**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY**

# QUANTUM CIRCUIT SYNTHESIS

**M.Sc. THESIS**

**Ömer Can SUSAM**

**Department of Nanoscience and Nanoengineering**

**Nanoscience and Nanoengineering Programme**

**JANUARY 2015**

**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE ENGINEERING AND TECHNOLOGY**

# QUANTUM CIRCUIT SYNTHESIS

**M.Sc. THESIS**

**Ömer Can SUSAM**
**(513121011)**

**Department of Nanoscience and Nanoengineering**

**Nanoscience and Nanoengineering Programme**

**Thesis Advisor: Asst. Prof. Mustafa ALTUN**

**JANUARY 2015**

**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

# KUANTUM DEVRE SENTEZİ

**YÜKSEK LİSANS TEZİ**

**Ömer Can SUSAM**
**(513121011)**

**Nanobilim ve Nanomühendislik Anabilim Dalı**

**Nanobilim ve Nanomühendislik Programı**

**Tez Danışmanı: Yrd. Doç. Mustafa ALTUN**

**OCAK 2015**

**Ömer Can Susam**, a **M.Sc.** student of ITU **Institute of / Graduate School of Science Engineering and Technology** student ID 513121011, successfully defended the **thesis** entitled "**QUANTUM CIRCUIT SYNTHESIS**", which he prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

| | | |
|---|---|---|
| **Thesis Advisor :** | **Asst. Prof. Mustafa ALTUN**<br>İstanbul Technical University | ..... |
| **Jury Members :** | **Prof. Dr. Hilmi ÜNLÜ**<br>İstanbul Technical University | ..... |
| | **Asst. Prof. Faik Başkaya**<br>Boğaziçi University | ..... |

**Date of Submission : 15 December 2014**
**Date of Defense : 20 January 2015**

***To my family,***

**FOREWORD**

I owe a great debt of gratitude to my supervisor Asst. Prof. Mustafa Altun for his inspiration and guidance.
I would like to thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) (2210-C) and Scientific Research Projects Agency of Istanbul Technical University (BAP).
Finally, I would like to thank my parents, my brother and my friends for their great support.

December 2014

Ömer Can SUSAM
(Electrical Engineer)

**TABLE OF CONTENTS**

**Page**

**ABBREVIATIONS**

**App** **:** Appendix
**C-NOT** **:** Controlled Not
**CNT** **:** C-Not, Not, Toffoli
**NCL** **:** Negative Control Lines
**NCV** **:** Not, C-Not, V gate
**OPT** **:** Optimum
**PN-CNT** **:** Positively and Negatively Controlled CNOT, Not, Toffoli
**PS** **:** Pick Smallest
**RC** **:** Reversible Cost
**QC** **:** Quantum Cost
**TB** **:** Top to Bottom

**LIST OF TABLES**

**Page**

**LIST OF FIGURES**

**Page**

## QUANTUM CIRCUIT SYNTHESIS

## SUMMARY

This thesis presents a new approach for the synthesis of quantum circuits. Quantum computers, more specifically quantum algorithms, take on the eyes with their computational promises. They paved the way for calculation of the complex problems that can't be solved in polynomial time by traditional counterparts.

Exploiting quantum mechanical phenomena and its features is the main power source of the quantum computing idea. This is also leaning on the concept that accepts information as a physical item. In addition, quantum mechanical unitarity brings reversibility for quantum algorithms and quantum circuits. This creates ideal application area for reversible computing and reversible circuit design. Reversible computing is motivating scientist and researchers for years to achieve energy efficient computation.

With the help of advancing technology, quantum computation become applicable. At this point, reversible quantum circuit design is coming forward that is the core of this computation. When compared with classical computation methods, quantum systems are very sensitive. This is one of the main reasons to synthesize these circuits minimally as possible. Depending on quantum computation method, each gate in quantum circuits corresponding to one or more pulse operations. Therefore, optimized circuits will improve both the security and the run time of the computation. Still, optimal circuit synthesis for higher bit count is hard to achieve, it can take very long time which is not practical at all. To compete with classical computation in a realistic manner, this is one of the main obstacles to overcome and it constitutes the main motivation why we aim at a fast synthesis algorithm in this thesis.

To fulfill the needs of this upcoming technology, we perform synthesis and optimization of quantum circuits in two main parts. In the first part, we propose a fast synthesis algorithm that implements any given reversible Boolean function with quantum gates. Instead of an exhaustive search on every given function, our algorithm creates a library of essential functions and performs sorting to obtain desired function. In the second part, we optimize our circuits by using templates. The proposed templates mainly consist of identical neighbor gates and Toffoli gates, realized with V, $V^{\dagger}$ and CNOT gates. We also improve the CNT library by taking negative control lines into account which provides important circuit cost reduction. We call this new library as PN-CNT that stands for “Positively and Negatively Controlled CNOT, Not, Toffoli”.

Quantum computers implemented in various ways so far. Each implementation has its own physical cost. For the calculation of quantum circuit cost, we preferred to use widely accepted NCV-111 cost metric in this thesis. Finally we compared our results with studies in the literature.

# KUANTUM DEVRE SENTEZİ

## ÖZET

Bu tez, tersinir devre sentezine dayalı kuantum devrelerin sentezlenmesi için yeni bir yöntem sunmaktadır. Kuantum bilgisayarlar, özellikle kuantum devrelerle oluşturulan kauntum algoritmalar, hesaplamalı alanda vaat ettikleriyle son yıllarda dikkatleri üzerine çekti. Bu bilgisayarlar, klasik bilgisayarların çözemediği kompleks problemleri hesaplamanın önünü açmaktadır.

Geleneksel olarak bilgisayarlar 0 veya 1 değeri alabilen bitler ile hesaplamaları gerçekleştirirler. Bitler ile hesaplama genel anlamda oldukça etkin ve verimli olmasına rağmen bazı önemli problemlerin çözümünde yetersiz kalmaktadır. Bunun en önemli nedeni ise bitlerin deterministik olarak çalışması, yani belirli bir zaman aralığında sadece 0 veya 1 değeri alabilmesidir. Richard Feynman tarafından önerilen kuantum bilgisayar fikri, kuantum mekaniğinden faydalanarak hesaplama işlemlerini gerçekleştirme mantığına dayanmaktadır. Bu hesaplama yönteminde, veri saklama elemanları "kübit" olarak adlandırılır. Kuantum mekaniğinin süperpozisyon prensibi gereği, kübitler 0 veya 1 aynı anda hem 0 hem 1 konumunda bulunabilir. Bir başka deyişle bir kübitin değeri, 0 veya 1 olma olasılığını belirtir. Böylece, pratik limitler dahilinde olanaksız olan bir çok problem, kuantum algoritmaları ile rahatlıkla çözebilmektedir. Bu problemlerin belki de en ünlüsü kriptolojide yaygın kullanılan yarı-asal sayıların çarpanlarına ayrılmasıdır. Shor'un kübit tabanlı çarpanlara ayırma algoritması, geleneksel anlamda çözümü yüzyıllar süren durumları, hızlıca çözmektedir.

Kuantum devreler, kübitler üzerinde işlem yapan kuantum kapılar kullanılarak inşa edilir. Kuantum devrelerin en önemli özelliklerinden birisi aynı zamanda tersinir (reversible) devreler olmalarıdır. Bu devrelerde, devrenin girişindeki ve çıkışındaki bit sayısı eşittir. Devrenin çıkışındaki değerler bize devrenin girişindeki değerler hakkında bilgi verir. Bu sayede devreleri çift yönlü olarak kullanabilmemize imkan sağlarlar. Tersinir devrelerin en büyük getirisi, bilgisayarlarda yüksek enerji tasarrufuna olanak sağlamalarıdır.

Kuantum bilgisayarlar ile ilgili yapılan deneysel uygulamalar her ne kadar emekleme aşamasında olsa da, kuantum hesaplamanın uygulanabilir olduğunu göstermektedir. Bu noktada kuantum hesaplamanın en önemli bölümü olan kuantum devre tasarımı ön plana çıkmaktadır. Çalışmalar en az sayıda kapı kullanarak optimum devreleri sentezlemeyi amaçlamaktadır. Optimize edilmiş devreler, bir yandan güvenilirliği artırırken, diğer bir yandan çalışma süresini düşürmektedir. Şuana kadar yapılan çalışmalarda, optimum devre sentezi sadece 4 bite kadar gerçekleştirildi. Son yapılan ve 84 kübit kullanılan deneyler göz önüne alındığında, optimal devre sentezi pratik olmaktan oldukça uzak kalmaktadır. Bu, çalışmamızı hızlı sentezleme algoritmalarına yönelten motivasyonların başında gelmektedir.

Literatürdeki birkaç çalışmada, özellikle yüksek bit sayısı ile sentezleme

yapılanlarda, “garbage output” birimleri kullanılmıştır. Bu birimler, ihtiyaç olduğunda kullanılmak üzere devrede fazladan bulundurulan veri yolları olarak düşünülebilir. Bir çok yöntemde, enerji verimililiği ile ilgili problemlerinden ötürü kullanımı düşünülmemiştir. Bu nedenle çalışmamızda “garbage ouput” kullanmadık.

Önerdiğimiz yöntem, istenilen fonksiyonu kuantum kapıları kullanarak verimli bir şekilde elde etmekte ve bunu iki ana aşamada yapmaktadır. İlk aşamada, permütasyona dayalı bir algoritma ile seçilen bit büyüklüğüne göre optimum sayıda kapı kullanılan “temel fonksiyonlar” sentezlenmektedir. Her temel fonksiyon için, en az kapı sayısından başlayıp, tüm kapıları ve permütasyonlarını deneyerek, fonksiyonu gerçekleştiren devreyi temel fonksiyonlar kütüphanesine ekler ve bir sonraki temel fonksiyonu aramaya başlar. Kütüphane tamamlandıktan sonra sıralama aşamasına geçilmektedir. Algoritmanın bu aşaması, en çok vakit alan kısım olmasına rağmen, temel fonksiyonların sayısının azlığı ve sıralama algoritmasının hızı, bu yöntemi literatürdeki çalışmalardan oldukça hızlı kılmaktadır. Optimum çözüm üretmeyen yöntemlerle bu devreler çok daha hızlı bir şekilde elde edilebilir ancak devre maliyetleri çok daha yüksek olacaktır. Temel fonksiyonların maliyetleri, oluşturulacak olan tüm devrelerin maliyetini etkileyeceğinden, devreleri optimum olarak sentezlemeyi tercih ettik.

