are used in this procedure: an NCV table that records the NCV realizations of MCT gates [106] and an NCVW table that records the final realization of MCT gates using NCVW gates. The procedure realizes an MCT gate by decomposition, so the tables are constructed by increasing values of the number of controls. The tables are indexed by the number of controls and the number of ancillary lines.

The circuits in Figure 5.14 (a) and 5.14 (b) are optimal realizations of the 2-control and 3-control MCT gates. Both require no ancillary line. The 20 gate realization of Figure 5.15 is the best NCVW realization that we could find for a 4-control MCT gate.

**Procedure 5.3. MCT Gate Decomposition Initialization:** *The NCV and NCVW tables are both initialized with a single NOT gate, a single CNOT gate, and the realization in Figure 5.14 (a) for 0, 1, and 2 controls respectively. No ancillaries are needed. For 3 controls, the NCV table entry is the circuit in Figure 5.14 (c), with one ancillary, whereas the the NCVW entry is the circuit in Figure 5.14 (b), with no ancillary line. The circuit in Figure 5.15 (one ancillary line) is used for the case of 4 controls for both tables since we could not find a better realization for this case using $W$ and $W^+$ gates.*

**Construction:** *The procedure considers a fixed order for the lines in an MCT gate as shown in Figure 5.17,* i.e. *the target followed by the ancillaries followed by the controls. Note that the order of the controls is irrelevant since all must be 1 to activate the gate.*

*For* $p = 5, 6, ...$

*For* $q = 1, 2, ..., p-2$*:*

- *The NCV realization of an MCT gate with $p$ controls and $q$ ancillary lines is determined using the procedure described in [106] and the circuit is placed in the appropriate entry in the NCV table.*
- *To find the best NCVW realization of an MCT gate with $p$ controls and $q$ ancillary lines, the decomposition structure of Equation 5.11 is applied for each partition of the set of controls $C$ into three disjoint non-empty subsets* i.e. *$C = \{C_0|C_1|C_2\}$. Note that since the order of the controls does not matter, there are $\sum_{k=1}^{p-2} k = (p-1)(p-2)/2$ such partitions for $p$ controls.*

*In each case, for each gate $T(C_i;t)$, $i = \{0,1,2\}$, in the decomposition, the corresponding realizations from the NCV table and the NCVW table are examined and the best combination is selected after applying the optimization method described in Section 3.3.3. Note that for each case, the $T_I(C_i;t)$ gate is replaced by the inverse realization of the optimized circuit for $T(C_i;t)$. The number of available ancillary lines for each gate $T(C_i;t)$ is $p+q-|C_i|$.*

- *Whichever of the NCV and the NCVW realizations has the fewer gates is used as the NCVW table entry for an MCT gate with p controls and q ancillary lines.*

An alternative approach to building the NCVW table is to start with the NCV decomposition [106] as the initial decomposition for step 1 of the algorithm and to try to substitute the subsequent MCT gates with their realizations from the NCV and NCVW tables. Experiments for up to 30-control MCT gates have verified that this alternative does not reduce the circuit gate count.

The order in which the ancillary lines are used to expand the $T(C_i;t)$ gates is an important factor that can change the final cost. For $T(C_0;a)$ and $T_I(C_0;a)$, $t$ is used as the first ancillary, followed by the ancillary lines, except the first, in order, followed by controls in $C_1$, followed by controls in $C_2$ as needed. For $T(C_1;a)$ and $T_I(C_1;a)$, $t$ is used as the first ancillary if it is not used in $T(C_0;a)$, followed by the ancillary lines except the first one in order, followed by controls in $C_0$ followed by controls in $C_2$ as needed. Finally, for $T(C_2;a)$ and $T_I(C_2;a)$, $t$ is used as the first ancillary if not used before, followed by the ancillary lines except the first one in order followed by controls in $C_0$ and $C_1$ respectively as needed. Experimental results have shown that this leads to the best circuits, but so far we have no proof that this approach is optimal.

### 5.2.3 Negative Controls

Some synthesis approaches use MPMCT gates with a mixture of positive and negative controls as their target library [44,89,112,113]. In this Section, the NCVW realization of MPMCT gates is discussed.

Figure 5.18 shows four different realizations of a 3-control MCT gate with (a) zero, (b) one, (c) two, and (d) three negative controls. As shown in the Figure, the NCVW cost of a 3-control MCT gate with some but not all negative controls is 13 and the NCVW cost of an all negative 3-control MCT gate is 14. Note that the realizations

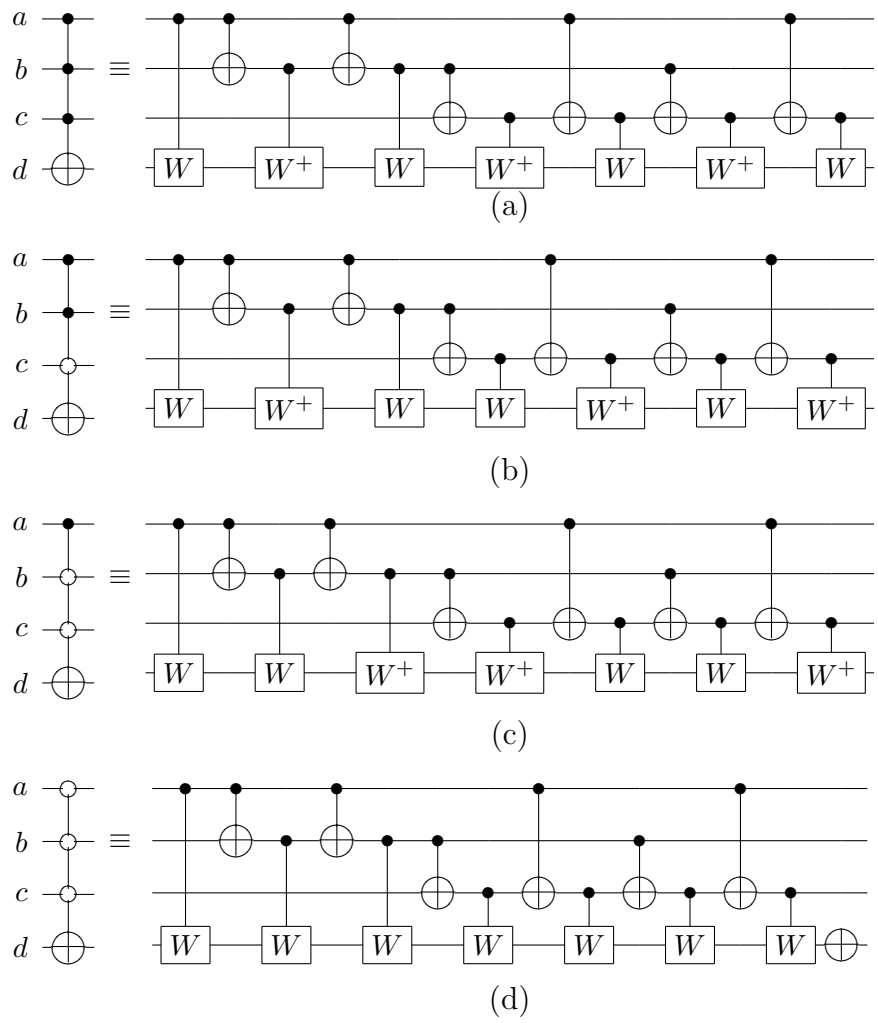



Figure 5.18: NCVW realizations of (a) $T(a, b, c; d)$, (b) $T(a, b, \overline{c}; d)$, (c) $T(a, \overline{b}, \overline{c}; d)$, (d) $T(\overline{a}, \overline{b}, \overline{c}; d)$.

in Figure 5.18 follow the same pattern as the realizations in Figure 5.9. Using the same approach, any MCT gate of size $n$ with some but not all negative controls can be realized by $2^n - 3$ quantum gates including the $2^{n-2}$-th root of NOT gate and no ancillaries, and an all negative control MCT gate of size $n$ can be realized by $2^n - 2$ quantum gates without extra ancillary lines.

NCVW circuits for larger MCT gates are constructed using the same decomposition structure as Figure 5.17 and following the approach described in Procedure 5.4. Note that for MCT gates of size 5 and higher, at least one ancillary line is required. However, the approach presented in Procedure 5.4 utilizes all available ancillaries (if there is more than 1 ancillary) to reduce the quantum cost. The result of the pro-

cedure is a table containing the realizations of MCT gates for different numbers of positive and negative controls and available ancillaries from 1 to $n-3$.

**Procedure 5.4. Mixed-Polarity MCT Gate Decomposition**
**Initialization:** *The NCV and NCVW tables are both initialized with a single NOT gate, a single CNOT gate, and the NCV realizations of 2-control MCT gates with different numbers of negative controls as in Figure 5.9. For 3 controls, the NCV table entry is filled using the method in Section 5.1.3, and The NCVW table is filled with the circuits shown in Figure 5.18. The method in Section 5.1.3 is also used to fill the entries for 4-control MCT gates in the NCV and NCVW tables.*
**Construction:** *As in Procedure 1, the lines in an MCT gate are ordered as the target followed by the ancillaries followed by the controls. However, in this case the controls are arranged as described below to balance the use of positive and negative controls across the circuit.*
*For* $p = 5, 6, ...$
*For* $r = 0, 1, ..., p$
*For* $q = 1, 2, ..., p-2$

- *The NCV realization of a p-control MCT gate with r negative controls,* $p-r$ *positive controls and q ancillary lines is determined using the procedures from Section 5.1.3 and the circuit is placed in the appropriate entry in the NCV table.*

- *To find the best NCVW realization of a p-control MCT gate with r negative controls,* $p-r$ *positive controls and q ancillary lines, consider* $(p-1)(p-2)/2$ *partitions of the set of controls C into three disjoint sets* $C_i$, $0 \leq i \leq 2$ *where each partition is handled as follows:*

  - *The* $m = p-r$ *positive controls are distributed so that there are* $m_i$ *positive controls in* $C_i$, $0 \leq i \leq 2$, *where*

$$m_0 = round(\frac{|C_0|}{|C|} \times m) \tag{5.12}$$

$$m_1 = round(\frac{|C_1|}{|C|} \times m) \tag{5.13}$$

$$m_2 = m - m_1 - m_0$$

  *and* $m_i > 0$ *if* $m > 2$. *Note that each* $C_i$ *contains* $|C_i| - m_i$ *negative controls.*

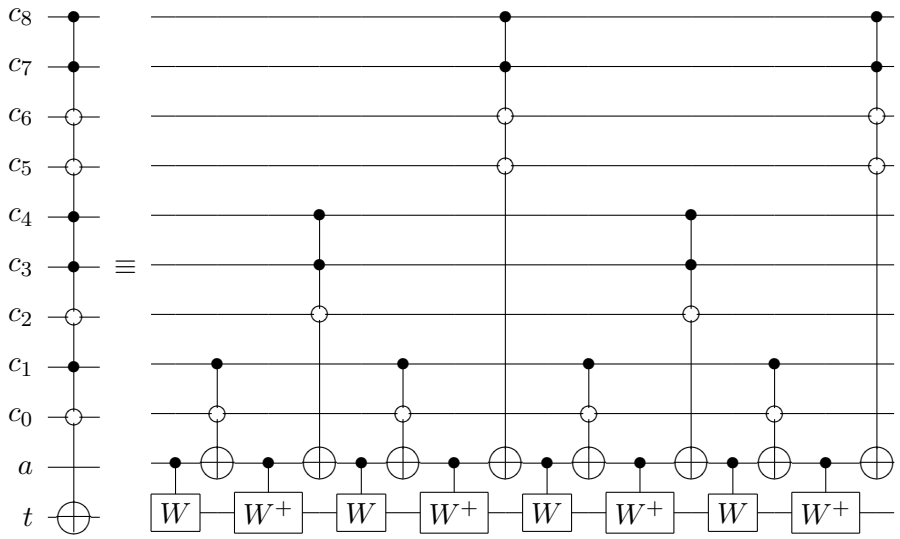



Figure 5.19: NCVW realization of $T(c_8, c_7, \overline{c_6}, \overline{c_5}, c_4, c_3, \overline{c_2}, c_1, \overline{c_0}; t)$.

  - *The decomposition structure of Equation 5.11 is applied. For each gate $T(C_i; t)$, $i = \{0, 1, 2\}$, in the decomposition, the corresponding realizations from the NCV table and the NCVW catalog are examined and the best combination is selected after applying the optimization method described in Section 3.3.3. Note that at each case, the $T_I(C_i; t)$ gate is replaced by the inverse realization of the optimized circuit for $T(C_i; t)$. The number of available ancillary lines for each gate $T(C_i; t)$ is $p + q - |C_i|$.*

    *Note that, if $m_i = 0$, meaning that all of the controls in the corresponding MCT gate are negative, the circuit of $T(C_i; a)$ is replaced with $T(; c_j)T(C_i'; a)$ $T(; c_j)$, where $c_j \in C_i'$ is a positive control.*

- *Whichever of the NCV and the NCVW realizations has the fewer gates is used as the NCVW table entry for an MCT gate with $r$ negative controls, $p - r$ positive controls and $q$ ancillary lines.*

**Example 5.5.** *Figure 5.19 shows the decomposition of the $T(c_8, c_7, \overline{c_6}, \overline{c_5}, c_4, c_3, \overline{c_2}, c_1, \overline{c_0}; t)$ gate as an example. The NCVW cost of this gate is 80.*

### 5.2.4 Experimental Results

The decomposition procedures for NCVW realization of MCT and MPMCT gates have been implemented using Python 2.7.1. The experiments were run on a computer

Table 5.5: NCVW realizations of MCT gates of size $n = [1 \dots 16]$ with ancillaries up to $n - 3$.

| | **Number of Ancillary Lines** | | | | | |
|---|---|---|---|---|---|---|
| **Controls c = n − 1** | **0** | **1** | **2** | **3** | **4** | **5** |
| **0** | 1 | | | | | |
| **1** | 1 | | | | | |
| **2** | 5 | | | | | |
| **3** | 13 | | | | | |
| **4** | — | 20 | | | | |
| **5** | — | 28 | | | | |
| **6** | — | 40 | | | | |
| **7** | — | 52 | | | | |
| **8** | — | 64 | | | | |
| **9** | — | 80 | 76 | | | |
| **10** | — | 96 | 88 | | | |
| **11** | — | 112 | 104 | 100 | | |
| **12** | — | 128 | 120 | 112 | | |
| **13** | — | 152 | 136 | 128 | 124 | |
| **14** | — | 176 | 158 | 144 | 136 | |
| **15** | — | 200 | 176 | 160 | 152 | 148 |

with a 3.2 GHz i5-650 CPU and 3.0 GB RAM. Table 5.5 shows the quantum cost of MCT gates with up to 15 controls using NCVW library versus the number of available ancillary lines ranging from zero to 5. Adding extra ancillaries does not improve the cost reported in the last entry of each row. All the NCVW costs are even due to the specific structure that we use for decomposition.

Comparing Tables 5.5 and the left part of Table 5.2 reveals that the NCVW realizations are consistently cheaper than the cheapest NCV circuits and for MCT gates with 7 or more controls one less ancillary line is required to achieve the smallest circuit. The results show 12.34% improvement on average over the state-of-the-art NCV realizations. The NCVW costs reported in Table 5.5 outperform the NCV costs reported in the second part of Table 5.2 by 43.42% on average.