Devre sentezleme aşaması, istenilen fonksiyonu elde etmemizi sağlayan sıralama süreci ile devam eder. Bu süreç, algoritmamıza hızını kazandıran, yöntemimizin en önemli bölümdür. Sıralama algoritmaları, matematik ve bilgisayar biliminde uzun süredir çalışılan bir konu olduğundan, birçok farklı sıralama algoritması geliştirilmiştir. Biz çalışmamızda, “Seçmeli Sıralama” algoritmasını kullandık. Bu sıralama yöntemi, verilen fonksiyonu, doğruluk tablosu ile karşılaştırarak satır satır kontrol edip, eşleşmeyen her durum için temel fonksiyonlardan birini kullanarak, fonksiyonu adım adım birim fonksiyona çevirmektedir. Diğer sıralama algoritmalarının aksine kaydırma veya bölme işlemlerini uygulamadığından, fazladan temel fonksiyon kullanımını önleyerek, devre maliyetini düşük tutmaktadır. Örneğin, birleştirmeli sıralama, verilen sıralama kümesini öncelikle alt kümelere ayırıp, bu alt kümeleri sıralamaktadır. Ardından, oluşturulan alt kümeler, parça parça birleştirilerek her yeni birleşmede yeni bir sıralama yapılmaktadır. Sıralamalardaki yer değiştirme işlemlerinin her biri ek bir temel fonksiyon kullanımına neden olmaktadır. Aynı şekilde, eklemeli sıralama algoritmasında kullanılan kaydırma işlemlerinin her biri, bir temel fonksiyona karşılık gelmektedir. Seçmeli algoritma ile oluşturulan devrelerin maliyeti, yerdeğiştirilecek olan satırların değiştirilme sırasının, doğru bir şekilde belirlenmesiyle iyileştirilebileceğini gösterdik. Bu amaçla, çalışmamıza her fonksiyon için optimum sıralamayı bulan ek bir bölüm ekledik. Eklediğimiz bu kısım bazı devrelerin maliyetini azaltırken, programın çalışma süresini artırmıştır.

İkinci aşama, oluşturduğumuz şablonları kullanarak, sentezlenen devrelerde optimizasyon yapmaktadır. Şablonlar, aynı fonksiyonu daha az sayıda kapıyla gerçekleyen ve devredeki eşdeğeri ile değiştirilerek toplam kapı sayısında düşüş sağlayan devrelerdir. Şablonlarımızı iki farklı yolla oluşturduk. Birincisi, tersinir kapı kütüphanemizi kullanarak. İkincisi de bu kütüphanedeki kapıların içlerinde bulunan kuantum kapıları göz önüne alarak. Birinci yöntem, sıralama algoritmasının uygulanmasından sonra, aynı iki kapının yanyana gelebileceği göz önünde bulundurularak üretilmiştir. İkinci türdeki şablonlarda, sentezlediğimiz devrelerde sıkça kullandığımız Toffoli kapısının, kuantum kapılarla (V, $V^{\dagger}$ ve CNOT) kaç farklı şekilde gerçekleştirilebileceğini inceledik. Bu aşamada, kompleks sayılardan oluşan

matrisleri kullanacağımız için, MATLAB programını kullandık. Devre içiersinde Toffoli kapısının yanına gelen CNOT kapılarından bir kısmının optimizasyon için kullanılabileceğini gösterdik.

Ayrıca çalışmamızda, pozitif kontrollü kapılara (CNT) ek olarak negatif kontrollülerinde sentezleme aşamasına eklenmesiyle devre maliyetlerinde önemli ölçüde iyileştirmeler elde ettik.

Kuantum hesaplama, deneysel olarak bir çok farklı şekilde gerçekleştirilmiştir. Her gerçeklemenin, kendine özgü prensipleri ve özellikleri olduğundan, algoritmalardaki kapıların uygulanış biçimi de farklı olmaktadır. Bu nedenle, her yöntem için ayrı kapı maliyetleri oluşmaktır. Çalışmamızki kuantum devrelerin maliyetlerini literatürde yaygın olarak kullanılan NCV-111 maliyet metriğini kullanarak hesapladık. Son olarak, yöntemimizi literatürde bulunan çalışmalar ile kıyasladık.

# 1. INTRODUCTION

With experimental realization of quantum computation, quantum circuit synthesis and optimization methods come into prominence. There is few reasons to synthesise these circuits optimally. Quantum systems are very sensitive when compared with our classical computation methods. Since each gate in quantum circuits corresponding to one or more pulse operations depending on quantum computation method, optimized circuits will boost the security on one hand and decrease the run time of the computation on the other hand. Nevertheless, optimal circuit synthesis for higher bit count still an issue. To compete with classical computation in a realistic manner, this is one of the main obstacles to overcome and it constitutes the main motivation why we aim at a fast synthesis algorithm in this study.

In this thesis, a new quantum circuit synthesis approach is presented which works with deterministic input and output values for a function. The thesis is organized as follows. In Second Chapter, background information for quantum computation introduced. In Third Chapter, reversible computation, including reversible circuits is explained. Chapter Four presents our synthesis algorithm by explaining essential functions, sorting section and optimization method based on template matching mechanism. In Chapter Five and Six, experimental results and conclusions presented respectively.

## 2. QUANTUM COMPUTING

Quantum computation is a concept that is based on the idea of using quantum mechanical phenomena and its features to make computation. It is first proposed by Richard Feynman at 1982 [1]. Quantum computation takes its power with promising to out-perform the calculation of certain problems when compared with classical computation. This is theoretically proved with few algorithms. One of them is Shor's factoring algorithm [2]. It shows that, quantum computers can easily factorize a semi-prime number which is actually leading to overcome our current security systems called RSA [3]. The other one is Grover's search algorithm, illustrates faster searching of database than any other classical algorithms [4].

On the other hand, according to Moore's law, integrated circuit performance will be doubled every 18 months [5]. This performance improvement is basically leaning on the transistor number in a single chip, and so far this is accomplished by shrinking transistors. Since the ultimate limit of this shrinkage is corresponding to the size of an atom [6], quantum computation provides a way out for this predicament. Although experimental applications are still in crawling stage, studies show that quantum computation is being applicable [7].

### 2.1 Building Blocks of Quantum Computation : Qubits

Bit is the basic unit of information that can only have one of two states, 0 or 1. In quantum information, this basic unit called as qubit (abbreviation of quantum bit). Additionally to 0 and 1 states, a qubit can also exist as a linear combination of the states $|0\rangle$ and $|1\rangle$ which is known as superposition state. General representation of this state is written as in Equation **(2.1).** α and β are complex numbers, that satisfies the normalization condition **(2.2)**

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad \textbf{(2.1)}$$

$$|\alpha|^2 + |\beta|^2 = 1 \quad \textbf{(2.2)}$$

The explanation of this concept as follows, measurement of the qubit $\psi$ can be resulted as state $|0\rangle$ with probability $|\alpha|^2$, and can be resulted as state $|1\rangle$ with probability $|\beta|^2$.

State of $\psi$ is a vector, in two-dimensional complex vector space. It is written in a vector notation as **(2.3)**

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.3}$$

As we live in our perceptional world in a deterministic way, it becomes harder to understand quantum mechanics probabilistic structure. To understand this phenomena in a better way lets give an example. A coin, can be considered as classical bit and every flip will have only one result, heads or tails. For the same example, before pulling your hand over the coin, the probability of having tails and heads as a result is 50% as shown **(2.4)**

$$|coin\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{2.4}$$

When result is seen (in other words, measurement is performed), qubit will collapse to measured basis state, tails or heads.

Realization of qubits may occur in different ways such as, polarizations of a photon in two different state, nuclear spin alignment in a uniform magnetic field, electron orbiting states of a single atom (Figure 2.1). For example, electron can exist in either ground '$|0\rangle$' or excited '$|1\rangle$'states, in the atom model. When light beam hits the atom with needed energy and time length, electron state can be changed from $|0\rangle$ to $|1\rangle$ and vice versa. The most important part is, if the time of beaming decreased, it is possible to move electron with $|0\rangle$ initial state, to the state between $|0\rangle$ and $|1\rangle$ which is showed as $|+\rangle$ state.

When Equation **(2.1)** and Equation **(2.2)** combined, $\psi$ can be written as **(2.5)**

$$|\psi\rangle = e^{i\gamma}\left(cos\frac{\theta}{2}|0\rangle + e^{i\varphi}sin\frac{\theta}{2}|1\rangle\right) \tag{2.5}$$

The pure state of a qubit is geometrically represented with Bloch sphere (Figure 2.2).

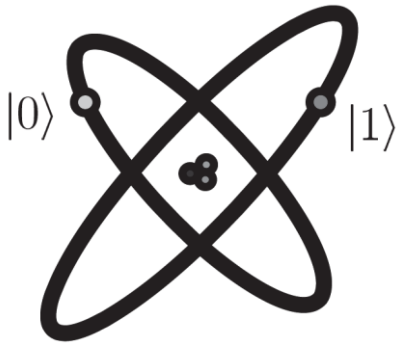



**Figure 2.1 :** Representation of a qubit in an atom with two electron levels.

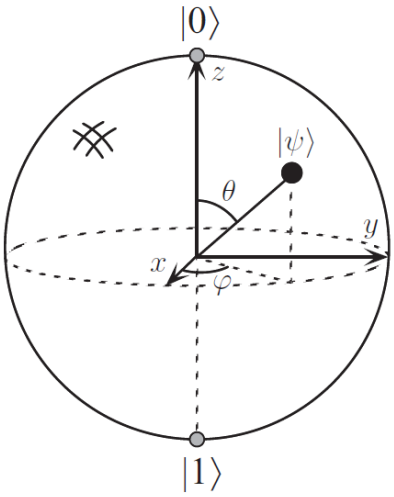



**Figure 2.2 :** Geometric representation of a qubit, Bloch sphere.

## 2.2 Quantum Gates

In classical computation, circuits constructed with wires that carries information around the circuit and logic gates which are used to manipulate this information. NOT gate is the easiest way to explain this situation, it operates 0 → 1 or 1 → 0 depending on its input, which interchanges between 0 and 1 states as can be seen. Same gate used in quantum computation, can be represented with 2x2 matrix as follows **(2.6)**

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \textbf{(2.6)}$$

As we defined in classical computation, quantum not gate operates in the same way, but in this concept, it interchanges the probability of $|0\rangle$ and $|1\rangle$ states between each other **(2.7)**.

$$\sigma_x |\psi\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \beta|0\rangle + \alpha|1\rangle \quad \textbf{(2.7)}$$

Figure 2.3 shows classical NOT gate and quantum NOT gate respectively operating on a bit and qubit.