Tables 5.6 and 5.7 show the results of implementing mixed-polarity MCT gates using the NCVW library. The results are reported for only two extremes of available ancillary lines 1 and $n - 3$. Here, we report the cost of MPMCT gates of size up to 16; nevertheless, it is possible to run the program for larger MPMCT gates. In both tables, the first column shows the number of controls of MPMCT gates and the next columns show the cost of the corresponding MPMCT gates with the number of negative controls ranging from 0 to $c$, where $c$ is the total number of controls. The columns 2 to 15 of Table 5.6 and Table 5.7 show that the NCVW cost of MPMCT

Table 5.6: NCVW cost of MPMCT gates of size $n = [1 \dots 16]$ with 1 ancillary line.

| | **Number of Negative Controls** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **c** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| 0 | 1 | | | | | | | | | | | | | | | |
| 1 | 1 | 3 | | | | | | | | | | | | | | |
| 2 | 5 | 5 | 6 | | | | | | | | | | | | | |
| 3 | 13 | 13 | 13 | 14 | | | | | | | | | | | | |
| 4 | 20 | 20 | 20 | 22 | 24 | | | | | | | | | | | |
| 5 | 28 | 28 | 28 | 30 | 32 | 34 | | | | | | | | | | |
| 6 | 40 | 40 | 40 | 40 | 42 | 44 | 46 | | | | | | | | | |
| 7 | 52 | 52 | 52 | 52 | 54 | 56 | 58 | 60 | | | | | | | | |
| 8 | 64 | 64 | 64 | 64 | 64 | 66 | 68 | 70 | 72 | | | | | | | |
| 9 | 80 | 80 | 80 | 80 | 80 | 82 | 84 | 86 | 88 | 90 | | | | | | |
| 10 | 96 | 96 | 96 | 96 | 96 | 96 | 98 | 100 | 102 | 104 | 106 | | | | | |
| 11 | 112 | 112 | 112 | 112 | 112 | 112 | 114 | 116 | 118 | 120 | 122 | 124 | | | | |
| 12 | 128 | 128 | 128 | 128 | 128 | 128 | 130 | 132 | 134 | 136 | 138 | 140 | 142 | | | |
| 13 | 152 | 152 | 152 | 152 | 152 | 152 | 152 | 154 | 156 | 158 | 160 | 162 | 164 | 166 | | |
| 14 | 176 | 176 | 176 | 176 | 176 | 176 | 176 | 176 | 178 | 180 | 182 | 184 | 186 | 188 | 190 | |
| 15 | 200 | 200 | 200 | 200 | 200 | 200 | 200 | 200 | 202 | 204 | 206 | 208 | 210 | 212 | 214 | 216 |

Table 5.7: NCVW cost of MPMCT gates of size $n = [1 \dots 16]$ with $n - 3$ ancillary lines.

| | **Number of Negative Controls** | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **c** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| 0 | 1 | | | | | | | | | | | | | | | |
| 1 | 1 | 3 | | | | | | | | | | | | | | |
| 2 | 5 | 5 | 6 | | | | | | | | | | | | | |
| 3 | 13 | 13 | 13 | 14 | | | | | | | | | | | | |
| 4 | 20 | 20 | 20 | 22 | 24 | | | | | | | | | | | |
| 5 | 28 | 28 | 28 | 30 | 32 | 34 | | | | | | | | | | |
| 6 | 40 | 40 | 40 | 40 | 42 | 44 | 46 | | | | | | | | | |
| 7 | 52 | 52 | 52 | 52 | 54 | 56 | 58 | 60 | | | | | | | | |
| 8 | 64 | 64 | 64 | 64 | 64 | 66 | 68 | 70 | 72 | | | | | | | |
| 9 | 76 | 76 | 76 | 76 | 76 | 78 | 80 | 82 | 84 | 86 | | | | | | |
| 10 | 88 | 88 | 88 | 88 | 88 | 88 | 90 | 92 | 94 | 96 | 98 | | | | | |
| 11 | 100 | 100 | 100 | 100 | 100 | 100 | 102 | 104 | 106 | 108 | 110 | 112 | | | | |
| 12 | 112 | 112 | 112 | 112 | 112 | 112 | 112 | 114 | 116 | 118 | 120 | 122 | 124 | | | |
| 13 | 124 | 124 | 124 | 124 | 124 | 124 | 124 | 126 | 128 | 130 | 132 | 134 | 136 | 138 | | |
| 14 | 136 | 136 | 136 | 136 | 136 | 136 | 136 | 136 | 138 | 140 | 142 | 144 | 146 | 148 | 150 | |
| 15 | 148 | 148 | 148 | 148 | 148 | 148 | 148 | 148 | 150 | 152 | 154 | 156 | 158 | 160 | 162 | 164 |

gates with $m$ negative controls $m = \{0, \dots, c-3\}$ stays the same as the cost of MPMCT gates with all positive controls until $m = \lfloor c/2 \rfloor$ and is incremented by 2 for each additional negative control after that.

The results show that for one available ancillary, using the NCVW library improves the NCV costs of [84] by 39.57% on average. Since our method utilizes all available ancillaries, it yields lower costs when the number of ancillaries is more than 1 and less than $n-3$ comparing to [84] that uses the same cost as for one available ancillary. As a result, for MCT gates with ancillaries in the range $\{2, \dots, n-4\}$ our method shows even more improvement. For example, if the number of ancillaries is in the range $\{\lfloor c/2 \rfloor - 2, \dots, n-4\}$, the cost of MCT gates would be the same as when there are

$n-3$ ancillaries available, which gives 46.54% improvement over [84]. The NCVW method improves the costs in [84] by 8.31% on average for the case where $n-3$ ancillaries are available.

The results of Table 5.6 show that the NCVW library can improve the NCV costs reported in Section 5.1.3 by 13.54% when there is one available ancillary. Also as Table 5.7 illustrates, the costs of MPMCT gates with $n-3$ ancillaries are improved by 4.63% using the NCVW library in comparison with the results in Section 5.1.3.

## 5.3 NCV-$|v_1\rangle$ Library

In the previous section, we discussed a new quantum gate library, NCVW, that could improve the quantum costs of a class of reversible gates called MPMCT gates. Continuing our search to find other libraries that can lead to better circuits, *e.g.* fewer gates, we consider a modified NCV library motivated by the approach introduced in [114]. We demonstrate that the new library leads to realizations for multiple-control Toffoli gates with far fewer quantum gates than have been found using the previous approaches. Our approach uses a structure similar to one recently introduced in [115]. However, while that work addressed a particular application, we here consider the realization of general reversible circuits.

If circuits with Boolean inputs use NCV gates only, the value of each qubit at each stage of the circuit is restricted to one of $\{0, v_0, 1, v_1\}$ where

$$v_0 = \frac{1}{2}\begin{pmatrix} 1+i \\ 1-i \end{pmatrix} \tag{5.14}$$

$$v_1 = \frac{1}{2}\begin{pmatrix} 1-i \\ 1+i \end{pmatrix} \tag{5.15}$$

The $NOT$, $V$, and $V^+$ operations over these four values are:

| $x$ | $NOT(x)$ | $V(x)$ | $V^+(x)$ |
|---|---|---|---|
| 0 | 1 | $v_0$ | $v_1$ |
| $v_0$ | $v_1$ | 1 | 0 |
| 1 | 0 | $v_1$ | $v_0$ |
| $v_1$ | $v_0$ | 0 | 1 |

As shown, $NOT$ is a complement operation, $V$ is the cycle $(0 \rightarrow v_0 \rightarrow 1 \rightarrow v_1 \rightarrow 0)$, and $V^+$ is the inverse cycle.

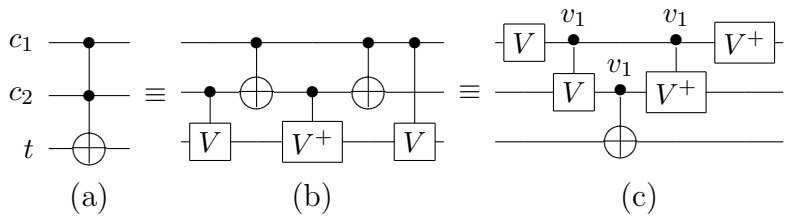



Figure 5.20: Mapping (a) a Toffoli gate to (b) NCV gates and (c) NCV-$|v_1\rangle$ gates.

In this work, we adopt a new quantum gate library which we call the *NCV-$|v_1\rangle$ library.*

**Definition 5.4.** *The* **NCV-$|v_1\rangle$ gate library** *is composed of (1) the three single-qubit gates (*i.e. *gates without a control line) performing the* NOT, V, *and* $V^+$ *operation as well as (2) single-control versions of these gates. In contrast to the NCV-library, and in keeping with the work in [114], the controlled gates perform the respective operation not when the control line is 1, but rather when the control line is set to the value $v_1$. We label control connections for NCV-$|v_1\rangle$ gates with $v_1$ to emphasize this fact.*

Besides the benefits in the physical implementation, as discussed in [114], this gate library enables a much more efficient mapping from an MCT gate circuit as we show below. The implementation cost of a quantum gate is heavily technology dependent. Here we assume that all the quantum gates in the NCV-$|v_1\rangle$ library have unit cost. Under this assumption, the cost of a quantum circuit is the gate count. This is clearly an approximation but a suitable one when considering technology independent optimization.

### 5.3.1 Proposed Mapping Method

The gates in the NCV-$|v_1\rangle$ library are controlled by a quantum value, $v_1$. Allowing the gates to be controlled by a value other than the data values, *i.e.* 0 or 1, is beneficial in the sense that it allows more compact structures. We apply the constraint of using only one controlling value in keeping with the work in [114]. However, allowing more than one controlling value *e.g.* 0, 1, and $v_1$ could lead to much better circuits. For example, a CNOT gate is a primitive gate in other libraries while it has cost 3 in the NCV-$|v_1\rangle$ library because it is controlled by 1 and has to be mapped to a CNOT that is controlled by $v_1$.

**Example 5.6.** *Figure 5.20 shows (a) a Toffoli gate, (b) its NCV realization, and (c) the new NCV-$|v_1\rangle$ realization.*

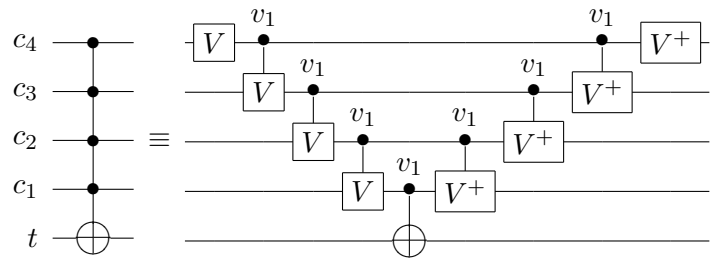



Figure 5.21: Mapping of a 4-control MCT gate to NCV-$|v_1\rangle$ gates.

Generalizing the structure in Figure 5.20 (c), every MCT gate with $c$ control lines can be realized using a $V$ gate and a $V^+$ gate for each control line as well as a single $CNOT$ gate which operates on the target line. This leads to a total of $2c + 1$ NCV-$|v_1\rangle$ gates for an MCT gate with $c$ controls.

**Example 5.7.** *The structure illustrated in Figure 5.21 shows the NCV-$|v_1\rangle$ mapping for a 4-control MCT gate.*

In the structure of Figure 5.21, the actual operation of the MCT gate (*i.e.* the inversion of the target line) is performed by a single CNOT gate controlled by the $v_1$ value. The $V$ gates ensure that the control line of the CNOT gate is set to $v_1$ if, and only if, all control lines $c_1$ to $c_4$ are equal to 1. The last $V^+$ gates are needed to undo the corresponding $V$ operations on the control lines in order to restore their values.

While for an MCT gate with 1 control line only, this results to a more expensive mapping (3 gates instead of 1), MCT gates with more than 2 controls can be realized with significant reductions in the number of gates compared to the established NCV mappings. Furthermore, the NCV-$|v_1\rangle$ mappings do not require any ancillary lines.

Realization of an MCT gate with no ancillary lines was investigated by Barenco *et al.* [5]. In that work, gates realizing higher-order roots-of-NOT are required and the number of gates required for an $n$-line MCT gates is $2^n - 3$. The complexity grows exponentially as opposed to the slow linear growth for the NCV-$|v_1\rangle$ realizations.

### 5.3.2 Negative Controls

A number of reversible circuit synthesis algorithms, *e.g.* [100, 116], produce circuits with MPMCT gates that have *negative* controls. Realization of mixed-polarity MCT gates in the NCV and NCVW libraries was studied in the previous sections. It was shown that negative controls often add extra costs to the circuit. To give an idea of the incremental cost, the NCV realization of a 3-controlled MPMCT gate has 14

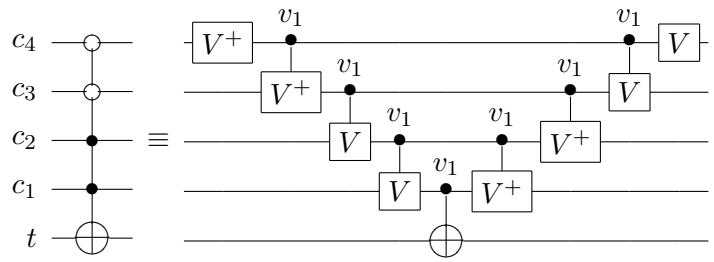



Figure 5.22: Mapping of a 4-control MPMCT gate to NCV-$|v_1\rangle$ gates.

gates for 0 or 1 negative controls, 16 for 2 negative controls, and 18 for 3 negative controls. For a 15-control MPMCT gate, the NCV cost ranges from 228 to 246 by changing the polarity of the controls if there is one ancillary line available. For the same gate if 13 ancillary lines are available, the NCV cost ranges from 152 to 168 by changing the polarity of the controls. See tables 5.3, 5.4, 5.6 and 5.7 for more details.

The situation is quite different for our new NCV-$|v_1\rangle$ gate realizations. Consider the structure illustrated in Figure 5.21. To change a positive control to a negative control, the $V$ gate on that line simply has to be swapped with the corresponding $V^+$ gate. Hence, our mapping always leads to $2c+1$ gates for $c$ controls regardless of the mix of positive and negative controls. Thus, in contrast to the NCV situation, the NCV-$|v_1\rangle$ gate circuits handle negative controls for free. And once again, no ancillary lines are required.

**Example 5.8.** *The structure illustrated in Figure 5.22 shows the NCV-$|v_1\rangle$ mapping for an MPMCT gate with 2 positive and two negative controls. Note that the $V^+$ gates on $c_4$ and $c_3$ lines take 0 to $v_1$, and the $V$ gates on $c_2$ and $c_0$ take 1 to $v_1$. Hence, if $c_4$ and $c_3$ are set to zero and $c_2$ and $c_1$ are set to 1, the target is inverted.*

### 5.3.3 Nearest-Neighbor Constraints

Some quantum technologies require that a circuit satisfies the *nearest-neighbor constraint* [117].

**Property 5.1.** *A circuit has* **nearest-neighbor property** *if the qubits in gates with more than one qubit are all adjacent. The constraint of having this property is called the* **nearest-neighbor constraint**.

The nearest neighbor constraint has been considered in the literature recently *e.g.* the 2-dimensional nearest-neighbor quantum architecture introduced in [118] for

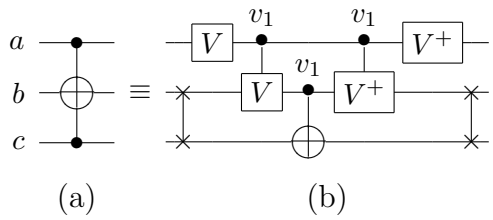



Figure 5.23: (a) MCT gate $T(\{a, c\}; b)$ and (b) NCV-$|v_1\rangle$ gate nearest neighbor circuit.

implementing Shor's factoring algorithm [24]. A circuit can be modified to satisfy this constraint by adding gates which swap the values of lines so that only adjacent lines are used. How best to locate the swap gates has been studied *e.g.* in [47, 119].

**Example 5.9.** *The NCV circuit in Figure 5.20(b) does not satisfy the nearest-neighbor constraint and can not be made nearest-neighbor by permuting the lines. The circuit in Figure 5.20(c) however has nearest-neighbor property as the controls and targets are adjacent in all the gates .*

In the general case, the existing mappings of MPMCT gates into NCV and NCVW quantum gates require additional swap gates to satisfy the nearest-neighbor constraint. Often, a significant number are required.