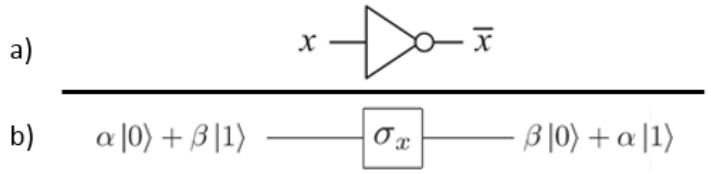



**Figure 2.3 :** (a) Not operation on classical bit and (b) qubit.

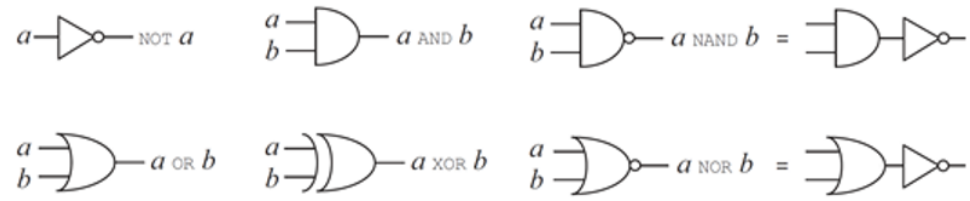
Also, multiple qubit operations can be realized in quantum computation. To understand multiple qubits and gates, here is an example with two qubits. In classical computation, with two bits we would have four possible states denoted as 00, 01, 10 and 11. In quantum computation these four states are called as "computational basis states" and denotation as follows $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ **(2.8)**. The most important part is, qubits can exist in superpositions of these states which gives its power to quantum computation.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \quad \textbf{(2.8)}$$

In contrast to classical computation gates (Figure 2.4 (a)), input and output numbers of quantum gates are always same (Figure 2.4 (b)). This difference can be seen in Figure 2.4. The unitary operation of $U(2^n)$ group is carried by the gate with $n$ inputs. Controlled-NOT or CNOT gate one of the essential gates in quantum computation. It has 2 inputs thus it carries $U(4)$ group unitary operation. One of its inputs called as control qubit and the other one is target qubit. CNOT gate represented in Figure 2.4 (b), control qubit and target qubit represented respectively on the top line and bottom line.

The definition of CNOT gate is similar with classical XOR gate, additional modulo two. The operation CNOT perform can be summarized as $|a, b\rangle \rightarrow |a, b \oplus \text{a}\rangle$. Result is stored in target qubit. Table 2.1 shows inputs and outputs for this gate. Matrix representation of this operation shown in Equation **(2.9)**.

a)




b)

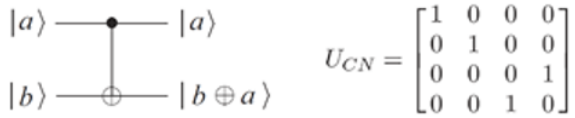



**Figure 2.4 :** (a) Classical gates, (b) quantum CNOT gate and its matrix representation.

**Table 2.1 :** CNOT gate input and output results.

| CNOT | |
|---|---|
| Input $\|a, b\rangle$ | Output $\|a, b \oplus \mathrm{a}\rangle$ |
| $\|00\rangle$ | $\|00\rangle$ |
| $\|01\rangle$ | $\|01\rangle$ |
| $\|10\rangle$ | $\|11\rangle$ |
| $\|11\rangle$ | $\|10\rangle$ |

$$U_{CN}\,|\psi\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_{00}|00\rangle \\ \alpha_{01}|01\rangle \\ \alpha_{10}|10\rangle \\ \alpha_{11}|11\rangle \end{bmatrix} = \begin{bmatrix} \alpha_{00}|00\rangle \\ \alpha_{01}|01\rangle \\ \alpha_{11}|11\rangle \\ \alpha_{10}|10\rangle \end{bmatrix} \tag{2.9}$$

Quantum computers are capable to simulate every classical computation with using Toffoli gate [8] [10]. Toffoli is similar to CNOT gate except it has one additional control bit. If both of the control bits are set to 1, target bit is flipped. Here is the summarized version of this operation $|a, b, c\rangle \rightarrow\ |a, b, c \oplus \mathrm{ab}\rangle$.

Simulation of all the classical gates can be realized by using Toffoli gate, in Figure 2.5(c) one of them is shown. Moreover, Toffoli can be used to do FANOUT. In the meaning of classical computation, FANOUT operation corresponds to creating copies of a bit from single wire which is forbidden by quantum mechanics in quantum computation [8]. On the other hand, Toffoli gate realizes this banned operation with initially set control and target bit, illustrated in Figure 2.5(b). Table 2.2 shows truth table of the Toffoli operation.

**Table 2.2 :** Toffoli gate input and output results.

| Toffoli | | | | | |
|---|---|---|---|---|---|
| Input | | | Output | | |
| $a$ | $b$ | $c$ | $a'$ | $b'$ | $c'$ |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

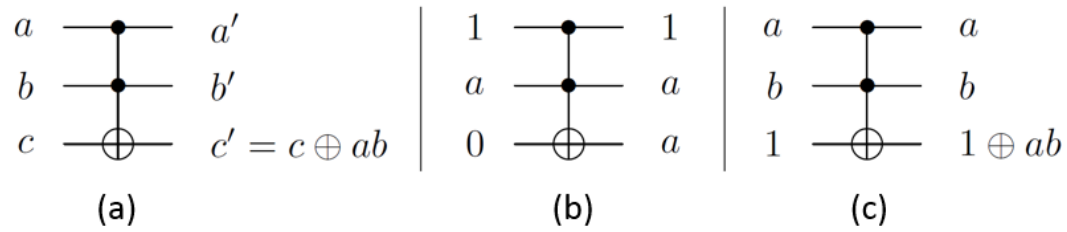



**Figure 2.5 :** (a) Representation of Toffoli gate, (b) FANOUT operation with Toffoli gate, (c) NAND gate simulated with Toffoli.

### 2.3 Quantum Circuits

Quantum circuits are constructed with quantum gates in a sequence. Each line in this circuit represents a qubit and it may correspond to a physical particle like a photon. Therefore, line number corresponds to qubit number of the computation. Quantum circuits (for example Figure 2.6) are read from left side to the right side. Since they are reversible, previous states of the qubits can be found by reading to backwards.

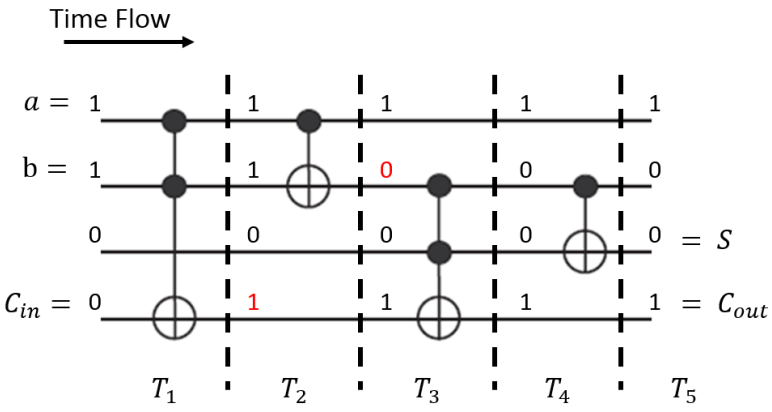



**Figure 2.6 :** Quantum Circuit example, Full-Adder constructed with CNOT and Toffoli gates, time flows left to right.

Figure 2.6 represents a quantum circuit. The circuit realizes a Full-Adder with using Toffoli and CNOT gates. Quantum circuits can be considered as a model for quantum computation to design quantum algorithms.

## 2.4 Quantum Algorithms

Classical computers are insufficient to solve some computational problems. Quantum computers can provide solution for these type of problems. Quantum algorithms are instruction sequences performed by quantum computers. These algorithms are generally exploits some features of quantum mechanics such as superposition and entanglement.

So far few examples of quantum algorithms has shown. Shor's factoring algorithm one of these. It is exponentially faster than any classical algorithm that performs same factoring calculation. The other one is Grover's searching algorithm for searching an unsorted database. It is quadratically faster when compared with classical algorithms.

Quantum algorithms includes also classical functions to realize their task. Therefore, it is important to synthesize these functions in quantum computational form. This is one of the aims of this thesis.

## 2.5 Construction of the Gates

Toffoli gate is one of the essential gates to realize computations in quantum computing as we described before. But the implementation of it requires few basic operations. To understand how it is constructed we should examine how elementary operations are build.

Matrix representations of quantum gates described with $U$, which stands for unitary matrix. This is explained as, if multiplication of the matrix and its adjoint (transpose of complex conjugation of the matrix **(2.10)**) equals to identity matrix then it is unitary **(2.11)**. Any unitary matrix can be used as a quantum gate [8].

$$U^{\dagger} = (\overline{U})^{T} \tag{2.10}$$

$$U^{\dagger}U = UU^{\dagger} = I \tag{2.11}$$

Each $2x2$ unitary matrix can be shown in the form as in the Equation **(2.12)** [9], where $\delta, \alpha, \theta$ and $\beta$ are real-valued,

$$\begin{bmatrix} e^{i(\delta+\frac{\alpha}{2}+\frac{\beta}{2})}\cos\frac{\theta}{2} & e^{i(\delta+\frac{\alpha}{2}-\frac{\beta}{2})}\sin\frac{\theta}{2} \\ -e^{i(\delta-\frac{\alpha}{2}+\frac{\beta}{2})}\cos\frac{\theta}{2} & e^{i(\delta-\frac{\alpha}{2}-\frac{\beta}{2})}\cos\frac{\theta}{2} \end{bmatrix} \quad \textbf{(2.12)}$$

Every $2x2$ unitary matrix can also be expressed as follows

$$\begin{bmatrix} e^{i\delta} & 0 \\ 0 & e^{i\delta} \end{bmatrix}\begin{bmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{bmatrix}\begin{bmatrix} cos\theta/2 & sin\theta/2 \\ -\sin\theta/2 & cos\theta/2 \end{bmatrix}\begin{bmatrix} e^{i\beta/2} & 0 \\ 0 & e^{-i\beta/2} \end{bmatrix} \quad \textbf{(2.13)}$$

Here is the simplified representation of these matrices respectively,

$$U = Ph(\delta)\, R_z(\alpha)R_y(\theta)R_z(\beta) \quad \textbf{(2.14)}$$

Table 2.3 shows gates that can be used to obtain every unitary matrix, which is the reason to call them as Elementary Gates. They can be obtained by giving certain values to the $\delta, \alpha, \theta$ and $\beta$ variables in the matrix in **(2.10)**.

**Table 2.3 :** Elementary Gates.

| Representation | Gate Matrix | Values of, $\delta$ | $\alpha$ | $\theta$ | $\beta$ |
|---|---|---|---|---|---|
| $R_y(\theta)$ | $\begin{bmatrix} cos\theta/2 & sin\theta/2 \\ -\sin\theta/2 & cos\theta/2 \end{bmatrix}$ | 0 | 0 | $\theta$ | 0 |
| $R_z(\alpha)$ | $\begin{bmatrix} e^{i\alpha/2} & 0 \\ 0 & e^{-i\alpha/2} \end{bmatrix}$ | 0 | $\alpha$ | 0 | 0 |
| $Ph(\delta)$ | $\begin{bmatrix} e^{i\delta} & 0 \\ 0 & e^{i\delta} \end{bmatrix}$ | $\delta$ | 0 | 0 | 0 |
| $\sigma_x$ | $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ | 0 | 0 | π | 0 |
| I | $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ | 0 | 0 | 0 | 0 |

To implement all of the unitary matrices as a quantum gate, $U = A\sigma_x B\sigma_x C$ gate combination can be used where, $A, B, C$ are single qubit gates that provides the equation $ABC = I$ [8] [9] (Figure 2.7).

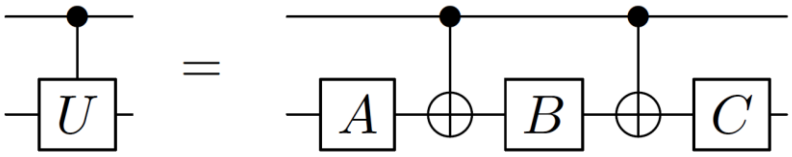



**Figure 2.7 :** Controlled U gate implementation with quantum gates.

In Figure 2.7, if control bit is set as $|1\rangle$ then $U = A\sigma_x B\sigma_x C$ operation is applied on the second qubit, on the contrary if control bit is set as $|0\rangle$, $ABC = I$ operation is applied on the second qubit, which is resulted without any change.

Quantum algorithm development process looks similar with low-level programming language in computer science, with elementary gates. In contrast, high level programming language is easier to use and may automate some parts of the computing system that makes developing process simpler and understandable. To increase the simplicity of quantum algorithm development process, more understandable and functional gates are needed then elementary gates. For this reason, highly used operations in quantum algorithms, in other words common gates, can be considered as higher-level versions of elementary gates, which are listed in Table 2.4.

**Table 2.4 :** Common Gates.

| Gate Name | Representation | Gate Matrix |
|---|---|---|
| Hadamard | $H$ | $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ |
| Pauli-X | $X$ | $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ |
| Pauli-Y | $Y$ | $\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$ |
| Pauli-Z | $Z$ | $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ |
| Phase | $S$ | $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ |
| $\pi/8$ | $T$ | $\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$ |

As we explained the implementation of controlled unitary matrices before, implementation of the multiple qubit gates with more than two qubits has similar realization as it shown in Figure 2.8.

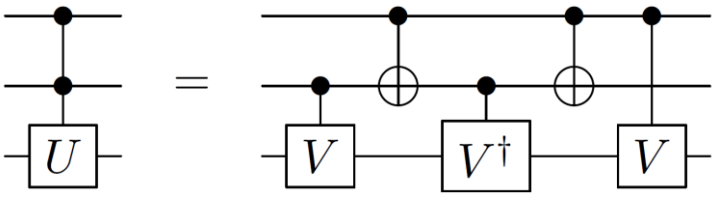



**Figure 2.8 :** Controlled-Controlled U gate with V and CNOT gates.

To realize Toffoli gate with V and CNOT gates, where $U = V^2 = \sigma_x$ , following value must be given to V gate,

$$V = (1-i)(I-iX)/2 = \begin{bmatrix} 0.5+0.5i & 0.5-0.5i \\ 0.5-0.5i & 0.5+0.5i \end{bmatrix} \quad \textbf{(2.15)}$$

Given value can be obtained by using common gates Phase (S) and Hadamard (H) as shown in Figure 2.9 and Equation **(2.16)**.

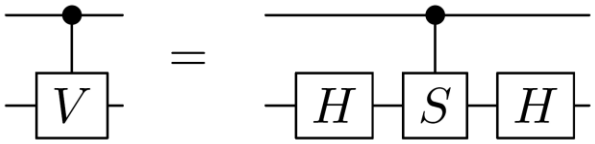



**Figure 2.9 :** Controlled-Controlled U gate with V and CNOT gates.

There is two possible condition for this V gate, control bit can either be 1 and 0. If control bit is set to 1:

$$HSH = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix} \quad \textbf{(2.16)}$$

If control bit is set to 0:

$$HH = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \textbf{(2.17)}$$

As a result V operation successfully realized with using Phase and Hadamard gate combination.

### 2.6 Cost Metric

Since each realization has its own methods to perform same operation, it is hard to synthesize quantum circuits optimally for each of these techniques. On the other hand, few cost metrics are suggested for high-level quantum gate operations [16]. In

this thesis, we consider these suggestions and selected widely used cost metric NCV-111 [17]. In this cost metric, each NOT, CNOT and V operations are costed with 1 as shown in Figure 2.10.

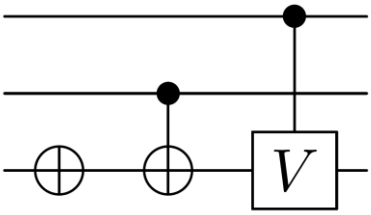



**Figure 2.10 :** NCV-111 cost metric, each of these gates are costed as 1.

## 2.7 Realization of Quantum Computation

It is very challenging to physically realize quantum computation. Qubit representation and preparation process not the only issue. There is also another problems such as close quantum system time-evolution which is described by unitary operator determined with its Hamiltonian. Hamiltonians of the system must be controlled to perform a quantum operation. Every quantum system has different Hamiltonians with different physical machine description and its applicability changable on one system to another. Which means, one quantum gate operation may simply applicable in one system but it may be difficult to apply same operation in other systems. Few of these physical realizations are listed below:

- **Optical Photon Quantum Computer:** Optical photons can be represented as qubits. They can be prepared to interact with each other in principle [8] [11] .
- **Ion Trap:** Confined Ions are used to represent qubits which is included in 2D lattice and can be moved within this lattice to serve local interactions [8] [12].
- **Nuclear Magnetic Resonance:** Nuclear spin states manipulation and detection is possible with using radiofrequency electromagnetic waves [8] [13].
- **Quantum Dot:** Spin states of two electrons within quantum dots are represented as quibts. It has a tunneling barrier between its two potential wells [14].

Many different methods are suggested for the implementation of quantum computation [15].

## 3. REVERSIBLE COMPUTING

Bijection functions in mathematics is a great example to understand reversibility in computing. In these functions, input and output sets have the same number of elements and each element has only one counterpart in other set (Figure 3.1). This means, the input value can be deduce by looking at the output value of a reversible function (Figure 3.2). This is not the only feature of these functions.

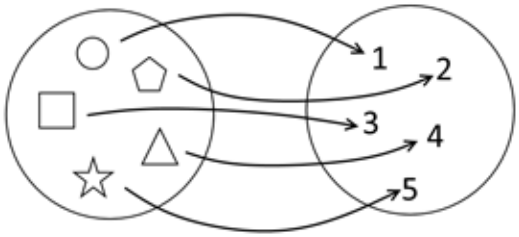



**Figure 3.1 :** Bijection Function.

Conventional computers are working with irreversible operations (Figure 3.3). Theoretical lower energy dissipation limit of an irreversible computation is calculated by Rolf Landauer [18]. It is suggested, if the computation is reversible, then computation can be performed nearly without dissipating energy [18] [19] [20].

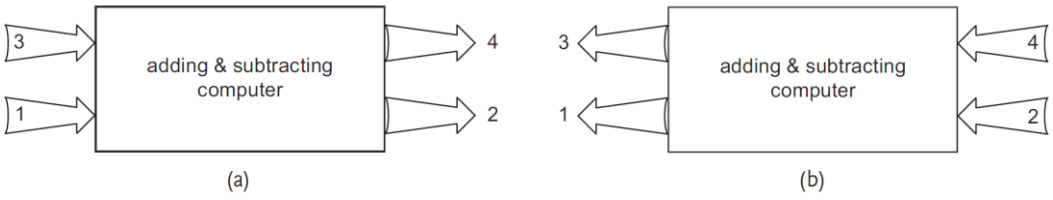



**Figure 3.2 :** Reversible logical operation.

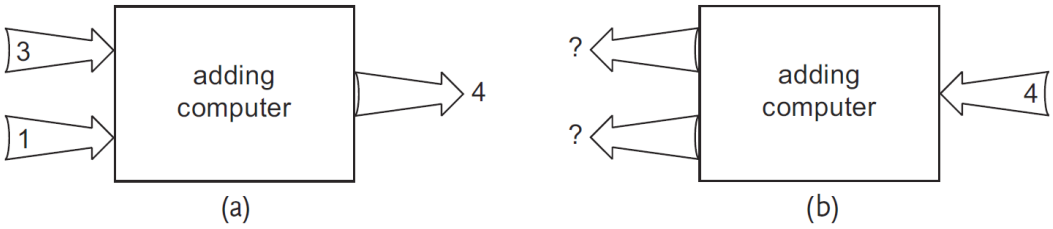



**Figure 3.3 :** Irreversible logical operation.

The main problem in irreversible computation is every logical delete operation on a bits information is leading an energy dissipation into the environment. The theoretical limit of this dissipation is calculated by the formula $k_b T ln2$ where $k_b$ is the Botlzman constant, T is the temperature of the circuit. When this limit compared with today's computers, it can be seen that, modern CPUs are far away from this limit [21].