In contrast, no intervening lines exist in the proposed structure depicted in Figure 5.20(c) and Figure 5.21, *i.e.* the nearest-neighbor condition is already satisfied. Moreover, this mapping remains nearest-neighbor if the target is the top line in which case the structure is inverted. If, however, an MPMCT gate is considered where the target is an inside line, or there are intervening unconnected lines, the structure is no longer nearest-neighbor and swap gates (depicted as two $\times$'s joined by a line) are required as shown in Figure 5.23.

As just justified, in many cases the proposed mapping can be directly applied to technologies requiring nearest-neighbor constraints. Developing good heuristics for handling the remaining cases is left for future work. Approaches *e.g.* introduced in [47, 119] may be exploited for this purpose.

### 5.3.4 Experimental Results

Table 5.8 shows the number of NCV gates required to realize MCT gates with up to 15 controls using the latest NCV mapping approach described in Section 5.1.2. For each number of controls, the rightmost gate count is the lowest possible. Blank entries

Table 5.8: Quantum gate counts for MCT gate realizations

| | NCV gates [106] Number of ancillary lines | | | | | | | NCV-$\|v_1\rangle$ gates | Ratio |
|---|---|---|---|---|---|---|---|---|---|
| $c$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | $2c+1$ | |
| 0 | 1 | | | | | | | 1 | 100% |
| 1 | 1 | | | | | | | 3 | 300% |
| 2 | 5 | | | | | | | 5 | 100% |
| 3 | | 14 | | | | | | 7 | 50.0% |
| 4 | | 20 | | | | | | 9 | 45.0% |
| 5 | | 32 | | | | | | 11 | 34.4% |
| 6 | | 44 | | | | | | 13 | 29.5% |
| 7 | | 64 | 56 | | | | | 15 | 23.4-26.8% |
| 8 | | 76 | 68 | | | | | 17 | 22.4-25.5% |
| 9 | | 96 | 88 | 80 | | | | 19 | 19.8-23.8% |
| 10 | | 108 | 100 | 92 | | | | 21 | 19.4-22.8% |
| 11 | | 132 | 120 | 112 | 104 | | | 23 | 17.4-22.1% |
| 12 | | 156 | 132 | 124 | 116 | | | 25 | 16.0-21.6% |
| 13 | | 180 | 156 | 148 | 136 | 128 | | 27 | 15.0-21.1% |
| 14 | | 204 | 180 | 172 | 148 | 140 | | 29 | 14.2-20.7% |
| 15 | | 228 | 204 | 198 | 172 | 160 | 152 | 31 | 13.6-20.4% |

indicate when the availability of more ancillary lines is not advantageous. The NCV gate counts given in Table 5.8 are the best known for NCV gate realizations of MCT gates. It is important to note (1) that at least one ancillary line is required for three or more controls and (2) that the gate counts grow quite quickly with the number of controls. In Table 5.8 for $c \geq 4$, assuming the maximum number of ancillaries required is available, the number of gates is $12c - 28$. If only one ancillary is available, for $c \geq 10$ the number of gates required is $24c - 132$. These formulas have been verified for up to 20 controls and it is believed they will continue to hold for larger numbers of controls.

The eighth column in Table 5.8 lists the number of NCV-$|v_1\rangle$ gates required to realize the corresponding MCT gate for up to 15 controls. The gate counts are significantly lower than the NCV costs. The ninth column shows the ratio of the size of NCV-$|v_1\rangle$ circuits to NCV circuits. The relative size of the NCV-$|v_1\rangle$ circuits to the NCV circuits drops to about 30% at 6 controls and approaches 20% at 15 controls. We again emphasize that the NCV-$|v_1\rangle$ gate circuits require no ancillary lines.

### Verification

Verification methods have been applied to confirm that the circuits produced by our methods are functionally equivalent to the original MCT gates. This is an interesting

Table 5.9: Quantum gate counts for an MCT gate with $c$ controls.

| **No. of ancillaries** | **NCV [84]** | **Our NCV** | **NCVW** | **NCV-$\|v_1\rangle$** |
|---|---|---|---|---|
| 1 | $24c-64$ | $24c-132$ | $24c-160$ | $2c+1$ |
| $c-2$ | $12c-22$ | $12c-28$ | $12c-32$ | $2c+1$ |

problem on its own. The original MCT gates are binary whereas the circuits our method produces use 4-valued logic gates. Also our basic MCT gate substitution relies on the fact that the inputs are restricted to values 0 and 1.

Our verification procedure uses *Quantum Multiple-valued Decision Diagrams* (QMDD) [72] and is basically the approach described in [95]. The differences are that the original MCT gates are treated as 4-valued as are of course the gates in the circuit we produce, and determining equivalence of the circuits is not just a matter of confirming the equality of the QMDD for the two circuits. Rather, equivalence checking requires a depth-first comparison of the two QMDD that restricts the input line values to 0 and 1, ignoring $v_0$ and $v_1$.

## 5.4 Summary

Quantum gate libraries are used in technology mapping and implementation of reversible gates, and for quantum cost calculation. The most common quantum gate library is the NCV library. In this chapter, we first reviewed the existing cost functions for mapping reversible gates such as MPMCT gates to this library. We further discussed possible improvements on the existing mapping approaches targeting the NCV library. Moreover, we explored two new quantum gate libraries, NCVW and NCV-$|v_1\rangle$, and proved the potential to improve the cost functions by using better quantum gate libraries and mapping approaches.

Table 5.9 summarizes the quantum costs (in terms of the number of quantum gates) of MCT gates with $c$ controls resulting from the existing and the proposed approaches. Besides having benefits like better nearest neighbor properties and no extra cost for negative controls, the NCV-$|v_1\rangle$ library significantly outperforms the other libraries in terms of the growth rate of the costs.

# Chapter 6

# Mapping Reversible Circuits to Quantum Circuits

Quite often technology mapping reversible to quantum circuits is performed simply by replacing each reversible gate with an elementary quantum gate realization with no attempt made to optimize across reversible gate boundaries. The number of gates in the resulting circuit is typically reported as the quantum cost of the reversible circuit, *e.g.* the costs reported on the reversible circuit benchmark sites: Maslovs [120] and REVLIB [9].

The quantum cost of the circuits resulting from technology mapping is usually considered as a defining factor in comparing efficiency of synthesis methods and in considering implementation trade-offs. In this chapter, we show that the mapping procedure itself has an important effect on the cost of the resulting circuit. We further discuss new optimization flows for mapping reversible cascades to quantum circuits. We show the methods targeting the three quantum gate libraries, NCV, NCVW, and NCV-$|v_1\rangle$. However, the proposed methods can be applied for other libraries.

## 6.1 Direct Mapping

Direct mapping of reversible to quantum circuits is done by substituting a quantum realization for each reversible gate in the target circuit. Thus, the quantum cost of a reversible circuit $G_0G_1 \dots G_n$ is obtained by:

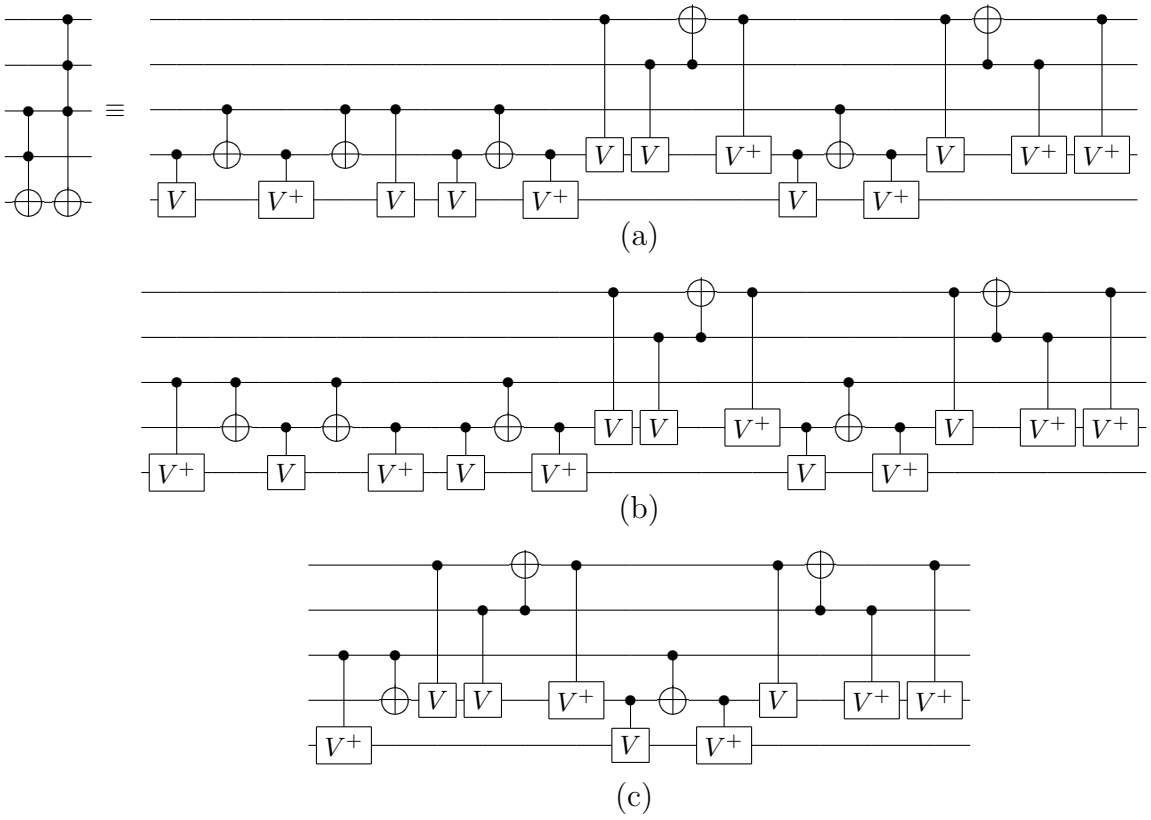



Figure 6.1: A reversible circuit. (a) Direct mapping with cost 19. (b) An alternate direct mapping. (c) Optimized mapping with cost 13.

$$C = \sum_{i=0}^{n} c_i \tag{6.1}$$

where $c_i$ is the quantum cost (quantum gate count) of reversible gate $G_i$. In the following example, we show that when we are mapping more than one reversible gate, the actual cost may be less than this sum.

**Example 6.1.** *Consider the reversible circuit in Figure 6.1 (a) consisting of a 2-control MCT gate followed by a 3-control MCT gate. The direct mapping of this circuit is obtained by substituting each gate with its realization from Figures 5.14 (a) and (c) as shown in the righthand side of Figure 6.1 (a). The cost of the resulting circuit is* $5 + 14 = 19$. *If the inverse realization (Property 2.3) of the 2-control MCT gate is used for mapping, the circuit in 6.1 (b) results which can be optimized by removing six gates (the third gate to the eighth gate canceled in the order 5&6, 4&7, and 3&8). Figure 6.1 (c) shows the result of using an optimized mapping with cost*

*13, less than the cost of a single 3-control MCT gate.*

The example above demonstrates how using an alternative mapping can lead to less expensive circuits. It also shows that sometimes more reversible gates yields less quantum cost if an optimization method is used and this fact is not obvious at the reversible gate level.

## 6.2 Optimized Mapping of MPMCT Circuits

The following procedure gives the proposed optimized flow for mapping MPMCT circuits to quantum circuits.

**Procedure 6.1. MPMCT to Quantum Circuit Mapping**

1. The initial reversible circuit is read.
2. The gate reduction procedure (Procedure 3.1) is used to reduce the quantum cost of the MPMCT circuit as described in Section 3.3.2.
3. Using the conventional moving rule (Property 3.1), certain pairs of MPMCT gates that are, or can be made, adjacent are examined to find a pair whose combined quantum cost when implemented as a single unit is lower than the cost of realizing the gates separately. Among all possible combinations, the pair with the minimum cost is selected and replaced by the corresponding quantum realization. If there are more than one pair with the minimum cost, the first one found will be mapped. The procedure involves a single pass through the circuit from the inputs toward the outputs. The approach is greedy and does not consider all possible reductions.
4. The quantum circuit resulting from the previous step is optimized using the gate reduction procedure (Procedure 3.1) and the method described in Section 3.3.3.

The details of each step of this procedure are as follows:

**Step 1:** As the circuit is read, equivalent MPMCT gate sequences are substituted for any Fredkin, Peres and inverse Peres gates. It should be noted that this does not lose the reduced cost of Peres and inverse Peres gates which is found in step 4. Similar substitutions can also be made for other reversible gates.

**Step 2:** In this step, the circuit is examined to find MPMCT gates that cancel each other. In addition, further reductions are applied as mentioned in section 3.3.2.

**Step 3:** The third step does the initial mapping from MPMCT gates to quantum gates. Each gate $G$ is considered in order. The procedure searches for a gate $H$ later in the circuit such that $G$ can be made adjacent to $H$ using the conventional moving rule (Property 3.1). This approach moves gates in both directions to find all possible combinations. More precisely, the gate $G$ is moved forward until the end of the circuit or once a blocking gate is found to find a pair whose combined quantum cost when implemented as a single unit is lower than the cost of realizing the gates separately. Then, all gates after the blocking gate are moved backward to make them adjacent to $G$. When considering such a pair of gates, the appropriate quantum implementations from the desired catalog of model circuits (NCV, NCVW, etc.) are used for the two gates. Inverse realization (Property 2.3), reverse realization (Property 2.4), inverse of the reverse realization, and variable reordering for controls of 2-control MPMCT gates are exhaustively applied to best reduce the cost of jointly implementing the pair.

The line order for mapping each gate in the pair is the target, followed by the ancillary lines available for both gates, followed by the ancillaries available only to the gate being mapped, followed by the control lines belonging to both gates, followed by the control lines used only by the gate being mapped. Experiments have shown this order leads to a high degree of quantum gate cost reduction. For a given $G$, every possible $H$ is considered, and the pair with the lowest quantum cost implementation is chosen. Among all the pairs with the lowest quantum cost, the first one found will be selected. Gate $H$ is replaced by the quantum gate sequence and gate $G$ is removed. The search for further reductions proceeds from the gate that followed $G$.

Note that the conventional moving rule (Property 3.1) is used at this step rather than our generalized moving rule (Property 3.2). Experiments have shown that using the new moving rule at this step does not give a significant cost improvement while it increases the running time of the program considerably. The reason is that most of the reductions that are not found at this step using the conventional rule will be found at the next step using our new moving rule.