On the other hand, instead of irreversible computation, reversibility can used to make energy efficient devices to achieve reversible computation. Last experimental approaches are promising for this purpose [22].

Examples of reversible operations with truth table shown in Table 3.1.

**Table 3.1 :** Truth tables for bit size 2: (a) identity function, (b) arbitrary function.

| a b | a′b′ |
|---|---|
| 0 0 | 0 0 |
| 0 1 | 0 1 |
| 1 0 | 1 0 |
| 1 1 | 1 1 |

(a)

| a b | a′b′ |
|---|---|
| 0 0 | 1 1 |
| 0 1 | 0 1 |
| 1 0 | 1 0 |
| 1 1 | 0 0 |

(b)

### 3.1 Reversible Circuits

Reversible logic circuits form a group for a specific bit size $n$ (Figure 3.4). Row number for this specific bit size $n$ equals to $2^n$. As the function is reversible, all of the input values must be on the output side as well. Which means, a reversible function can be form a permutation of these input values. Since there is $2^n$ inputs, all possible reversible functions number is equals to $2^n!$. Table 3.2 showing total reversible function numbers for a specific bit size.

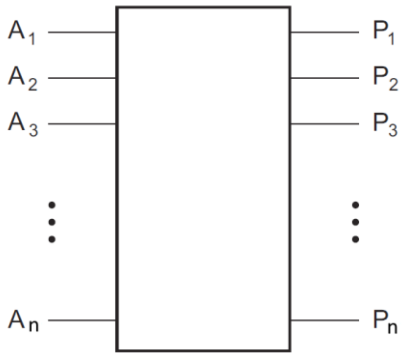



**Figure 3.4 :** Reversible circuit for bit size $n$.

**Table 3.2 :** Total reversible functions for bit size n.

| n | # of Reversible Function | # of Quantum Function |
|---|---|---|
| 1 | 2 | $\infty^4$ |
| 2 | 24 | $\infty^{16}$ |
| 3 | 40320 | $\infty^{64}$ |
| 4 | 20922789888000 | $\infty^{256}$ |

Since quantum computing is also reversible, reversible computing can be considered as a subset of quantum computing. The possible total function comparison between these computation types can be seen in Table 3.2. In quantum computing, matrices has $n^2$ degrees of freedom for n bit sized circuits (n x n square matrix) therefore total possibilities can be calculated as $\infty^{n^2}$ (there is $\infty$ possible ways to describe a real element in unitary matrix). Unitary matrices ($U^\dagger U = UU^\dagger = I$) restricts $2n^2$ degrees of freedom that formed by qubits which has two degrees of freedom (real and imaginary parts).

To synthesize all reversible functions, a universal gate library can be constructed with using the gates explained in Quantum Computing chapter.

# 4. QUANTUM CIRCUIT SYNTESIS

In this thesis, we focus on synthesis of deterministic quantum circuits. In literature, various circuit synthesis methods proposed to synthesis these deterministic quantum circuits [17] [23] [24] [25]. Most of them focused on heuristic synthesis. There are also optimal synthesis methods, but these are way behind when run times compared with others. Contrariwise, optimized circuits will decrease the run time of quantum computer, because each gate in these circuits are corresponding one or more signals depending on realization method [27]. Moreover, optimized circuits will boost the security of the computation [26]. Nevertheless, optimal circuit synthesis achieved only up to four bits in the literature [28]. When we consider recent experiments [29] where 84 qubits are used, optimal circuit synthesis can take very long time, therefore it is not practical at all. This is the main motivation why we aim at a fast synthesis algorithm in this study.

In our synthesis process, we face a decision of whether or not to use garbage outputs. Few academic works use garbage outputs, especially those synthesizing functions with high number of bits [30]. Garbage outputs are additional bits to use when they needed. In general, they are not considered favorable because of the problems in energy efficiency [10]. In this thesis, we do not use garbage outputs. In the literature, it is well known that all reversible functions can be implemented without a need of a garbage bit by using the CNT (CNOT, Not, Toffoli) gate library. This library includes gates with only positive control lines. In this thesis, we improve the CNT library by taking negative control lines into account which provides important circuit cost reduction. We call this new library as PN-CNT that stands for “Positively and Negatively Controlled CNOT, Not, Toffoli”. Figure 4.1 illustrates CNT and PN-CNT libraries.

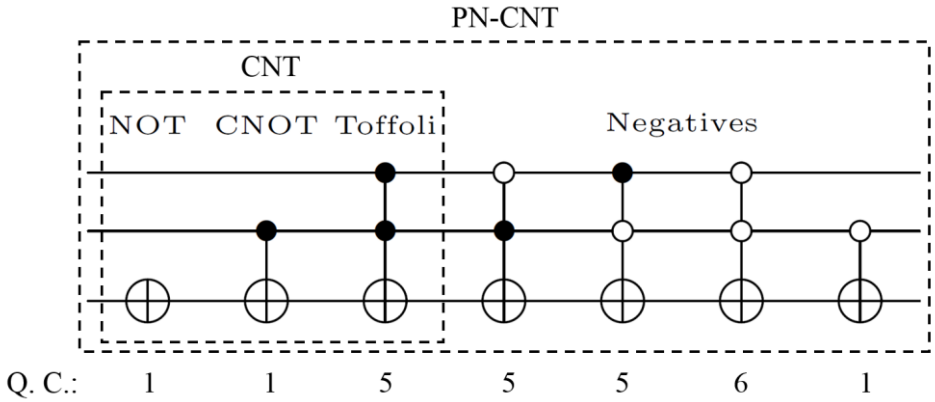



**Figure 4.1 :** CNT and PN-CNT libraries with showing Quantum Costs for each gate.

Our synthesis method efficiently implements any desired reversible function with quantum gates. Our method has two steps. In the first step, we use permutational trial based algorithm to find "essential functions" with optimum gate usage for selected bit size. Instead of using optimal circuits for essential functions, one could use circuits that are not necessarily optimal that would result in a faster algorithm, but also a larger circuit. Circuit synthesis approach, is followed by a sorting process to obtain desired functions. This is the key part of our algorithm, where we gain our synthesis speed. Although optimum circuit synthesis a time consuming process, because of the essential functions' fewness and sorting algorithms' speed, total synthesis time is still very short compared to the studies in the literature. In the second step, we perform optimization by constructing our templates. Templates are reversible circuits realizing the same function with different gate combinations that results in different circuit area costs. We construct our templates in two ways that are by directly using the gates from our PN-CNT library and by considering the inner quantum structure of these gates. The proposed templates not only optimize our synthesized circuits but also show us that optimum area solutions proposed in the literature are not actually optimum; they can be improved.

### 4.1 Negative Control Lines

Toffoli gates with negative control lines based on the same principle like their positive counterparts. Unless, they accept zero instead of one for its negative control line (s) to invert the target value. Table 4.1 and Figure 4.2 illustrates how these negative gates are work.

**Table 4.1 :** Truth table of negative controlled gates.

| n-CNOT | | np-Toffoli | | nn-Toffoli | |
|---|---|---|---|---|---|
| Input cba | Output c'b'a' | Input cba | Output c'b'a' | Input cba | Output c'b'a' |
| 000 | 100 | 000 | 000 | 000 | 000 |
| 001 | 101 | 001 | 001 | 001 | 001 |
| 010 | 010 | 010 | 110 | 010 | 110 |
| 011 | 011 | 011 | 011 | 011 | 011 |
| 100 | 000 | 100 | 100 | 100 | 100 |
| 101 | 001 | 101 | 101 | 101 | 101 |
| 110 | 110 | 110 | 010 | 110 | 010 |
| 111 | 111 | 111 | 111 | 111 | 111 |

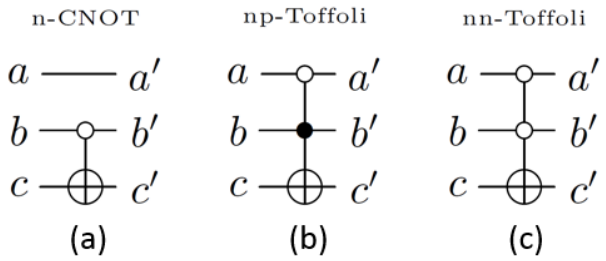



**Figure 4.2 :** Negative control lined gates, (a)n-CNOT, (b)np-Toffoli, (c) nn-Toffoli.

### 4.2 Essential Functions

For a certain bit size n, a reversible Boolean function has a truth table with $2^n$ rows that is resulting in $2^n!$ possible functions. Instead of implementing each of these functions separately, we consider very small amount of the total functions called as essential functions. To implement an essential function, we first consider a truth table such that input and output bit values are always identical. Then we switch any two rows (elements) without changing others. The result is an essential function (Figure 4.3). The total number of essential functions for a specific bit size n is:

$$C\binom{2^n}{2} = \frac{2^n * (2^n - 1)}{2} = 2^{n-1} * (2n - 1) \quad \textbf{(4.1)}$$

Table 4.2 visualizes this formula. The ratio of the number of total and essential functions increases exponentially with the bit size. This means that the more bits we have, the more beneficial our approach is. The reason behind this the function synthesis process is the main time consuming part of all approaches. By limiting this process, we minimize the run time of our algorithm.