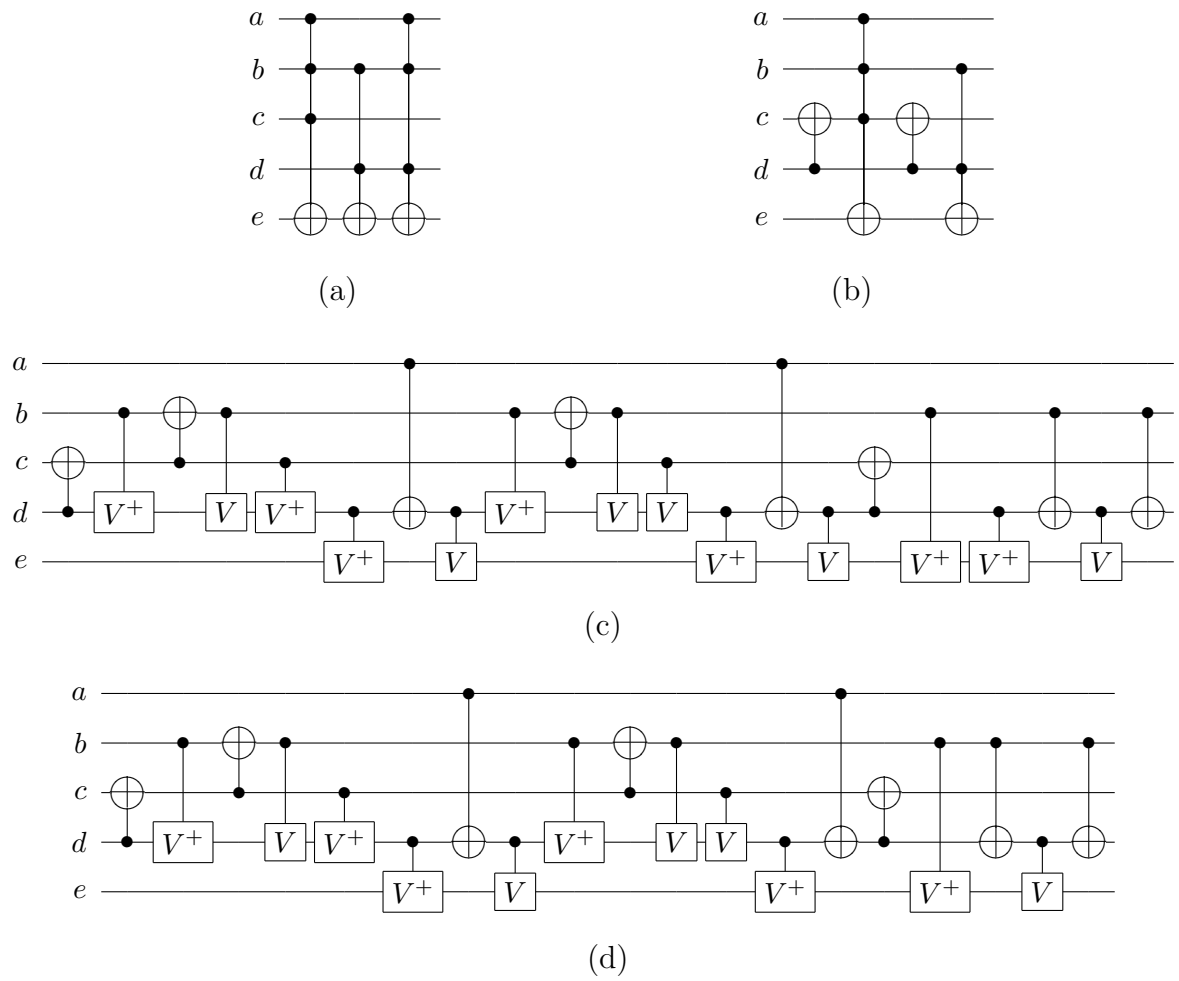



Figure 6.2: Mapping an MPMCT cascade to its NCV realization

**Step 4:** The reduction rules of section 3.2 are applied to the quantum circuit found in the previous step. In this step, reductions are found that are not identified in step 3 which only considers the MPMCT gates in pairs.

**Example 6.2.** *Consider the circuit in Figure 6.2 (a). The rightmost gate can be combined with the second gate or the first gate. Our procedure combines the third and first gates as that leads to the greater cost reduction. The resulting circuit is shown in Figure 6.2 (b). Quantum gate expansion (step 3 of our procedure) yields the circuit in Figure 6.2 (c) which has 21 gates. Finally, step 4 uses the gate reduction procedure (Procedure 3.1) to eliminate two gates (the fourth and seventh from the right in Figure 6.2 (c)) giving the cost 19 circuit shown in Figure 6.2 (d). This is much better than the cost of 33 which would be found by substituting separate quantum implementations*

*for the three MPMCT gates in the initial circuit.*

Since all MPMCT gates are self-inverse (Property 2.2), it follows from Property 2.4 that the reverse of an MPMCT circuit realizes the inverse of the reversible function realized by the original circuit. Hence if we use our procedures to map the reverse circuit to a quantum realization, reversing that realization yields a quantum circuit for the function realized by the original circuit. This means we can apply our techniques to an MPMCT circuit and its reverse and choose the lower cost of the two resulting quantum circuits. We term this the *bidirectional* approach.

The above bidirectional approach can be used to map MPMCT circuits to quantum circuits consisting of any order root-of-NOT gates such as NCV, NCVW, etc.

Due to the regular nature of the NCV-$|v_1\rangle$ cost model (see Section 5.3), many of the quantum gates that are used to change and restore the values on the controls can be canceled in MPMCT gate boundaries. An important factor in identifying these cancelations is the starting point and order in which the MPMCT gate controls are mapped. choosing a wise order in mapping MPMCT circuit may lead to even more reductions. The following procedure shows a heuristic method that considers a different order in mapping controls of MPMCT gates hoping to achieve more reduction after optimization at the quantum level. In this procedure, instead of a fixed order, the controls of an MPMCT gate are mapped starting from the ones that are in common with its neighbors.

**Procedure 6.2. Alternative MPMCT to NCV-$|v_1\rangle$ Mapping**
*For all $G_i$ in a given MCT circuit $G_0 \dots G_n$:*

1. *Consider $G_k$ as the next gate in the circuit ($k = i + 1$).*
2. *Let $CC = \{$controls of $G_i\} \cap \{$controls of $G_k\}$ be a set of common controls.*
3. *While $CC \neq \emptyset$:*
   - $k = k + 1$.
   - $CC = CC \cap \{$controls of $G_k\}$.
4. *Map MPMCT gates $G_i \dots G_{k-1}$ to their NCV-$|v_1\rangle$ realizations so that the order of the controls that are mapped for each gate starts from the controls in the $CC$ set.*
5. *Set $i = k + 1$.*

The above procedure finds a set of common controls among as many consecutive MPMCT gates as possible and for those gates starts the mapping from the controls in common. Since this approach considers local constraints to determine the mapping structure comparing to the direct mapping that uses a fixed order among all gates in a circuit, while it will be helpful to introduce more reductions in some circuits, it may deteriorate the results of mapping for some circuits.

## 6.3 Experimental Results

The optimized mapping flows described in the previous section were examined for circuits targeting three quantum gate libraries, NCV, NCVW, and NCV-$|v_1\rangle$. The results are reported in the following subsections.

### 6.3.1 MPMCT to NCV Circuit Mapping

A prototype implementation of the described mapping procedure (Procedure 6.1) for MPMCT to NCV circuit mapping has been implemented using Python 2.6.5. We have applied the bidirectional approach to a benchmark test suite containing 150 circuits from the REVLIB web site [9]. The only circuits not considered were the large ones for the unstructured reversible functions urf1 - urf6. Our prototype program uses REVLIB [9] *real* circuit format. The experiments were run on a system with a 3.2 GHz i5-650 CPU and 3.0 GB RAM.

The results are summarized in Table 6.1. The table includes only the 74 circuits for which our approach resulted in a quantum gate cost reduction of 15% or more. The table gives the REVLIB circuit name and file identification number and the quantum cost given in REVLIB. The column labeled *Initial Cost* is the cost of the REVLIB circuit using the costs from Table 5.2 substituting for each MPMCT gate with no optimization across gate boundaries. They differ for three reasons. First since we only consider quantum gates NOT, CNOT, $V$ and $V^+$, the cost of a 3-control Toffoli gate is 14 [71] and not 13 as used in REVLIB. Second, the procedure presented here for finding NCV realizations for individual MPMCT gates uses the actual number of ancillary lines available, not just the two extremes. Third, that procedure produces lower cost NCV circuits than do earlier methods which are the basis for the cost model used in REVLIB. For most circuits, our initial cost is lower than the REVLIB cost. However, this is not always the case.

As noted in Chapter 5, the techniques described here can not be applied to an $n$-line MPMCT circuit which contains a gate using all $n$ lines ($n-1$ controls and a target). This is because at least one ancillary line is required to map an MPMCT gate to NCV gates [71]. For such circuits, we add an ancillary line. This is identified by a * after the name. This is not the case for the results reported in REVLIB. The cost model employed in REVLIB is based on the work in Barenco *et al.* [5] which assumes a $2^{c-1}$-th root-of-NOT gate is available to realize a $c$-control MPMCT gate in a circuit with $c+1$ lines. We anticipate that realizing gates progressively higher roots of NOT may be prohibitive in many technologies and adding one extra line will be preferable. Also we expect that synthesis methods can be made to avoid the situation in most cases.

The column *MCT Reduction* gives the quantum cost after applying Step 2 (MPMCT gate reduction) in our mapping procedure (Procedure 6.1). The column *Quantum Expansion* gives the quantum cost after Step 3 in which MPMCT gates are expanded to their quantum equivalent and optimized in pairs. The column *Quantum Reduction* is the final quantum cost (number of gates) for the circuit after applying Step 4 to remove redundant quantum gates. The *% Cost Reduction* compares the cost of each final circuit to the cost given in REVLIB. The CPU times given were determined using the Python time function. They show the relative performance of the program for various circuits but should not be given too much weight as our implementation has not been optimized and uses many heuristics that could be further developed.

As noted, we used the bidirectional approach in our experiments. The results presented for each circuit are for the better (least cost) of treating the MPMCT circuit as given and in reverse order.

For the circuits listed in Table 6.1, the cumulative cost reduction is 42.21% and the average improvement is 30.32%. The cumulative cost reduction for all the circuits in the test suite is 42.02% and the average improvement is 16.74%. As expected, the results in Table 6.1 show that the cost reduction resulting from applying our mapping procedure is very circuit dependent.

As mentioned above, a possible variant of the MPMCT to quantum circuit mapping procedure (Procedure 6.1) is to use our new moving rule (Property 3.2) at Step 3 of the algorithm to find the pairs that can be combined with lower quantum cost. Surprisingly, applying property 3.2 leads to a cumulative cost reduction of 42.27% for the circuits in Table 6.1 which is close to the result of using the conventional moving rule. But, as also mentioned above, using the new moving rule increases the time

complexity significantly. Our experiments thus confirm that using the new moving rule in this way offers no improvement over using the conventional moving rule.

The same benchmarks have been run with the added optimization of factoring the MPMCT gates with a single added helper line as described in [85]. The results are reported in Table 6.2 for the circuits listed in Table 6.1. Note that an ancillary line has been added to every circuit in this experiment. The cumulative cost reduction for the circuits in Table 6.2 is 53.83% compared to the costs given in REVLIB and on average the improvement is 37.33%.

Table 6.1: Results for optimized mapping MPMCT circuits to 74 NCV circuits.

| REVLIB Circuit | REVLIB Cost | Initial Cost | MCT Reduction | Quantum Expansion | Quantum Reduction | % Cost Reduction | CPU (sec.) |
|---|---|---|---|---|---|---|---|
| 3_17_13 | 14 | 14 | 14 | 11 | 11 | 21.43 | 0.047 |
| 4gt12-v0_86* | 58 | 50 | 38 | 36 | 36 | 37.93 | 0.093 |
| 4gt12-v0_87* | 54 | 46 | 34 | 34 | 34 | 37.04 | 0.063 |
| 4gt12-v0_88* | 41 | 32 | 32 | 30 | 30 | 26.83 | 0.063 |
| 4gt12-v1_89* | 45 | 37 | 37 | 37 | 37 | 17.78 | 0.047 |
| 4gt4-v0_72* | 54 | 46 | 34 | 34 | 34 | 37.04 | 0.093 |
| 4gt4-v0_73* | 89 | 83 | 49 | 49 | 49 | 44.94 | 0.156 |
| 4gt4-v0_78* | 53 | 45 | 45 | 45 | 45 | 15.09 | 0.093 |
| 4gt4-v0_79* | 49 | 41 | 41 | 41 | 41 | 16.33 | 0.047 |
| 4gt4-v0_80* | 37 | 28 | 28 | 28 | 28 | 24.32 | 0.047 |
| 4gt4-v1_74* | 57 | 49 | 49 | 46 | 46 | 19.3 | 0.047 |
| 4mod5-v0_18 | 25 | 25 | 19 | 17 | 17 | 32 | 0.063 |
| 4mod5-v0_19 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.016 |
| 4mod5-v0_20 | 9 | 9 | 9 | 7 | 7 | 22.22 | 0.016 |
| 4mod5-v1_22 | 9 | 9 | 9 | 7 | 6 | 33.33 | 0.031 |
| 4mod5-v1_23 | 24 | 24 | 18 | 16 | 16 | 33.33 | 0.047 |
| 4mod7-v0_94 | 38 | 40 | 40 | 34 | 32 | 15.79 | 0.047 |
| 4mod7-v0_95 | 38 | 40 | 40 | 34 | 32 | 15.79 | 0.047 |
| 4mod7-v1_96 | 39 | 41 | 41 | 35 | 33 | 15.38 | 0.063 |
| alu-v2_31 | 101 | 107 | 89 | 83 | 83 | 17.82 | 0.141 |
| cycle10_2_110 | 1202 | 764 | 764 | 720 | 720 | 40.1 | 1.266 |
| decod24-v0_38 | 18 | 18 | 18 | 14 | 14 | 22.22 | 0.016 |
| decod24-v2_43 | 18 | 18 | 18 | 14 | 14 | 22.22 | 0.031 |
| fredkin_6 | 15 | 15 | 15 | 13 | 12 | 20 | 0.016 |
| ham15_107 | 1831 | 1598 | 1250 | 1161 | 1155 | 36.92 | 4.781 |
| ham15_108 | 453 | 392 | 389 | 358 | 356 | 21.41 | 0.891 |
| ham3_102 | 9 | 9 | 9 | 7 | 7 | 22.22 | 0.016 |
| ham3_103 | 10 | 10 | 10 | 8 | 8 | 20 | 0.015 |
| hwb4_49* | 65 | 67 | 59 | 55 | 54 | 16.92 | 0.14 |
| hwb6_56* | 1530 | 1280 | 1256 | 1159 | 1150 | 24.84 | 5.843 |
| hwb6_57* | 1171 | 951 | 881 | 874 | 872 | 25.53 | 4.547 |
| hwb7_59* | 5236 | 3983 | 3797 | 3515 | 3500 | 33.16 | 8.922 |
| hwb7_60* | 4170 | 3130 | 3006 | 2999 | 2989 | 28.32 | 9.437 |
| hwb7_61* | 3876 | 3135 | 3063 | 2869 | 2863 | 26.14 | 12.39 |

Continued on the next page.