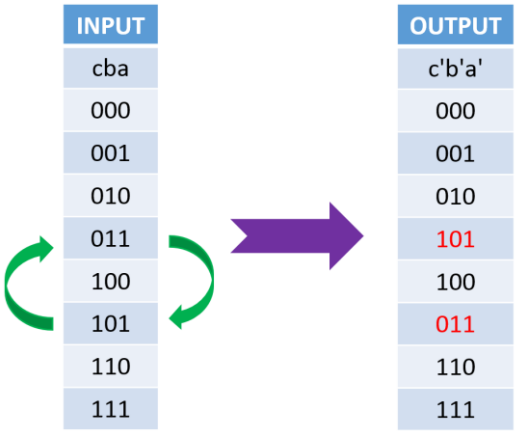



**Figure 4.3 :** Essential Function for 3 bit circuit, 2 row switched their position between each other.

**Table 4.2 :** Essential Function Number According To Bit Size.

| **Bit Size** | **Functions** | |
|---|---|---|
| | ***# of Essential Functions*** | ***# of Total Functions*** |
| 2 | 6 | 24 |
| 3 | 28 | 40320 |
| 4 | 120 | 20922789888000 |
| 5 | 469 | 2.613308e + 35 |
| 6 | 2016 | 1.268869e + 89 |

To synthesize essential functions, we develop an algorithm that results in optimal circuit sizes in terms of the reversible gate costs. Our algorithm based on permutation trials. At first, essential functions are determined by our algorithm and essential function library is created. For considered bit size, all possible gates are placed into gate library. Seeking process picks one of these essential functions respectively and starts circuit construction trials with circuit size 0, which results with identical function. The function obtained by this trials, compared with the picked essential function, if it is the one that algorithm is looking for, the circuit added to essential function library and program picks another essential function to realize it. If it is not, permutational trials are continued where it left. When all possible permutations are applied for the certain circuit size, circuit size incremented by one. Figure 4.4 represents a diagram for this part of the program. After all essential functions are obtained, any desired function can be synthesized by sorting them.

Essential functions are very few when compared with the whole function set for a specific bit. That is the reason why we preferred to use an optimal synthesis algorithm among non-optimal algorithms based on exact methods, constructive

approaches, decision diagrams and exclusive sum of products [31] . Since circuit size is another criteria, the run time of our algorithm is slightly worse than non-optimal algorithms, but it results in optimal circuit size.

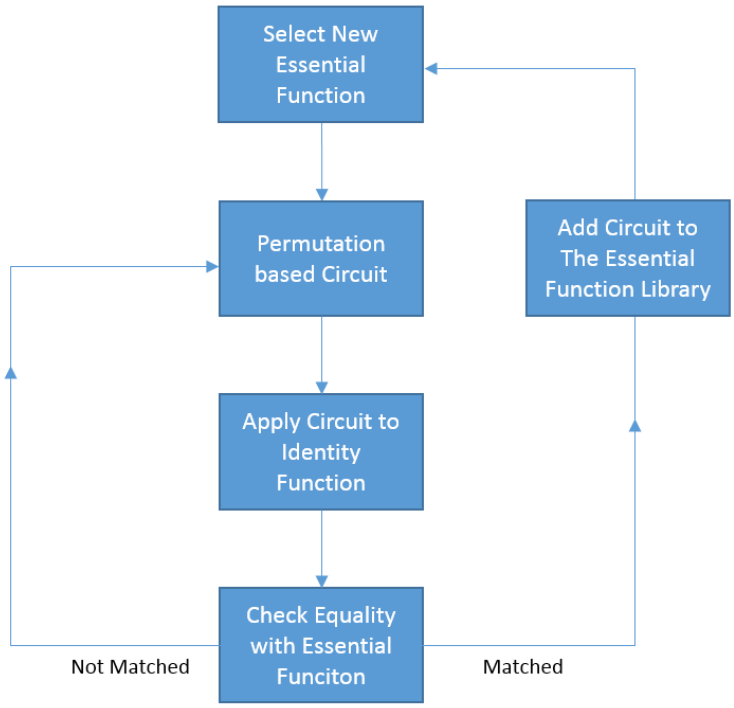



**Figure 4.4 :** Flow chart of the algorithm.

### 4.3 Sorting

After achieving essential functions, they can used to implement any given function with desired sorting algorithm. Table 4.3 shows an example of this; $f_1$ and $f_2$ are essential functions used to obtain $F$. Red lines are indicating the swapped rows. Figure 4.5 shows the circuit realization of $F$.

**Table 4.3 :** Realization of a Function With Essential Functions.

| $f_1$ | | + $f_2$ | | = $F$ | |
|---|---|---|---|---|---|
| Input cba | Output cba | Input cba | Output cba | Input cba | Output cba |
| 000 | 000 | 000 | **100** | 000 | **100** |
| 001 | 001 | 001 | 001 | 001 | 001 |
| 010 | 010 | 010 | 010 | 010 | 010 |
| 011 | **101** | 011 | 011 | 011 | **101** |
| 100 | 100 | 100 | **000** | 100 | **000** |
| 101 | **011** | 101 | 101 | 101 | **011** |
| 110 | 110 | 110 | 110 | 110 | 110 |
| 111 | 111 | 111 | 111 | 111 | 111 |

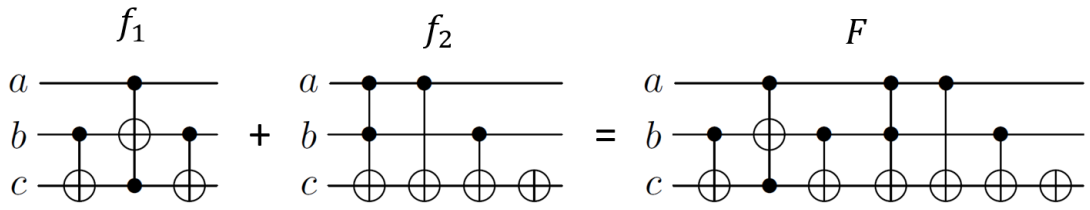



**Figure 4.5 :** Essential Function $f_1$ and $f_2$ used to obtain $F$ function.

Sorting algorithms well studied in the literature. We use a selection sort algorithm among many different options. The reason is that it checks the equivalence of the input and output bits row by row. If input and output bits are equal, the algorithm goes to the next row; if they are not equal, the algorithm makes them equivalent using essential functions. This process uses essential functions effectively. However, operations like sliding or dividing used in other sorting algorithms necessitates relatively larger amount of essential function usage. For example, merge sort divides the set into subsets, and sorts these subsets first. After subset sorting completed, equal sized subsets are joined together in pairs forming new subsets. These new subsets sorted again. This process results with extra usage of essential functions. Similarly, sliding process used in insertion sort method has the same handicap. Each slide requires an extra essential function. Figure 4.6 illustrates difference between sorting algorithms, using essential functions.

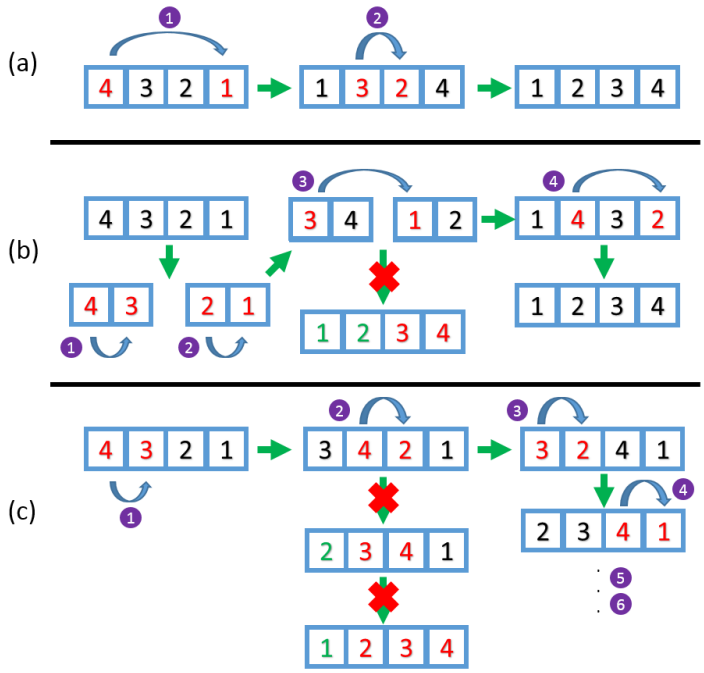



**Figure 4.6 :** Sorting process with essential functions: (a) Selection Sort, (b) Merge Sort, (c) Insertion Sort. Numbers near arrows counts used essential functions.

Selection sort can be improved when correct sequence applied to the function. Table 4.4 shows an example for this situation; reds represent mismatched rows, greens represents corrected rows. Four rows are not in their correct line for a given function. This creates 12 possible ways and two of them presented in this table. Top-Down column of Table 4.4 shows a selection algorithm that starts from the first row and proceeds through to the last row. Optimal column of Table 4.4 column shows the optimal sorting algorithm for this function that obtained after a trial of all possible sequences. E.F. stands for essential functions, shows which essential function is used. Cost of each essential function written in the below it and total cost of the function shown at right bottom corner of the table. Circuit realizations can be seen in Figure 4.7.

If we call mismatching line number as $m$ for a function, all possible sequences can be calculated with $m!/2$. For the higher mismatched line numbers, total possible ways increases.