Continued from the previous page.

| REVLIB Circuit | REVLIB Cost | Initial Cost | MCT Reduction | Quantum Expansion | Quantum Reduction | % Cost Reduction | CPU (sec.) |
|---|---|---|---|---|---|---|---|
| hwb7_62* | 2611 | 1995 | 1995 | 1978 | 1973 | 24.44 | 6.282 |
| hwb8_113* | 16530 | 11522 | 11042 | 10346 | 10328 | 37.52 | 25.734 |
| hwb8_114* | 14699 | 9840 | 9576 | 8823 | 8815 | 40.03 | 43.954 |
| hwb8_115* | 14691 | 9840 | 9576 | 8823 | 8815 | 40 | 44.594 |
| hwb8_116* | 7015 | 4867 | 4867 | 4832 | 4825 | 31.22 | 17.703 |
| hwb8_117* | 7013 | 4867 | 4867 | 4832 | 4825 | 31.2 | 18.062 |
| hwb8_118* | 16522 | 11522 | 11042 | 10346 | 10328 | 37.49 | 25.328 |
| hwb9_119* | 44714 | 32349 | 31164 | 28719 | 28660 | 35.9 | 85.922 |
| hwb9_120* | 44702 | 32349 | 31164 | 28719 | 28660 | 35.89 | 85.671 |
| hwb9_121* | 44665 | 32312 | 31136 | 28689 | 28629 | 35.9 | 89.515 |
| hwb9_122* | 44653 | 32312 | 31136 | 28689 | 28629 | 35.89 | 85.891 |
| hwb9_123* | 22510 | 14787 | 14545 | 14507 | 14487 | 35.64 | 135.359 |
| mod5adder_127 | 125 | 109 | 109 | 106 | 104 | 16.8 | 0.36 |
| mod5d1_63 | 11 | 11 | 11 | 10 | 9 | 18.18 | 0.015 |
| mod5d2_70 | 16 | 16 | 13 | 13 | 13 | 18.75 | 0.031 |
| mod5mils_65 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.047 |
| mod5mils_71 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.032 |
| peres_9 | 6 | 6 | 6 | 4 | 4 | 33.33 | 0.016 |
| plus127mod8192_162* | 73357 | 44382 | 38158 | 35382 | 35348 | 51.81 | 70.546 |
| plus63mod4096_163* | 32539 | 18314 | 15432 | 14668 | 14652 | 54.97 | 22.844 |
| plus63mod8192_164* | 45025 | 24038 | 20732 | 19578 | 19566 | 56.54 | 30.64 |
| rd32-v0_66 | 12 | 12 | 12 | 8 | 6 | 50 | 0.016 |
| rd32-v0_67 | 8 | 12 | 12 | 8 | 6 | 25 | 0.016 |
| rd32-v1_68 | 13 | 13 | 13 | 9 | 7 | 46.15 | 0.063 |
| rd32-v1_69 | 9 | 13 | 13 | 9 | 7 | 22.22 | 0.031 |
| rd53_130 | 232 | 199 | 199 | 195 | 195 | 15.95 | 0.718 |
| rd53_131 | 119 | 101 | 95 | 91 | 90 | 24.37 | 0.437 |
| rd53_132 | 117 | 101 | 95 | 91 | 90 | 23.08 | 0.328 |
| rd53_133 | 128 | 104 | 86 | 78 | 72 | 43.75 | 0.281 |
| rd53_134 | 120 | 104 | 86 | 78 | 72 | 40 | 0.265 |
| rd53_135 | 77 | 72 | 69 | 60 | 59 | 23.38 | 0.36 |
| rd53_136 | 75 | 72 | 69 | 60 | 59 | 21.33 | 0.375 |
| rd53_138 | 44 | 44 | 44 | 35 | 32 | 27.27 | 0.187 |
| rd73_140 | 76 | 76 | 76 | 61 | 58 | 23.68 | 0.281 |
| rd84_142 | 112 | 112 | 112 | 94 | 90 | 19.64 | 0.704 |
| sym6_145 | 777 | 567 | 215 | 213 | 212 | 72.72 | 1.531 |
| sym9_146 | 108 | 108 | 108 | 88 | 87 | 19.44 | 0.516 |
| sym9_148 | 4368 | 3696 | 672 | 672 | 672 | 84.62 | 4.203 |
| sys6-v0_111 | 72 | 72 | 72 | 59 | 55 | 23.61 | 0.36 |
| toffoli_double_4 | 10 | 10 | 7 | 7 | 7 | 30 | 0.016 |
| **Total** | **458358** | **307101** | **283857** | **265397** | **264904** | ** | **848.812** |

* An ancillary line is added for decomposing MPMCT gates that use all lines of the circuit.

** Cumulative Improvement: 42.21%, Average Improvement: 30.32%

Table 6.2: Results for optimized mapping MPMCT circuits to 74 NCV circuits with an added helper line.

| REVLIB Circuit | REVLIB Cost | Initial Cost | MCT Reduction | Quantum Expansion | Quantum Reduction | % Cost Reduction | CPU (sec.) |
|---|---|---|---|---|---|---|---|
| 3_17_13 | 14 | 14 | 14 | 11 | 11 | 21.43 | 0.062 |
| 4gt12-v0_86 | 58 | 50 | 42 | 36 | 35 | 39.66 | 0.094 |
| 4gt12-v0_87 | 54 | 46 | 38 | 36 | 35 | 35.19 | 0.079 |
| 4gt12-v0_88 | 41 | 32 | 28 | 28 | 28 | 31.71 | 0.063 |
| 4gt12-v1_89 | 45 | 37 | 32 | 31 | 31 | 31.11 | 0.047 |
| 4gt4-v0_72 | 54 | 46 | 38 | 38 | 37 | 31.48 | 0.172 |
| 4gt4-v0_73 | 89 | 83 | 45 | 41 | 41 | 53.93 | 0.219 |
| 4gt4-v0_78 | 53 | 45 | 45 | 45 | 45 | 15.09 | 0.078 |
| 4gt4-v0_79 | 49 | 41 | 41 | 41 | 41 | 16.33 | 0.047 |
| 4gt4-v0_80 | 37 | 28 | 28 | 28 | 28 | 24.32 | 0.063 |
| 4gt4-v1_74 | 57 | 49 | 44 | 42 | 42 | 26.32 | 0.062 |
| 4mod5-v0_18 | 25 | 25 | 19 | 17 | 17 | 32 | 0.047 |
| 4mod5-v0_19 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.016 |
| 4mod5-v0_20 | 9 | 9 | 9 | 7 | 7 | 22.22 | 0.016 |
| 4mod5-v1_22 | 9 | 9 | 9 | 7 | 6 | 33.33 | 0.031 |
| 4mod5-v1_23 | 24 | 24 | 18 | 16 | 16 | 33.33 | 0.046 |
| 4mod7-v0_94 | 38 | 40 | 32 | 26 | 24 | 36.84 | 0.109 |
| 4mod7-v0_95 | 38 | 40 | 32 | 26 | 24 | 36.84 | 0.094 |
| 4mod7-v1_96 | 39 | 41 | 33 | 27 | 25 | 35.9 | 0.14 |
| alu-v2_31 | 101 | 107 | 54 | 43 | 42 | 58.42 | 0.094 |
| cycle10_2_110 | 1202 | 764 | 378 | 355 | 355 | 70.47 | 0.547 |
| decod24-v0_38 | 18 | 18 | 18 | 14 | 14 | 22.22 | 0.016 |
| decod24-v2_43 | 18 | 18 | 18 | 14 | 14 | 22.22 | 0.031 |
| fredkin_6 | 15 | 15 | 15 | 13 | 12 | 20 | 0.016 |
| ham15_107 | 1831 | 1598 | 888 | 841 | 835 | 54.4 | 8.219 |
| ham15_108 | 453 | 392 | 292 | 262 | 257 | 43.27 | 1.656 |
| ham3_102 | 9 | 9 | 9 | 7 | 7 | 22.22 | 0.016 |
| ham3_103 | 10 | 10 | 10 | 8 | 8 | 20 | 0.016 |
| hwb4_49 | 65 | 67 | 51 | 44 | 42 | 35.38 | 0.093 |
| hwb6_56 | 1530 | 1280 | 971 | 892 | 881 | 42.42 | 4.36 |
| hwb6_57 | 1171 | 951 | 798 | 781 | 777 | 33.65 | 2.61 |
| hwb7_59 | 5236 | 3983 | 2730 | 2529 | 2500 | 52.25 | 15.578 |
| hwb7_60 | 4170 | 3130 | 2555 | 2537 | 2515 | 39.69 | 8.531 |
| hwb7_61 | 3876 | 3135 | 2288 | 2151 | 2132 | 44.99 | 12.187 |
| hwb7_62 | 2611 | 1995 | 1672 | 1661 | 1652 | 36.73 | 5.625 |
| hwb8_113 | 16530 | 11522 | 8201 | 7688 | 7635 | 53.81 | 47.11 |
| hwb8_114 | 14699 | 9840 | 7192 | 6677 | 6643 | 54.81 | 42.437 |
| hwb8_115 | 14691 | 9840 | 7192 | 6677 | 6643 | 54.78 | 41.219 |
| hwb8_116 | 7015 | 4867 | 3969 | 3943 | 3924 | 44.06 | 32.61 |
| hwb8_117 | 7013 | 4867 | 3969 | 3943 | 3924 | 44.05 | 32.359 |
| hwb8_118 | 16522 | 11522 | 8201 | 7688 | 7635 | 53.79 | 49.109 |
| hwb9_119 | 44714 | 32349 | 22985 | 21599 | 21506 | 51.9 | 90.39 |
| hwb9_120 | 44702 | 32349 | 22985 | 21599 | 21506 | 51.89 | 90.469 |
| hwb9_121 | 44665 | 32312 | 22967 | 21587 | 21496 | 51.87 | 92.406 |
| hwb9_122 | 44653 | 32312 | 22967 | 21587 | 21496 | 51.86 | 90.906 |
| hwb9_123 | 22510 | 14787 | 11614 | 11569 | 11526 | 48.8 | 64.047 |

Continued on the next page.

Continued from the previous page.

| REVLIB Circuit | REVLIB Cost | Initial Cost | MCT Reduction | Quantum Expansion | Quantum Reduction | % Cost Reduction | CPU (sec.) |
|---|---|---|---|---|---|---|---|
| mod5adder_127 | 125 | 109 | 104 | 100 | 99 | 20.8 | 0.64 |
| mod5d1_63 | 11 | 11 | 11 | 10 | 9 | 18.18 | 0.031 |
| mod5d2_70 | 16 | 16 | 13 | 13 | 13 | 18.75 | 0.031 |
| mod5mils_65 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.031 |
| mod5mils_71 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.031 |
| peres_9 | 6 | 6 | 6 | 4 | 4 | 33.33 | 0.016 |
| plus127mod8192_162 | 73357 | 44382 | 34989 | 32458 | 32426 | 55.8 | 79.719 |
| plus63mod4096_163 | 32539 | 18314 | 13863 | 13192 | 13176 | 59.51 | 27.109 |
| plus63mod8192_164 | 45025 | 24038 | 18780 | 17741 | 17724 | 60.64 | 33.188 |
| rd32-v0_66 | 12 | 12 | 12 | 8 | 6 | 50 | 0.015 |
| rd32-v0_67 | 8 | 12 | 12 | 8 | 6 | 25 | 0.016 |
| rd32-v1_68 | 13 | 13 | 13 | 9 | 7 | 46.15 | 0.016 |
| rd32-v1_69 | 9 | 13 | 13 | 9 | 7 | 22.22 | 0.016 |
| rd53_130 | 232 | 199 | 191 | 185 | 185 | 20.26 | 0.922 |
| rd53_131 | 119 | 101 | 92 | 90 | 90 | 24.37 | 0.625 |
| rd53_132 | 117 | 101 | 92 | 90 | 90 | 23.08 | 0.594 |
| rd53_133 | 128 | 104 | 80 | 72 | 68 | 46.88 | 0.141 |
| rd53_134 | 120 | 104 | 80 | 72 | 68 | 43.33 | 0.188 |
| rd53_135 | 77 | 72 | 58 | 56 | 56 | 27.27 | 0.172 |
| rd53_136 | 75 | 72 | 58 | 56 | 56 | 25.33 | 0.172 |
| rd53_138 | 44 | 44 | 44 | 35 | 32 | 27.27 | 0.172 |
| rd73_140 | 76 | 76 | 76 | 61 | 58 | 23.68 | 0.281 |
| rd84_142 | 112 | 112 | 112 | 94 | 90 | 19.64 | 0.828 |
| sym6_145 | 777 | 567 | 154 | 150 | 150 | 80.69 | 1.391 |
| sym9_146 | 108 | 108 | 108 | 88 | 87 | 19.44 | 0.5 |
| sym9_148 | 4368 | 3696 | 545 | 541 | 541 | 87.61 | 8.672 |
| sys6-v0_111 | 72 | 72 | 72 | 59 | 55 | 23.61 | 0.359 |
| toffoli_double_4 | 10 | 10 | 7 | 7 | 7 | 30 | 0.016 |
| **Total** | **458358** | **307101** | **225151** | **212447** | **211607** | ** | **890.034** |

** Cumulative Improvement: 53.83%, Average Improvement: 37.33%

### 6.3.2 MPMCT to NCVW Circuit Mapping

The MPMCT to NCVW mapping procedure was implemented using Python 2.6.5. The circuits considered are from the REVLIB web site [9]. Our experiments were run on a system with a 3.2 GHz i5-650 CPU and 3.0 GB RAM. Table 6.3 presents the results of mapping MPMCT circuits to NCVW circuits for 78 benchmark circuits from REVLIB for which the improvement has been 15% or more. A * after the circuit name indicates the addition of a single ancillary line as before. Our program applies the methods above to the circuit in both the forward and the reverse direction. The results are reported for each circuit for the better of the two directions.

We report (1) the quantum cost from REVLIB, (2) the initial NCVW cost which is found by direct mapping MPMCT gates to the NCVW catalog circuits corresponding to Table 5.5, (3) the NCVW costs after MPMCT gates reduction is applied, (4) the NCVW cost after quantum gate expansion, (5) the NCVW cost after quantum gate reduction which is the NCVW cost of the final circuit, (6) the percentage cost reduction comparing the final NCVW cost to the REVLIB cost and (7) the CPU time for all steps.

Table 6.3: NCVW realizations of 78 REVLIB benchmarks.

| REVLIB Circuit | REVLIB Cost | Initial Cost | MCT Reduction | Quantum Expansion | Quantum Reduction | % Cost Reduction | CPU (sec.) |
|---|---|---|---|---|---|---|---|
| sym9_148 | 4368 | 3612 | 665 | 665 | 659 | 84.91 | 25.906 |
| sym6_145 | 777 | 543 | 203 | 201 | 199 | 74.39 | 4.719 |
| plus63mod8192_164* | 45025 | 22208 | 19318 | 18863 | 18856 | 58.12 | 109.157 |
| plus63mod4096_163* | 32539 | 16808 | 14322 | 13820 | 13813 | 57.55 | 81.984 |
| plus127mod8192_162* | 73357 | 41002 | 35550 | 34193 | 34178 | 53.41 | 218.672 |
| rd32-v0_66 | 12 | 12 | 12 | 8 | 6 | 50.00 | 0.047 |
| rd32-v1_68 | 13 | 13 | 13 | 9 | 7 | 46.15 | 0.078 |
| 4gt4-v0_73* | 89 | 80 | 48 | 48 | 48 | 46.07 | 0.485 |
| rd53_133 | 128 | 104 | 86 | 78 | 72 | 43.75 | 0.937 |
| cycle10_2_110 | 1202 | 694 | 694 | 682 | 682 | 43.26 | 3.250 |
| hwb8_114* | 14699 | 9131 | 8888 | 8392 | 8378 | 43.00 | 142.516 |
| hwb8_115* | 14691 | 9131 | 8888 | 8392 | 8378 | 42.97 | 142.906 |
| hwb8_113* | 16530 | 10736 | 10282 | 9804 | 9787 | 40.79 | 78.500 |
| hwb8_118* | 16522 | 10736 | 10282 | 9804 | 9787 | 40.76 | 78.469 |
| hwb9_123* | 22510 | 13494 | 13492 | 13456 | 13434 | 40.32 | 185.672 |
| rd53_134 | 120 | 104 | 86 | 78 | 72 | 40.00 | 0.922 |
| ham15_107 | 1831 | 1509 | 1184 | 1115 | 1107 | 39.54 | 14.188 |
| hwb9_119* | 44714 | 29842 | 29010 | 27389 | 27340 | 38.86 | 272.609 |
| hwb9_121* | 44665 | 29805 | 28982 | 27359 | 27311 | 38.85 | 258.141 |
| hwb9_120* | 44702 | 29842 | 29010 | 27389 | 27340 | 38.84 | 260.406 |
| hwb9_122* | 44653 | 29805 | 28982 | 27359 | 27311 | 38.84 | 257.672 |
| 4gt12-v0_86* | 58 | 49 | 38 | 36 | 36 | 37.93 | 0.328 |
| 4gt12-v0_87* | 54 | 45 | 34 | 34 | 34 | 37.04 | 0.187 |
| 4gt4-v0_72* | 54 | 45 | 34 | 34 | 34 | 37.04 | 0.281 |
| hwb7_59* | 5236 | 3772 | 3613 | 3363 | 3352 | 35.98 | 58.438 |
| hwb8_116* | 7015 | 4547 | 4547 | 4505 | 4496 | 35.91 | 108.171 |
| hwb8_117* | 7013 | 4547 | 4547 | 4505 | 4496 | 35.89 | 108.219 |
| 4mod5-v1_22 | 9 | 9 | 9 | 7 | 6 | 33.33 | 0.047 |
| 4mod5-v1_23 | 24 | 24 | 18 | 16 | 16 | 33.33 | 0.172 |
| peres_9 | 6 | 6 | 6 | 4 | 4 | 33.33 | 0.015 |
| hwb7_60* | 4170 | 2966 | 2844 | 2838 | 2829 | 32.16 | 30.234 |
| 4mod5-v0_18 | 25 | 25 | 19 | 17 | 17 | 32.00 | 0.141 |
| 4mod5-v0_19 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.047 |
| mod5mils_65 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.093 |
| mod5mils_71 | 13 | 13 | 10 | 9 | 9 | 30.77 | 0.094 |

Continued on the next page.

Continued from the previous page.

| REVLIB Circuit | REVLIB Cost | Initial Cost | MCT Reduction | Quantum Expansion | Quantum Reduction | % Cost Reduction | CPU (sec.) |
|---|---|---|---|---|---|---|---|
| toffoli_double_4 | 10 | 10 | 7 | 7 | 7 | 30.00 | 0.078 |
| hwb7_61* | 3876 | 2974 | 2906 | 2743 | 2731 | 29.54 | 41.891 |
| hwb6_57* | 1171 | 913 | 845 | 836 | 833 | 28.86 | 7.671 |
| hwb7_62* | 2611 | 1901 | 1901 | 1884 | 1878 | 28.07 | 18.719 |
| rd53_138 | 44 | 44 | 44 | 35 | 32 | 27.27 | 0.594 |
| 4gt12-v0_88* | 41 | 32 | 32 | 30 | 30 | 26.83 | 0.172 |
| hwb6_56* | 1530 | 1227 | 1204 | 1126 | 1122 | 26.67 | 17.656 |
| rd32-v0_67 | 8 | 12 | 12 | 8 | 6 | 25.00 | 0.047 |
| rd53_135 | 77 | 71 | 68 | 59 | 58 | 24.68 | 1.313 |
| hwb4_49* | 65 | 65 | 57 | 51 | 49 | 24.62 | 0.438 |
| rd53_131 | 119 | 101 | 95 | 91 | 90 | 24.37 | 1.125 |
| 4gt4-v0_80* | 37 | 28 | 28 | 28 | 28 | 24.32 | 0.172 |
| rd73_140 | 76 | 76 | 76 | 61 | 58 | 23.68 | 1.016 |
| sys6-v0_111 | 72 | 72 | 72 | 59 | 55 | 23.61 | 1.141 |
| rd53_132 | 117 | 101 | 95 | 91 | 90 | 23.08 | 1.125 |
| alu-v2_31 | 101 | 101 | 84 | 78 | 78 | 22.77 | 0.578 |
| rd53_136 | 75 | 71 | 68 | 59 | 58 | 22.67 | 1.297 |
| 4gt4-v0_79* | 49 | 40 | 40 | 38 | 38 | 22.45 | 0.375 |
| 4mod5-v0_20 | 9 | 9 | 9 | 7 | 7 | 22.22 | 0.047 |
| decod24-v0_38 | 18 | 18 | 18 | 14 | 14 | 22.22 | 0.047 |
| decod24-v2_43 | 18 | 18 | 18 | 14 | 14 | 22.22 | 0.079 |
| ham3_102 | 9 | 9 | 9 | 7 | 7 | 22.22 | 0.031 |
| hwb4_50* | 63 | 65 | 57 | 51 | 49 | 22.22 | 0.437 |
| rd32-v1_69 | 9 | 13 | 13 | 9 | 7 | 22.22 | 0.062 |
| 3_17_13 | 14 | 14 | 14 | 11 | 11 | 21.43 | 0.125 |
| 4gt4-v0_78* | 53 | 44 | 44 | 44 | 42 | 20.75 | 0.265 |
| ham15_108 | 453 | 387 | 378 | 362 | 360 | 20.53 | 5.485 |
| 4gt12-v1_89* | 45 | 36 | 36 | 36 | 36 | 20.00 | 0.156 |
| fredkin_6 | 15 | 15 | 15 | 13 | 12 | 20.00 | 0.031 |
| ham3_103 | 10 | 10 | 10 | 8 | 8 | 20.00 | 0.047 |
| rd84_142 | 112 | 112 | 112 | 94 | 90 | 19.64 | 2.640 |
| sym9_146 | 108 | 108 | 108 | 88 | 87 | 19.44 | 1.422 |
| mod5d2_70 | 16 | 16 | 13 | 13 | 13 | 18.75 | 0.109 |
| 4mod7-v0_94 | 38 | 38 | 38 | 32 | 31 | 18.42 | 0.125 |
| 4mod7-v0_95 | 38 | 38 | 38 | 32 | 31 | 18.42 | 0.125 |
| mod5adder_127 | 125 | 107 | 107 | 104 | 102 | 18.40 | 1.266 |
| mod5d1_63 | 11 | 11 | 11 | 10 | 9 | 18.18 | 0.078 |
| 4mod7-v1_96 | 39 | 39 | 39 | 33 | 32 | 17.95 | 0.188 |
| rd53_130 | 232 | 196 | 196 | 192 | 191 | 17.67 | 2.579 |
| 4gt4-v1_74* | 57 | 48 | 48 | 47 | 47 | 17.54 | 0.218 |
| 4_49_16* | 60 | 60 | 57 | 53 | 50 | 16.67 | 0.406 |
| 4gt13_91 | 30 | 30 | 27 | 25 | 25 | 16.67 | 0.157 |
| one-two-three-v0_98 | 40 | 40 | 40 | 34 | 34 | 15.00 | 0.187 |
| **Total** | **458551** | **284605** | **264825** | **253107** | **252662** | ** | **2555.423** |

** Cumulative Improvement: 44.90%, Average Improvement: 31.08%

Table 6.4: Benchmarks for which the NCVW cost is greater than the NCV cost.

| REVLIB Circuit | NCV Cost [81] | NCVW Cost | % Cost Increase |
|---|---|---|---|
| 4gt4-v1_74* | 46 | 47 | -2.17 |
| ham15_108 | 356 | 360 | -1.12 |
| one-two-three-v0_97 | 62 | 63 | -1.61 |
| one-two-three-v1_99 | 31 | 32 | -3.23 |

Overall, for the circuits reported in Table 6.3, our methods yield a 44.9% cumulative improvement and a 31.08% improvement on average compared to the costs reported in REVLIB. The majority of the improvement (37.9%) comes from the catalog circuits. The rest comes from our mapping and quantum reduction techniques. The cumulative and average improvements over all the circuits in the test suite are 44.71% and 10.6% respectively.

Table 6.4 shows the four cases where the NCVW circuit is more costly than the NCV circuit. This is a result of the fact that our methods use many heuristics.

Table 6.5 shows the benchmarks for which the NCVW circuit is an improvement over the NCV circuit. The overall improvement for these examples is 4.71%.

Table 6.5: Benchmarks where NCVW circuit cost is less than NCV circuit cost.

| REVLIB Circuit | NCV Cost [81] | NCVW Cost | % Cost Reduction |
|---|---|---|---|
| 4_49_16* | 55 | 50 | 9.09 |
| 4gt10-v1_81 | 35 | 32 | 8.57 |
| 4gt12-v1_89* | 37 | 36 | 2.7 |
| 4gt13_91 | 28 | 25 | 10.7 |
| 4gt4-v0_73* | 49 | 48 | 2.04 |
| 4gt4-v0_78* | 45 | 42 | 6.67 |
| 4gt4-v0_79* | 41 | 38 | 7.32 |
| 4gt5_75 | 22 | 21 | 4.55 |
| 4gt5_76 | 27 | 26 | 3.7 |
| 4gt5_77 | 28 | 26 | 7.14 |
| 4mod7-v0_94 | 32 | 31 | 3.13 |
| 4mod7-v0_95 | 32 | 31 | 3.13 |
| 4mod7-v1_96 | 33 | 32 | 3.03 |
| alu-v2_30* | 103 | 98 | 4.85 |
| alu-v2_31 | 83 | 78 | 6.02 |
| alu-v2_32 | 38 | 35 | 7.89 |
| alu-v4_36 | 28 | 27 | 3.57 |
| cycle10_2_110 | 720 | 682 | 5.28 |
| decod24-enable_126 | 77 | 75 | 2.6 |
| decod24-v1_41* | 23 | 20 | 13 |

Continued on the next page.

Continued from the previous page.

| REVLIB Circuit | NCV Cost [81] | NCVW Cost | % Cost Reduction |
|---|---|---|---|
| decod24-v3_45* | 35 | 31 | 11.4 |
| ham15_107 | 1155 | 1107 | 4.16 |
| ham15_109 | 198 | 190 | 4.04 |
| ham7_104 | 84 | 76 | 9.52 |
| ham7_105 | 64 | 59 | 7.81 |
| hwb4_49* | 54 | 49 | 9.26 |
| hwb4_50* | 54 | 49 | 9.26 |
| hwb4_51* | 73 | 71 | 2.74 |
| hwb5_53* | 282 | 270 | 4.26 |
| hwb5_54* | 234 | 220 | 5.98 |
| hwb5_55 | 95 | 93 | 2.11 |
| hwb6_56* | 1150 | 1122 | 2.43 |
| hwb6_57* | 872 | 833 | 4.47 |
| hwb6_58 | 132 | 127 | 3.79 |
| hwb7_59* | 3500 | 3352 | 4.23 |
| hwb7_60* | 2989 | 2829 | 5.35 |
| hwb7_61* | 2863 | 2731 | 4.61 |
| hwb7_62* | 1973 | 1878 | 4.82 |
| hwb8_113* | 10328 | 9787 | 5.24 |
| hwb8_114* | 8815 | 8378 | 4.96 |
| hwb8_115* | 8815 | 8378 | 4.96 |
| hwb8_116* | 4825 | 4496 | 6.82 |
| hwb8_117* | 4825 | 4496 | 6.82 |
| hwb8_118* | 10328 | 9787 | 5.24 |
| hwb9_119* | 28660 | 27340 | 4.61 |
| hwb9_120* | 28660 | 27340 | 4.61 |
| hwb9_121* | 28629 | 27311 | 4.6 |
| hwb9_122* | 28629 | 27311 | 4.6 |
| hwb9_123* | 14487 | 13434 | 7.27 |
| mod5adder_127 | 104 | 102 | 1.92 |
| mod5adder_128 | 84 | 80 | 4.76 |
| mod5adder_129 | 76 | 74 | 2.63 |
| one-two-three-v0_98 | 35 | 34 | 2.86 |
| plus127mod8192_162* | 35348 | 34178 | 3.31 |
| plus63mod4096_163* | 14652 | 13813 | 5.73 |
| plus63mod8192_164* | 19566 | 18856 | 3.63 |
| rd53_130 | 195 | 191 | 2.05 |
| rd53_135 | 59 | 58 | 1.69 |
| rd53_136 | 59 | 58 | 1.69 |
| rd53_137 | 59 | 58 | 1.69 |
| sym6_145 | 212 | 199 | 6.13 |
| sym9_148 | 672 | 659 | 1.93 |
| **Total** | **265465** | **252958** | **4.71** |

### 6.3.3 MPMCT to NCV-$|v_1\rangle$ Circuit Mapping

The procedures for mapping MPMCT circuits to NCV-$|v_1\rangle$ circuits have been implemented using Python 2.7.1. Our experiments were run on a computer with a Core 2 Duo 2.66 GHz CPU and 4.0 GB RAM. We used a test suite of 138 circuits from REVLIB [9]. The results are shown in Table 6.6. Due to space limitations, the table shows only those circuits for which the improvement was 20% or more (73 of the 138 circuits).

Each row of the Table gives: (1) The name of the circuit including the REVLIB file index number. Note that Fredkin and Peres gates in the REVLIB circuits are substituted by MPMCT gate realizations before applying our techniques. (2) The quantum gate count given on the REVLIB site. (3) The NCV gate count for circuits as reported in Table 6.1. (4) The gate count for the direct mapping each MPMCT gate to the $2c + 1$ NCV-$|v_1\rangle$ gate realization in the given circuit. (5) The direct mapping NCV-$|v_1\rangle$ gate count is reported for the circuit found by first applying the MPMCT optimization as in Section 3.3.2. (6) The NCV-$|v_1\rangle$ gate count for the circuit from (5) optimized at the NCV-$|v_1\rangle$ gate level using the Gate Reduction Procedure (Procedure 3.1).

The cost reduction for the circuits listed in Table 6.6 is 82.14% with respect to the costs reported in REVLIB and 69.17% with respect to the NCV costs determined by the optimized MPMCT to NCV mapper described in Section 6.3.1. The cumulative cost reduction for all the circuits in the test suite is 81.83% and the average improvement is 23.89% compared to the REVLIB costs.

As the results in Table 6.6 show, the MPMCT to NCV-$|v_1\rangle$ circuit mapper does very well for medium to large circuits since those circuits tend to have more MPMCT gates with greater than 2 controls than do the small circuits. In addition the smaller circuits often have a high proportion of CNOT gates which are not primitives in the NCV-$|v_1\rangle$ library and are implemented with three NCV-$|v_1\rangle$ gates.

The total improvement comes primarily from the new MPMCT to NCV-$|v_1\rangle$ gate mapping. The MPMCT and NCV-$|v_1\rangle$ optimizations that are applied before and after mapping respectively reduce the quantum cost by a further 4.5% and 30.5% on average. Note that additional lines are never required for NCV-$|v_1\rangle$ circuits. Considering unit delay for all 1-qubit and 2-qubit quantum gates as in [1], NCV-$|v_1\rangle$ circuits are much faster than the NCV circuits as they have lower logic depth.

Table 6.6: Experimental Results for mapping MPMCT circuits to NCV-$|v_1\rangle$ circuits.