**Table 4.4 :** Realization of a Function With Essential Functions.

| **Top-Down** | | | | | **Optimal** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 000 | 0 | 0 | 0 | 0 | 000 | 0 | 0 | 0 | 0 |
| **111** | **7** | 1 | 1 | 1 | **111** | **7** | **3** | 1 | 1 |
| **001** | **1** | **7** | 2 | 2 | **001** | **1** | **1** | **3** | 2 |
| **010** | **2** | **2** | **7** | 3 | **010** | **2** | **2** | **2** | 3 |
| 100 | 4 | 4 | 4 | 4 | 100 | 4 | 4 | 4 | 4 |
| 101 | 5 | 5 | 5 | 5 | 101 | 5 | 5 | 5 | 5 |
| 110 | 6 | 6 | 6 | 6 | 110 | 6 | 6 | 6 | 6 |
| **011** | **3** | **3** | **3** | 7 | **011** | **3** | 7 | 7 | 7 |
| E.F. | 1-7 | 2-7 | 3-7 | ✓ | E.F. | 7-3 | 1-3 | 2-3 | ✓ |
| Cost | 4 | 4 | 1 | **9** | Cost | 1 | 2 | 2 | **5** |

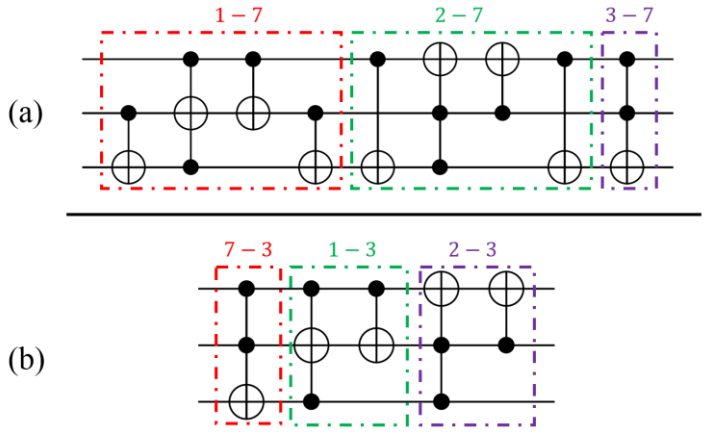



**Figure 4.7 :** Representations of the functions in Table 4.4. (a) TD approach, (b) OPT approach.

We try different sequences; top-down, pick smallest and optimal circuit. Worst case complexities for these sequences are calculated by considering the time spent on row check procedure. $k$ representing the total line number that equals to $2^n$ for bit size $n$.

### 4.3.1 Top-Down

Top-down checks identical function and desired function row by row from top to bottom. Each time mismatched row found, essential function is applied (Table 4.4(a)). Its complexity is $O(k)$.

### 4.3.2 Pick Smallest

This one checks mismatched lines like previous. Instead of correcting each mismatched line instantly, it stores the costs of essential functions in memory. When checking process complete, it applies the low costed essential function and repeats checking process. For this one, complexity is $O(k^2)$.

### 4.3.3 Optimum

Optimal circuits found by checking all possible sequences. The first sequence and its cost directly saved to the memory. When new sequence obtained, the cost for this sequence compared with the cost in the memory. Low costed circuit and its cost stored in the memory after each comparison thus optimal sequence found within all possibilities. Complexity of this method is $O(k^k)$.

These methods compared with their run time (in seconds) and reversible average cost for bit size 3 in Table 4.5.

**Table 4.5 :** Sequence Comparison.

| | TD | PS | OPT |
|---|---|---|---|
| R.C. | 16.69 | 15.75 | 14.48 |
| Time | 8 | 8 | 377 |

## 4.4 Optimization

Optimization process used to minimize gate number in a circuit that realizes a function. We divided this process in mainly two parts. In the first part, after synthesis of each function, circuits are scanned to find identical neighbor gates and *templates*. In the second part, synthesized circuits scrutinized in quantum circuit level. Each

gate corresponding to a mixture of subgates expanded to its quantum structure. Thus, invisible identical neighbors and templates revealed.

In both optimization sections, we use templates that includes only two gate. New templates can be constructed by increasing the gate number within it. Unfortunately, this is limitless process to follow up. Main reason to limit them with two gates is to keep the balance between run time and optimization in the circuits.

### 4.4.1 Templates Using Circuit Library

We look for pre-defined templates in our sorting based synthesis and replace them with their optimal equivalent to reduce total quantum cost. Because of the sorting sequence, essential functions used one after the other that sometimes forms identical neighbor gates. This results at least two or more gate reduction in the final circuit, shown in Figure 4.8. Additionally we create 12 more templates for negative gates. One of them shown in Figure 4.9.

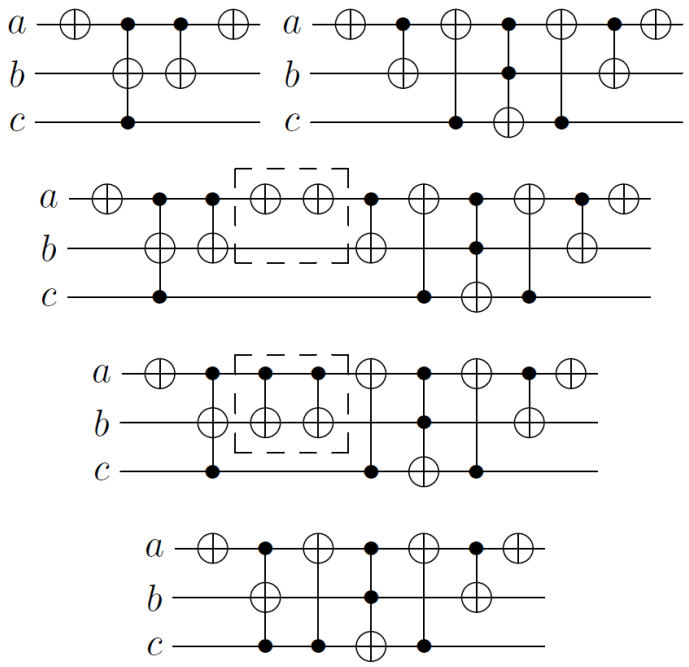



**Figure 4.8 :** Sorting sequence with identical neighbor gates; 4 gates removed from the circuit.

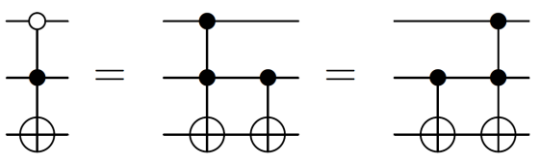


**Figure 4.9 :** Templates for gates with a negative control line.

### 4.4.2 Templates Using Quantum Structure of Gates

After optimizing our circuits using PN-CNT library based templates, here we perform a second optimization process using quantum inner structure of the gates. In this part, Toffoli gates are expanded to their quantum circuit structure (Figure 4.10a, Figure 4.10c). Rescan process begins to find new identical neighbors. If a Toffoli gate has an appropriate CNOT gate neighbor (Figure 4.10b, Figure 4.10d), second reduction is applied. There are 12 different situations for positive controlled Toffoli-CNOT case in 3 bit reversible circuits.

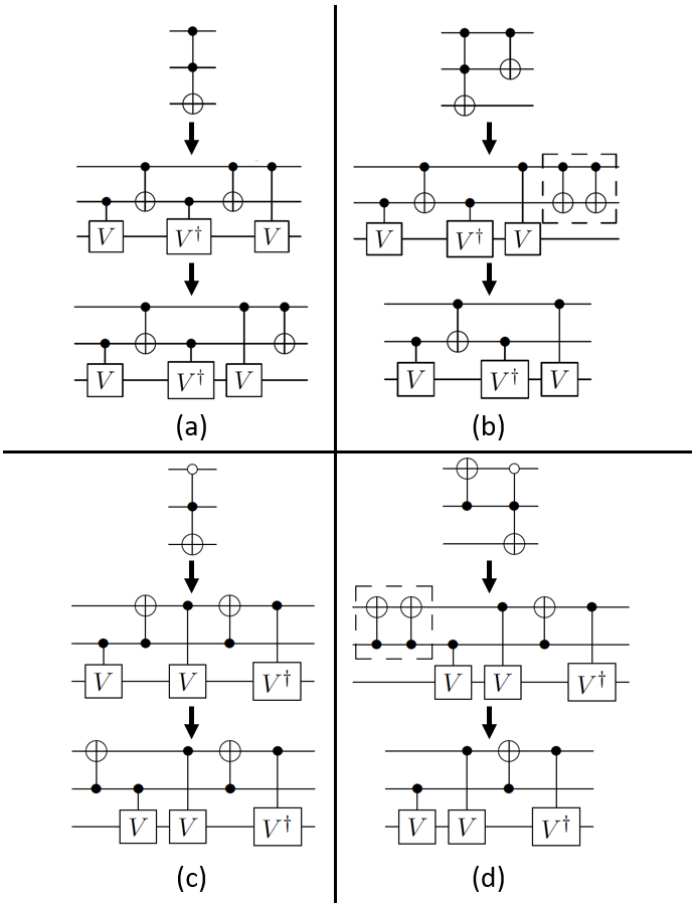



**Figure 4.10 :** (a) Toffoli gate with its inner quantum realization. (b) Template for positively controlled Toffoli. (c) Single negatively controlled Toffoli gate its quantum realization. (d) Template with single negative controlled Toffoli.

We experimentally obtain all Toffoli realizations with using permutation based algorithm in MATLAB. This is similar to optimal circuit synthesis process in essential functions section. Permutation starts with using a gate from NCV library. If there is no match after checking all possibilities, gate number increases in the circuit. This process continues until a match found. Table 4.6 shows the results. Minimum

size row shows how many gates needed to realize the specific Toffoli function. For this minimum size, total realization row shows how many different ways this function can be realize. All realizations are shown in Appendix I.

**Table 4.6 :** Experimental Results for Toffoli Realization.

| | Toffoli | pn-Toffoli | nn-Toffoli |
|---|---|---|---|
| Minimum Size | 5 | 5 | 6 |
| Total Realizations | 40 | 40 | 112 |

## 5. RESULTS

Implementations realized in C. All experiments run on a 3.20-GHz Intel Core i5 CPU (only single core used) with 4.00 GB memory.

Table 5.1 represents results of our proposed approach and optimal methods in the literature. Run times are in seconds. Reversible costs are calculated by counting each gate in CNT library cost as one. Quantum costs are calculated by considering each gate in NCV library cost as one (Figure 4.1). Numbers in "size" column represents the number of gates used. Numbers in other columns represents the number of functions implemented with the corresponding gate number in size column. For example, the first row tells that 28 gates are used to implement 4 different functions based on the TD approach. "Proposed Approach" column shows the results of our algorithm. CNT library based synthesis results shown under CNT column. Comparing PS and OPT approaches, OPT overwhelms PS with a 22.06% improvement in reversible circuit cost and 15.17% improvement in quantum circuit cost. When run times compared, PS is far better than OPT.