| Benchmark | Previous Approaches | | Proposed Approaches | | | % Improv. wrt | % Improv. wrt |
|---|---|---|---|---|---|---|---|
| | REVLIB [9] | NCV [106] | Direct Mapping | MPMCT Opt. | NCV-$|v_1\rangle$ Opt. | REVLIB | NCV |
| plus63mod8192_164 | 45025 | 19566 | 6620 | 5921 | 2135 | 95.26 | 89.09 |
| plus127mod8192_162 | 73357 | 35348 | 12318 | 10910 | 3972 | 94.59 | 88.76 |
| plus63mod4096_163 | 32539 | 14652 | 5327 | 4672 | 1779 | 94.53 | 87.86 |
| cycle10_2_110 | 1202 | 720 | 219 | 219 | 91 | 92.43 | 87.36 |
| sym9_148 | 4368 | 672 | 1722 | 616 | 374 | 91.44 | 44.35 |
| sym6_145 | 777 | 212 | 276 | 187 | 118 | 84.81 | 44.34 |
| hwb8_114 | 14699 | 8815 | 4456 | 4378 | 3235 | 77.99 | 63.30 |
| hwb8_115 | 14691 | 8815 | 4456 | 4378 | 3237 | 77.97 | 63.28 |
| hwb9_121 | 44665 | 28629 | 13149 | 12920 | 10156 | 77.26 | 64.53 |
| hwb9_122 | 44653 | 28629 | 13149 | 12920 | 10156 | 77.26 | 64.53 |
| hwb9_119 | 44714 | 28660 | 13168 | 12938 | 10180 | 77.23 | 64.48 |
| hwb9_120 | 44702 | 28660 | 13168 | 12938 | 10180 | 77.23 | 64.48 |
| hwb8_113 | 16530 | 10328 | 5065 | 4957 | 3786 | 77.10 | 63.34 |
| hwb8_118 | 16522 | 10328 | 5065 | 4957 | 3786 | 77.09 | 63.34 |
| ham15_107 | 1831 | 1155 | 836 | 724 | 447 | 75.59 | 61.30 |
| hwb9_123 | 22510 | 14487 | 9151 | 9145 | 5704 | 74.66 | 60.63 |
| hwb7_59 | 5236 | 3500 | 2017 | 1969 | 1434 | 72.61 | 59.03 |
| hwb8_116 | 7015 | 4825 | 3383 | 3383 | 2109 | 69.94 | 56.29 |
| hwb8_117 | 7013 | 4825 | 3383 | 3383 | 2109 | 69.93 | 56.29 |
| hwb7_61 | 3876 | 2863 | 1622 | 1596 | 1226 | 68.37 | 57.18 |
| 4gt4-v0_72 | 54 | 34 | 30 | 25 | 18 | 66.67 | 47.06 |
| 4gt4-v0_73 | 89 | 49 | 73 | 53 | 31 | 65.17 | 36.73 |
| rd53_133 | 128 | 72 | 68 | 65 | 45 | 64.84 | 37.50 |
| hwb6_56 | 1530 | 1150 | 766 | 756 | 546 | 64.31 | 52.52 |
| hwb7_60 | 4170 | 2989 | 2286 | 2121 | 1524 | 63.45 | 49.01 |
| hwb7_62 | 2611 | 1973 | 1495 | 1495 | 957 | 63.35 | 51.50 |
| rd53_134 | 120 | 72 | 68 | 65 | 45 | 62.50 | 37.50 |
| rd53_130 | 232 | 195 | 112 | 112 | 93 | 59.91 | 52.31 |
| 4gt4-v1_74 | 57 | 46 | 31 | 31 | 23 | 59.65 | 50.00 |
| hwb6_57 | 1171 | 872 | 829 | 728 | 473 | 59.61 | 45.76 |
| 4gt12-v1_89 | 45 | 37 | 23 | 23 | 19 | 57.78 | 48.65 |
| ham15_108 | 453 | 356 | 320 | 321 | 202 | 55.41 | 43.26 |
| rd53_131 | 119 | 90 | 76 | 70 | 55 | 53.78 | 38.89 |
| rd53_132 | 117 | 90 | 76 | 70 | 55 | 52.99 | 38.89 |
| alu-v2_31 | 101 | 83 | 69 | 70 | 48 | 52.48 | 42.17 |
| mod5adder_127 | 125 | 104 | 75 | 75 | 60 | 52.00 | 42.31 |
| alu-v2_30 | 114 | 103 | 82 | 79 | 55 | 51.75 | 46.60 |
| 4gt4-v0_80 | 37 | 28 | 21 | 21 | 18 | 51.35 | 35.71 |
| 4gt12-v0_86 | 58 | 36 | 54 | 49 | 30 | 48.28 | 16.67 |
| 4gt12-v0_87 | 54 | 34 | 42 | 37 | 28 | 48.15 | 17.65 |
| hwb5_53 | 315 | 282 | 257 | 254 | 166 | 47.30 | 41.13 |
| mod5adder_128 | 83 | 84 | 59 | 59 | 45 | 45.78 | 46.43 |
| decod24-v3_45 | 35 | 35 | 25 | 25 | 19 | 45.71 | 45.71 |
| 4gt12-v0_88 | 41 | 30 | 25 | 25 | 23 | 43.90 | 23.33 |
| one-two-three-v0_97 | 71 | 62 | 57 | 57 | 40 | 43.66 | 35.48 |

Continued on the next page.

Continued from the previous page.

| Benchmark | Previous Approaches | | Proposed Approaches | | | % Improv. wrt | % Improv. wrt |
|---|---|---|---|---|---|---|---|
| | REVLIB [9] | NCV [106] | Direct Mapping | MPMCT Opt. | NCV-$\|v_1\rangle$ Opt. | REVLIB | NCV |
| alu-v2_32 | 39 | 38 | 31 | 28 | 22 | 43.59 | 42.11 |
| 4mod7-v1_96 | 39 | 33 | 31 | 31 | 23 | 41.03 | 30.30 |
| rd53_135 | 77 | 59 | 70 | 68 | 46 | 40.26 | 22.03 |
| fredkin_6 | 15 | 12 | 15 | 15 | 9 | 40.00 | 25.00 |
| decod24-enable_126 | 86 | 77 | 72 | 72 | 52 | 39.53 | 32.47 |
| 4mod7-v0_94 | 38 | 32 | 30 | 30 | 23 | 39.47 | 28.13 |
| 4mod7-v0_95 | 38 | 32 | 30 | 30 | 23 | 39.47 | 28.13 |
| 4gt4-v0_79 | 49 | 41 | 37 | 37 | 30 | 38.78 | 26.83 |
| rd53_136 | 75 | 59 | 70 | 68 | 46 | 38.67 | 22.03 |
| 4gt4-v0_78 | 53 | 45 | 49 | 49 | 33 | 37.74 | 26.67 |
| mod5adder_129 | 77 | 76 | 65 | 65 | 48 | 37.66 | 36.84 |
| 4gt5_77 | 28 | 28 | 20 | 20 | 18 | 35.71 | 35.71 |
| rd53_137 | 65 | 59 | 60 | 60 | 42 | 35.38 | 28.81 |
| one-two-three-v0_98 | 40 | 35 | 38 | 38 | 26 | 35.00 | 25.71 |
| ham7_104 | 83 | 84 | 91 | 91 | 55 | 33.73 | 34.52 |
| decod24-v2_43 | 18 | 14 | 20 | 20 | 12 | 33.33 | 14.29 |
| hwb5_54 | 248 | 234 | 276 | 276 | 166 | 33.06 | 29.06 |
| 4gt10-v1_81 | 34 | 35 | 28 | 28 | 23 | 32.35 | 34.29 |
| hwb5_55 | 104 | 95 | 96 | 97 | 71 | 31.73 | 25.26 |
| 4_49_16 | 60 | 55 | 64 | 61 | 41 | 31.67 | 25.45 |
| hwb4_49 | 65 | 54 | 71 | 63 | 46 | 29.23 | 14.81 |
| 4mod5-v0_18 | 25 | 17 | 31 | 33 | 18 | 28.00 | -5.88 |
| decod24-v1_41 | 22 | 23 | 22 | 22 | 16 | 27.27 | 30.43 |
| hwb4_51 | 81 | 73 | 119 | 101 | 59 | 27.16 | 19.18 |
| hwb4_50 | 63 | 54 | 71 | 63 | 46 | 26.98 | 14.81 |
| ham7_105 | 65 | 64 | 81 | 81 | 51 | 21.54 | 20.31 |
| rd84_142 | 112 | 90 | 126 | 126 | 89 | 20.54 | 1.11 |
| hwb6_58 | 142 | 132 | 168 | 169 | 113 | 20.42 | 14.39 |
| **Total** | **459696** | **266245** | **127451** | **121639** | **82079** | **82.14** | **69.17** |
| **Un-weighted Average** | | | | | | **53.73** | **41.22** |

## 6.4 Summary

This chapter completes the proposed optimized mapping flow for reversible to quantum circuit mapping by using the concepts from the previous chapters. For each targeted quantum library the corresponding techniques were described and the associated optimized mapping results were reported. The results show that using an optimized flow, the quantum cost of reversible circuits can be improved by about 50%. The cost reduction is significant and proves the importance of using an optimized mapping flow.

# Chapter 7

# Conclusions and Future Work

The increasing demand for higher computational power has led the trend of developing technologies to move towards building highly integrated circuits. Products with 20nm CMOS technology have already entered the market and future designs are being planned based on 14nm technologies. The integration of circuits becomes more challenging as it goes forward and it is believed it will hit fundamental limits in the near future. To face the upcoming challenges, alternative technologies are required. Reversible logic and quantum technologies are alternatives that may address this problem.

Although promising results have been obtained in the reversible and quantum design automation so far, this research area is still in its early years. While a lot of effort has been made in some steps of the reversible and quantum design automation flow such as the synthesis of reversible and quantum circuits, some areas such as technology mapping reversible to quantum circuits have remained open and require further consideration.

Synthesis of reversible and quantum circuits is an intractable problem and often resulting circuits are not optimal. On the other hand, the circuits resulting from synthesis are frequently mapped to circuits of elementary quantum gates for cost calculation or for physical implementation. The technology mapping approaches that can be found in the literature are not optimized and introduce even more redundancies while propagating the ones existing in the synthesized circuit. Much of the generated redundancies in technology mapping approaches come from the non-optimal cost models that are used.

In this dissertation, an optimized methodology was introduced for technology mapping reversible circuits to quantum circuits. The proposed flow consists of three

major steps. The first step is the high level optimization in which a circuit resulting from a synthesis approach is optimized mostly in terms of the quantum cost as the number of elementary quantum gates. We generalized the moving rule for gate rearrangement at this stage and showed that it leads to finding more reductions using any optimization approach that uses the gate rearrangement for finding possible cancelations. We also proposed optimization techniques in Chapters 3 and 4 that employed the new moving rule to improve the quantum cost of the existing benchmark circuits.

At the second step of our optimized technology mapping flow, the reversible circuit is technology mapped into a target quantum library using a quantum cost model. Previous work on this step involves direct mapping of individual gates to their realizations based on an identified cost model. In this part, we modified the existing quantum cost models as discussed in Chapter 5. Furthermore, new quantum gate libraries were introduced and the corresponding cost models for a class of reversible circuits were discussed. In Chapter 6, we explored possible approaches to map reversible gates to quantum gates and appropriate optimized mapping methods were proposed for the existing and our new target quantum gate libraries.

Finally, the last stage of the optimized reversible to quantum mapping methodology is to optimize the quantum circuit resulting from the mapping step. We introduced new optimization methods for optimizing quantum circuits at this step.

Recently, some efforts have been made to introduce reversible circuit design flows for realizing reversible or irreversible functions [45]. In this dissertation, we completed the reversible circuits design methodology by connecting the end point of the existing flows, which is usually reversible circuits, to actual implementable quantum circuits. The methods that we proposed use the uniform REVLIB file format for reversible functions and circuits thereby linking the proposed approaches together and to the previous levels of the design.

The techniques described in this dissertation were focused on a certain class of reversible gates; However, the proposed flow is applicable to other reversible gate libraries. In addition, we here discussed three target quantum gate libraries, but the proposed methods can be modified to use other quantum gate libraries. In particular, the NCV and NCVW quantum gate libraries considered in this dissertation contained only positive control gates. In our future work, we will consider libraries containing $V$, $V^+$, $W$, $W^+$, and CNOT gates with negative controls.

We used a decision diagram structure called Decision Diagram for a Matrix Function (DDMF) in our optimization method. This structure has limitations and cannot

be used for all quantum circuits. Our future work includes replacing DDMFs with a better structure in our optimization methods.

The conventional and the new moving rules described in Chapter 3 are based on the assumption that matrices describing the target operations for the gates being interchanged commute. The matrices for all target operations that were used in this dissertation commute. Our future work includes modifying the new moving rule to consider other gate libraries that do not have this property.

The functionality of controlled reversible gates (such as MPMCT gates) is invariant under the permutation of the controls of the same polarity. This property can be used in optimized mapping of such gates to quantum circuits as the quantum realizations change by changing the order of the controls. In our proposed mappings, we used this property for smaller MPMCT gates (Toffoli gates). Considering control permutation for larger reversible gates is in our future work. The proposed mapping structures for NCV and NCVW libraries are the best known so far, but we have no proof that they are optimal and searching for better mappings is in our future work.

As noted in Chapter 5, some technologies require additional constraints in the circuit such as having the nearest-neighbor property. Adding this property to a circuit is expensive and requires including extra Swap gates in the circuit. Our future work will concentrate on the nearest-neighbor problem and in particular on how to incorporate that constraint into existing synthesis, optimization and mapping methods.

# Bibliography

[1] H. Thapliyal and N. Ranganathan. Design of efficient reversible logic based binary and bcd adder circuits. *ACM J. on Emerging Technologies in computing Systems*, 2012. to appear.

[2] D. Maslov, G. W. Dueck, and D. M. Miller. Simplification of Toffoli networks via templates. In *Symp. on Integrated Circuits and System Design*, pages 53–58, 2003.

[3] D. Maslov, G. W. Dueck, and D. M. Miller. Toffoli network synthesis with templates. *IEEE Trans. on CAD*, 24(6):807–817, 2005.

[4] D. Maslov, C. Young, D. M. Miller, and G. W. Dueck. Quantum circuit simplification using templates. In *Proc. Design, Automation and Test in Europe*, pages 1208–1213, 2005.

[5] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, M. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457–3467, 1995.

[6] D. Maslov and G. W. Dueck. Improved quantum cost for $n$-bit Toffoli gates. *Electronic Letters*, 39(25):1790–1791, 2003.

[7] D. Maslov and D. M. Miller. Comparison of the cost metrics through investigation of the relation between optimal NCV and optimal NCT 3-qubit reversible circuits. *IET Computers and Digital Techniques*, 1(2):98–104, 2007.

[8] M. Szyprowski and P. Kerntopf. Reducing quantum cost of pairs of multi-control Toffoli gates. In *Workshop on Boolean Problems*, pages 263–268, 2012.

[9] R. Wille, D. Große, L. Teuber, G. W. Dueck, and R. Drechsler. RevLib: An online resource for reversible functions and reversible circuits. In *Int'l Symp. on Multi-Valued Logic*, pages 220–225, 2008. RevLib is available at www.revlib.org.

[10] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, 1965.

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

[12] C. H. Bennett. Logical reversibility of computation. *IBM J. Research and Development*, 17(6):525–532, 1973.

[13] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[14] C. H. Bennett. Notes on the history of reversible computation. *IBM J. Research and Development*, 32(1):16–23, 1988.

[15] A. N. Al-Rabadi. *Reversible Logic Synthesis: From Fundamentals to Quantum Computing*. Springer-Verlag, 2003.

[16] B. Desoete and A. De Vos. A reversible carry-look-ahead adder using control gates. *INTEGRATION, the VLSI J.*, 33(1-2):89–104, 2002.

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

[18] R. Cuykendall and D. R. Andersen. Reversible optical computing circuits. *Optics Letters*, 12(7):542–544, 1987.

[19] T. Song, S. Wang, and X. Wang. The design of reversible gate and reversible sequential circuit based on DNA computing. In *Proc. Int'l Conf. on Intelligent System and Knowledge Engineering*, 2008.

[20] H. Thapliyal and M. B. Srinivas. The need of DNA computing: reversible designs of adders and multipliers using Fredkin gate. In *Proc. of SPIE*.

[21] R. C. Merkle. Reversible electronic logic using switches. *Nanotechnology*, 4(1):21–40, 1993.

[22] J. P. McGregor and R. B. Lee. Architectural enhancements for fast subword permutations with repetitions in cryptographic applications. In *Proc. Int'l Conf. on Comp. Design*, pages 453–461, 2001.

[23] D. C. Marinescu and G. M. Marinescu. *Approaching Quantum Computing*. Pearson Education, 2004.

[24] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. of Computing*, 26(5):1484 – 1509, 1997.

[25] D. Große, X. Chen, G. W. Dueck, and R. Drechsler. Exact SAT-based Toffoli network synthesis. In *Proc. Great Lakes Symp. VLSI*, pages 96–101, 2007.

[26] M. H. A. Khan and M. Perkowski. Multi-output ESOP synthesis with cascades of new reversible gate family. In *6th Int'l Symp. on Representations and Methodology of Future Computing Technologies*, pages 144–153, 2003.

[27] P. Kerntopf. A new heuristic algorithm for reversible logic synthesis. In *Proc. Design Automation Conf.*, pages 834–837, 2004.

[28] D. Maslov. *Reversible Logic Synthesis*. PhD thesis, University of New Brunswick, 2003.

[29] D. Maslov, G. W. Dueck, and D. M. Miller. Techniques for the synthesis of reversible Toffoli networks. *ACM Transactions on Design Automation of Electronic Systems*, 12(4):42.1–42.28, 2007.

[30] O. Golubitsky, S. M. Falconer, and D. Maslov. Synthesis of the optimal 4-bit reversible circuits. In *Proc. Design Automation Conf.*, pages 653 – 656, 2010.

[31] P. Kerntopf, M. Perkowski, and K. Podlaski. Synthesis of reversible circuits: A view on the state-of-the-art. In *Proc. IEEE Int'l Conf. on Nanotechnology*, 2012. to appear.

[32] P. Kerntopf. Research on reveresible computation in the year 2011. 2011. preprint.

[33] Z. Sasanian, M. Saeedi, M. Sedighi, and M. Saheb Zamani. A cycle-based synthesis algorithm for reversible logic. In *Proc. ASP Design Automation Conf.*, pages 745–750, 2009.

[34] M. Saeedi, M. Saheb Zamani, M. Sedighi, and Z. Sasanian. Reversible circuit synthesis using a cycle-based approach. *ACM J. on Emerging Technologies in computing Systems*, 6(4):13.1–13.26, 2010.

[35] D. Große, R. Wille, G. W. Dueck, and R. Drechsler. Exact multiple-control Toffoli network synthesis with sat techniques. *IEEE Trans. on CAD*, 28(5):703–715, 2009.

[36] D. Große, X. Chen, and R. Drechsler. Exact Toffoli network synthesis of reversible logic using Boolean satisfiability. *IEEE Dallas/CAS Workshop on Design, Applications, Integration and Software*, pages 51–54, 2006.

[37] E. Curtis and M. Perkowski. A transformation based algorithm for ternary reversible logic synthesis using universally controlled ternary gates. In *Proc. Int'l Workshop on Logic Synthesis*, 2004.

[38] G. W. Dueck, D. Maslov, and D. M. Miller. Transformation-based synthesis of networks of Toffoli/Fredkin gates. In *Proc. IEEE Canadian Conf. on Electrical and Computer Engineering*, 2003.

[39] K. Iwama, Y. Kambayashi, and S. Yamashita. Transformation rules for designing CNOT-based quantum circuits. In *Proc. Design Automation Conf.*, pages 419–424, 2002.

[40] D. M. Miller, D. Maslov, and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In *Proc. Design Automation Conf.*, pages 318–323, 2003.

[41] M. Lukac and M. Perkowski. Evolutionary approach to quantum symbolic logic synthesis. In *Proc. Int'l Conf. on Evolutionary Computation*, pages 3374 – 3380, 2008.

[42] M. Lukac, M. Perkowski, and M. Kameyama. Evolutionary quantum logic synthesis of Boolean reversible logic circuits embedded in ternary quantum space using structural restrictions. In *Proc. Int'l Conf. on Evolutionary Computation*, pages 1–8, 2010.

[43] R. Wille and R. Drechsler. BDD-based synthesis of reversible logic for large functions. In *Proc. Design Automation Conf.*, pages 270–275, 2009.

[44] M. Soeken, R. Wille, C. Hilken, N. Przigoda, and R. Drechsler. Synthesis of reversible circuits with minimal lines for large functions. In *Proc. Workshop on Reversible Computation*, pages 59–70, 2011.

[45] R. Wille and R. Drechsler. *Towards a Design Flow for Reversible Logic*. Springer, 2010.

[46] Y. Hirata, M. Nakanishi, S. Yamashita, and Y. Nakashima. An efficient method to convert arbitrary quantum circuits to ones on a linear nearest neighbor architecture. In *Proc. Int'l Conf. on Quantum, Nano and Micro Technologies*, pages 26 – 33, 2009.

[47] M. Saeedi, R. Wille, and R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing*, 10(3):355–377, 2011.

[48] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Computers*, 35(8):677–691, 1986.

[49] S. Yamashita, S. I. Minato, and D. M. Miller. An efficient verification of quantum circuits under a practical restriction. In *Proc IEEE Int'l Conf. on Computer and Information Technology.*, pages 873–879, 2008.

[50] D. M. Miller and M. A. Thornton. *Multiple-Valued Logic: Concepts and Representations*. Morgan and Claypool, 2008.

[51] P. A. M. Dirac. A new notation for quantum mechanics. *Mathematical Proc. of the Cambridge Philosophical Society*, 35(3):416–418, 1939.

[52] R. Van Meter and M. Oskin. Architectural implications of quantum computing technologies. *ACM J. on Emerging Technologies in computing Systems*, 2(1):31–63, 2006.

[53] L. M. K. Vandersypen, M. Steffen, G. Breyta, C. S. Yannoni, M. H. Sherwood, and I. L. Chuang. Experimental realization of shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, 414, 2001.

[54] N. Boulant, K. Edmonds, J. Yang, M. A. Pravia, and D. G. Cory. Experimental demonstration of an entanglement swapping operation and improved control in nmr quantum-information processing. *Physical Review A*, 68(3):032305, 2003.

[55] T. D. Ladd, J. R. Goldman, F. Yamaguchi, and Y. Yamamoto. All-silicon quantum computer. *Physical Review Letters*, 89(1):017901, 2002.

[56] D. Kielpinski1, C. Monroe, and D. J. Wineland. Architecture for a large-scale ion-trap quantum computer. *Nature*, 417, 2002.

[57] W. K. Wooters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982.

[58] D. Maslov and G. W. Dueck. Reversible cascades with minimal garbage. *IEEE Trans. on CAD*, 23(11):1497–1509, 2004.

[59] D. M. Miller, R. Wille, and G. W. Dueck. Synthesizing reversible circuits from irreversible specifications using Reed-Muller spectral techniques. In *Proc. Reed-Muller Workshop*, pages 87–96, 2009.

[60] D. M. Miller, R. Wille, and G. W. Dueck. Synthesizing reversible circuits for irreversible functions. In *Proc. European Conf. on Digital Systems Design*, pages 749–756, 2009.

[61] M. Perkowski, R. Fiszer, P. Kerntopf, and M. Lukac. An approach to synthesis of reversible circuits for partially specified functions. In *Proc. IEEE Int'l Conf. on Nanotechnology*, 2012. to appear.

[62] T. Toffoli. Reversible computing. Tech Memo LCS/TM-151, MIT Lab for Comp. Sci, 1980.

[63] A. Peres. Reversible logic and quantum computers. *Physical Review A*, 32(6):3266–3276, 1985.

[64] E. Fredkin and T. Toffoli. Conservative logic. *Int'l J. of Theoretical Physics*, 21:219–253, 1982.

[65] A. De Vos, S. Burignat, and M. K. Thomsen. Reversible implementation of a discrete integer linear transformation. *J. of Multiple-valued Logic and Soft Computing*, 18(1):25–35, 2012.

[66] A. De Vos. Reversible computer hardware. *Electronic Notes in Theoretical Computer Science*, 253(6):17–22, 2010.

[67] T. Sasao. AND-EXOR expressions and their optimization. In T. Sasao, editor, *Logic Synthesis and Optimization*, pages 287–312. Kluwer Academic Publisher, 1993.

[68] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[69] M. Saeedi, N. MohammadZadeh, M. Sedighi, and M. Saheb Zamani. Evaluation and improvement of quantum synthesis algorithms based on a thorough set of metrics. In *Proc. European Conf. on Digital Systems Design*, pages 490–493, 2008.

[70] T. Monz, P. Schindler, J. T. Barreiro, M. Chwalla, D. Nigg, W. A. Coish, M. Harlander, W. Hänsel, M. Hennrich, and R. Blatt. 14-qubit entanglement: Creation and coherence. *Physical Review Letters*, 106(13), 2011.

[71] D. M. Miller. Lower cost quantum gate realizations of multiple-control Toffoli gates. In *IEEE Pacific Rim Conference*, pages 308–313, 2009.

[72] D. M. Miller and M. A. Thornton. QMDD: A decision diagram structure for reversible and quantum circuits. In *Proc. Int'l Symp. on Multiple-valued Logic*, pages 30–30, 2006.

[73] S. Yamashita, S. Minato, and D. M. Miller. DDMF: An efficient decision diagram structure for design verification of quantum circuits under a practical restriction. *IEICE Trans. on Fundamentals*, E91-A(12):3793–3802, 2008.

[74] C. Y. Lee. Representation of switching circuits by binary decision diagrams. *Bell System Technical Journal*, 38:985–999, 1959.

[75] S. B. Akers. Binary decision diagrams. *IEEE Trans. on Computers*, 27(6):509–516, 1978.

[76] R. Drechsler and D. Sieling. Binary decision diagrams in theory and practice. *Int'l Journal on Software Tools for Technology Transfer*, 3:112–136, 2001.

[77] C. E. Shannon. Symbolic analysis of relay and switching circuits. *Trans. of the AIEE*, 57:713–723, 1938.

[78] D. M. Miller and Z. Sasanian. Lowering the quantum gate cost of reversible circuits. In *Proc. Midwest Symp. on Circuits and Systems*, pages 260–263, 2010.

[79] Z. Sasanian and D. M. Miller. Mapping a multiple-control Toffoli gate cascade to an elementary quantum gate circuit. In *Proc. Workshop on Reversible Computation*, pages 83–90, 2010.

[80] Z. Sasanian and D. M. Miller. Mapping a multiple-control Toffoli gate cascade to an elementary quantum gate circuit. *J. of Multiple-valued Logic and Soft Computing*, 18(1):83–98, 2012.

[81] Z. Sasanian and D. M. Miller. A new methodology for optimizing quantum realizations of reversible circuits. 2012. In preparation.

[82] M. Soeken, Z. Sasanian, R. Wille, D. M. Miller, and R. Drechsler. Optimizing the mapping of reversible circuits to four-valued quantum gate circuits. In *Proc. Int'l Symp. on Multiple-valued Logic*, pages 173–178, 2012.

[83] D. Maslov, G. W. Dueck, and D. M. Miller. Fredkin/Toffoli templates for reversible logic synthesis. In *Proc. Int'l Conf. on CAD*, pages 256–261, 2003.

[84] D. Maslov, G. W. Dueck, D. M. Miller, and C. Negrevergne. Quantum circuit simplification and level compaction. *IEEE Trans. on CAD*, 27(3):436–444, 2008.

[85] R. Wille, D. M. Miller, and R. Drechsler. Reducing reversible circuit cost by adding lines. In *Proc. Int'l Symp. on Multiple-valued Logic*, pages 217 – 222, 2010.

[86] D. Maslov and M. Saeedi. Reversible circuit optimization via leaving the Boolean domain. *IEEE Trans. on CAD*, 30(6):806–816, 2011.

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

[88] A. Mishchenko and M. Perkowski. Fast heuristic minimization of exclusive sum-of-products. In *Proc. 5th Int'l Reed-Muller Workshop*, pages 242–250, 2001.

[89] J. E. Rice, K. B. Fazel, M. A. Thornton, and K. B. Kent. Toffoli gate cascade generation using ESOP minimization and QMDD-based swapping. In *Proc. Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pages 63–72, 2009.

[90] A. Gaidukov. Algorithm to derive minimum ESOP for 6-variable function. In *5th Int'l Workshop on Boolean Problems*, pages 141–148, 2002.

[91] N. Song and M. Perkowski. Minimization of exclusive sum of products expressions for multi-output multiple-valued input, incompletely specified functions. *IEEE Trans. on CAD*, 15:385–395, 1996.

[92] J. Zhong and J. C. Muzio. Using crosspoint faults in simplifying Toffoli networks. In *International IEEE Northeast Workshop on Circuits and Systems*, pages 129–132, 2006.

[93] M. Arabzadeh, M. Saeedi, and M. Saheb Zamani. Rule-based optimization of reversible circuits. In *Proc. ASP Design Automation Conf.*, pages 849–854, 2010.

[94] M. Karnaugh. The map method for synthesis of combinational logic circuits. *Trans. of A.I.E.E.*, 72(1):593–599, 1953.

[95] R. Wille, D. Große, D. M. Miller, and R. Drechsler. Equivalence checking of reversible circuits. In *Proc. Int'l Symp. on Multiple-valued Logic*, pages 324 – 330, 2009.

[96] S. Yamashita, S. Minato, and D. M. Miller. Synthesis of semi-classical quantum circuits. In *Proc. Workshop on Reversible Computation*, pages 93–99, 2010.

[97] Z. Sasanian and D. M. Miller. Reversible and quantum circuit optimization: A functional approach. In *Proc. Workshop on Reversible Computation*, pages 111–122, 2012.

[98] Z. Sasanian and D. M. Miller. *Reversible and Quantum Circuit Optimization: A Functional Approach*, volume 7581 of *Springer's Lecture Notes in Computer Science*. Springer Verlag, 2012. to appear.

[99] Z. Sasanian and D. M. Miller. NCV realization of MCT gates with mixed controls. In *Proc. Pacific Rim Conf. on Communications, Computers and Signal Processing*, pages 567–571, 2011.

[100] M. Soeken, R. Wille, C. Hilken, N. Przigoda, and R. Drechsler. Synthesis of reversible circuits with minimal lines for large functions. In *Proc. ASP Design Automation Conf.*, pages 85–92, 2012.

[101] D. P. DiVincenzo. The physical implementation of quantum computation. *Fortschr. Phys.*, 48:771–783, 2000.

[102] F. Schmidt-Kaler, H. Häffner, M. Riebe, S. Gulde, G. P. T. Lancaster, T. Deuschle, C. Becher, C. F. Roos, J. Eschner, and R. Blatt. Realization of the Cirac-Zoller controlled-NOT quantum gate. *Nature*, 422:408–411, 2003.

[103] A. I. Trifanov. Quantum gate implementation based on nonlinear optical phenomena. In *Proc. Int'l Conf. on Days on Diffraction*, pages 177–180, 2008.

[104] S. Salemian and S. Mohammadnejad. Quantum Hadamard gate implementation using planar lightwave circuit and photonic crystal structures. *American J. of Applied Sciences*, 5(9):1144–1148, 2008.

[105] D. M. Miller and Z. Sasanian. Improving the NCV realization of multiple-control Toffoli gates. In *Workshop on Boolean Problems*, pages 37–44, 2010.

[106] D. M. Miller, R. Wille, and Z. Sasanian. Elementary quantum gate realizations for multiple-control Toffolli gates. In *Proc. Int'l Symp. on Multiple-valued Logic*, pages 217–222, 2011.

[107] Z. Sasanian and D. M. Miller. Transforming MCT circuits to NCVW circuits. In *Proc. Workshop on Reversible Computation*, pages 163–174, 2011.

[108] Z. Sasanian and D. M. Miller. NCVW quantum gate realization of MCT gates. 2012. in preparation.

[109] Z. Sasanian and D. M. Miller. *Transforming MCT Circuits to NCVW Circuits*, volume 7165 of *Springer's Lecture Notes in Computer Science*. Springer Verlag, pages 77-88, 2011.

[110] Z. Sasanian, R. Wille, and D. M. Miller. Realizing reversible circuits using a new class of quantum gates. In *Proc. Design Automation Conf.*, pages 36–41, 2012.

[111] R. Wille. Private communication. 2011.

[112] M. Saeedi, M. Sedighi, and M. Saheb Zamani. A novel synthesis algorithm for reversible circuits. In *Proc. Int'l Conf. on CAD*, pages 65–68, 2007.

[113] N. M. Nayeem and J. E. Rice. Improving ESOP-based synthesis of reversible logic. In *Proc. Reed-Muller Workshop*, pages 57–62, 2011.

[114] A. Muthukrishnan and C. R. Stroud. Multivalued logic gates for quantum computation. *Physical Review A*, 62:052309, 2000.

[115] Y. Wang and M. Perkowski. Improved complexity of quantum oracles for ternary grover algorithm for graph coloring. In *Proc. Int'l Symp. on Multiple-valued Logic*, pages 294 – 301, 2011.

[116] K. Fazel, M. Thornton, and J. E. Rice. ESOP-based Toffoli gate cascade generation. In *Proc. Pacific Rim Conf. on Communications, Computers and Signal Processing*, pages 206–209, 2007.

[117] S. J. Devitt A. G. Fowler and L. C. L. Hollenberg. Implementation of Shor's algorithm on a linear nearest neighbour qubit array. *Quantum Information and Computation*, 4(4):237–245, 2004.

[118] P. Pham and K. M. Svore. A 2D nearest neighbor quantum architecture for factoring. In *Proc. Workshop on Reversible Computation*, pages 158–170, 2012.

[119] M. H. A. Khan. Cost reduction in nearest neighbour based synthesis of quantum Boolean circuits. *Engineering Letters*, 16:1–5, 2008.

[120] D. Maslov. Reversible logic synthesis benchmarks page. http://www.cs.uvic.ca/~dmaslov/, 2009.