Our PN-CNT approach uses the library of PN-CNT presented in Figure 4.1. Note that this approach gives the best result in reversible costs. Column named Optimal, presents optimal results of the studies in the literature that focused on synthesis with reversible gates. We convert these reversible circuits to quantum form and apply our quantum optimization method illustrated in Figure 4.10. Thus, we reduce the optimal CNT quantum cost by 5.61%, from 13.88 to 13.10.

Comparing our synthesis approach with the optimal ones, our run times are always better at the cost of the circuit size. This is an expected result for 3 bit circuits. Here, an important point is that our synthesis approach can effectively work in higher bits. For example to implement 4 bit circuits, 120 essential functions are needed out of all 20922789888000 functions. However, the optimal synthesis method is not practically applicable even for 5 bit circuits. If the optimal method is used to synthesize 5 bit circuits, it will take $447 \times 10^{18}$ years that is justified using the numbers in Table 4.2.

**Table 5.1 :** Reversible Functions Obtained With Specific Gate Number According To 3 Bits.

| Size | Proposed Approach | | | | | | Optimal | |
|---|---|---|---|---|---|---|---|---|
| | CNT | | | PN-CNT | | | CNT | NCL |
| | TD | PS | OPT | TD | PS | OPT | | |
| 28 | 4 | | | | | | | |
| 27 | 29 | | | | | | | |
| 26 | 90 | 10 | | | | | | |
| 25 | 207 | 31 | | | | | | |
| 24 | 436 | 145 | | | | | | |
| 23 | 791 | 238 | | | | | | |
| 22 | 1252 | 682 | 1 | | | | | |
| 21 | 1954 | 1031 | 0 | | | | | |
| 20 | 2523 | 1625 | 5 | | | | | |
| 19 | 3349 | 2720 | 47 | 4 | 9 | | | |
| 18 | 3772 | 3129 | 167 | 22 | 11 | | | |
| 17 | 4125 | 4022 | 473 | 132 | 156 | | | |
| 16 | 4211 | 4383 | 1283 | 249 | 224 | | | |
| 15 | 3842 | 4179 | 2748 | 1126 | 582 | | | |
| 14 | 3522 | 4126 | 4657 | 1758 | 1422 | 1 | | |
| 13 | 2835 | 3528 | 6018 | 2988 | 2388 | 64 | | |
| 12 | 2308 | 3104 | 6586 | 4686 | 3690 | 364 | | |
| 11 | 1706 | 2389 | 5696 | 5158 | 5509 | 1160 | | |
| 10 | 1239 | 1772 | 4347 | 5945 | 6493 | 2500 | | |
| 9 | 843 | 1203 | 3137 | 5752 | 5906 | 5820 | | |
| 8 | 547 | 818 | 2178 | 4485 | 5007 | 8756 | 577 | |
| 7 | 340 | 531 | 1354 | 3512 | 3966 | 8656 | 10253 | |
| 6 | 194 | 322 | 825 | 2321 | 2623 | 6837 | 17049 | 3236 |
| 5 | 111 | 181 | 438 | 1190 | 1400 | 3996 | 8921 | 20480 |
| 4 | 52 | 80 | 208 | 615 | 623 | 1611 | 2780 | 13282 |
| 3 | 25 | 43 | 100 | 286 | 232 | 452 | 625 | 2925 |
| 2 | 9 | 18 | 42 | 78 | 66 | 90 | 102 | 369 |
| 1 | 3 | 9 | 9 | 12 | 12 | 12 | 12 | 27 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| R.C. | 15.97 | 14.82 | 11.55 | 9.79 | 9.50 | 7.31 | 5.86 | 4.57 |
| Q.C. | 34.14 | 25.63 | 21.74 | 32.85 | 30.82 | 28.97 | 13.88 | - |
| Time | 9 | 9 | 6m33s | 12 | 12 | 5m59s | 40 [32] | - [33] |

# 6. CONCLUSIONS AND RECOMMENDATIONS

Quantum computation became one of the most valuable computer science topic in last years. Quantum circuit synthesis and optimization methods come into prominence with the experiments that focuses quantum computation. Reversible quantum circuit design has an important role at this point, which is the core of this computation.

In this thesis, a new method for synthesis and optimization of quantum circuits presented. A fast synthesis algorithm that implements any given reversible Boolean function with quantum gates is proposed. Instead of an exhaustive search on every given function, proposed algorithm creates a library of essential functions and performs sorting. Our synthesis algorithm is considerably faster than the optimal ones presented in the literature. We assume that this difference will increase exponentially for large bit sizes.

Circuits optimized by using templates. We limit our templates with considering only two gates to balance run time and cost reduction. The templates that we propose mainly consist of Toffoli gates with negative and positive controlling lines. These templates not only optimize our synthesized circuits but also show us that they can used to optimize proposed solutions in the literature. We reduce the previously proposed optimal CNT synthesis cost by 5.61%.

We are currently working on our sequence selection algorithm to find optimal sequences in heuristic way. Additionally, low-level circuit synthesis is one of our primary aims to reduce quantum operations in quantum computers. As a future work, we will seek deeper optimization methods with using elementary gates and pulse signals.

## REFERENCES


[1] **Feynman, Richard P.** (1982). Simulating physics with computers. International journal of theoretical physics 21.6 467-488.

[2] **Shor, P. W.** (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM journal on computing, 26(5), 1484-1509.

[3] **Rivest, R. L., Shamir, A., & Adleman, L.** (1978). A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120-126.

[4] **Grover, L. K.** (1996). A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth annual ACM Symposium on Theory of Computing, pp. 212-219. ACM.

[5] **Moore, G. E.** (1965). Cramming more components onto integrated circuits.

[6] **Stolze, J., & Suter, D.** (2008). Quantum computing: a short course from theory to experiment. John Wiley & Sons.

[7] **Lucero, E., Barends, R., Chen, Y., Kelly, J., Mariantoni, M., Megrant, A.,** et al. (2012). Computing prime factors with a Josephson phase qubit quantum processor. Nature Physics, 8(10), 719-723.

[8] **Nielsen, M. A., & Chuang, I. L.** (2010). Quantum computation and quantum information. Cambridge university press.

[9] **Barenco, A., Bennett, C. H., Cleve, R., DiVincenzo, D. P., Margolus, N., Shor, P.,** et al. (1995). Elementary gates for quantum computation. Physical Review A, 52(5), 3457.

[10] **Toffoli, T.** (1980). Reversible computing, pp. 632-644. Springer Berlin Heidelberg.

[11] **O'Brien, J. L.** (2007). Optical quantum computing. Science, 318(5856), 1567-1570.

[12] **Kielpinski, D., Monroe, C., & Wineland, D. J.** (2002). Architecture for a large-scale ion-trap quantum computer. Nature, 417(6890), 709-711.

[13] **Cory, D. G., Laflamme, R., Knill, E., Viola, L., Havel, T. F., Boulant, N.,** et al. (2000). NMR based quantum information processing: Achievements and prospects. Fortschritte der Physik, 48(9-11), 875-907.

[14] **Loss, D., & DiVincenzo, D. P.** (1998). Quantum computation with quantum dots. Physical Review A, 57(1), 120.

[15] **DiVincenzo, D. P.** (2000). The physical implementation of quantum computation. arXiv preprint quant-ph/0002077.

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

[17] **Shende, V. V., Bullock, S. S., & Markov, I. L.** (2006). Synthesis of quantum-logic circuits. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 25(6), 1000-1010.

[18] **Landauer, R.** (1961). Irreversibility and heat generation in the computing process. IBM journal of research and development, 5(3), 183-191.

[19] **Bennett, C. H.** (2003). Notes on Landauer's principle, reversible computation, and Maxwell's Demon. Studies In History and Philosophy of Science Part B: Studies In History and Philosophy of Modern Physics, 34(3), 501-510.

[20] **De Vos, A.** (1999). Reversible computing. Progress in Quantum Electronics, 23(1), 1-49.

[21] **Markov, Igor L.** (2014). Limits on fundamental limits to computation. Nature 512.7513, 147-154.

[22] **Lambson, B., Carlton, D., & Bokor, J.** (2011). Exploring the thermodynamic limits of computation in integrated systems: Magnetic memory, nanomagnetic logic, and the landauer limit. Physical review letters, 107(1), 010604.

[23] **Maslov, D., Dueck, G. W., & Miller, D. M.** (2005). Toffoli network synthesis with templates. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 24(6), 807-817.

[24] **Große, D., Wille, R., Dueck, G. W., & Drechsler, R.** (2009). Exact multiple-control toffoli network synthesis with SAT techniques. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 28(5), 703-715.

[25] **Gupta, P., Agrawal, A., & Jha, N. K.** (2006). An algorithm for synthesis of reversible logic circuits. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 25(11), 2317-2330.

[26] **Shor, P. W.** (1995). Scheme for reducing decoherence in quantum computer memory. Physical review A, 52(4), R2493.

[27] **Fedorov, A., Steffen, L., Baur, M., Da Silva, M. P., & Wallraff, A.** (2011). Implementation of a Toffoli gate with superconducting circuits. Nature, 481(7380), 170-172.

[28] **Golubitsky, O., & Maslov, D.** (2012). A study of optimal 4-bit reversible toffoli circuits and their synthesis. Computers, IEEE Transactions on, 61(9), 1341-1353.

[29] **Bian, Z., Chudak, F., Macready, W. G., Clark, L., & Gaitan, F.** (2013). Experimental Determination of Ramsey Numbers. Physical review letters, 111(13), 130505.

[30] **Wille, R., & Drechsler, R.** (2009). BDD-based Synthesis of Reversible Logic for Large Functions. In Proceedings of the 46th Annual Design Automation Conference, pp. 270-275. ACM.

[31] **Schönborn, E., Datta, K., Wille, R., Sengupta, I., Rahaman, H., & Drechsler, R.** (2014). Optimizing DD-based Synthesis of Reversible Circuits using Negative Control Lines. Proceedings of the 2014 IEEE Seventeenth Design and Diagnostics of Electronic Circuits & Systems.

[32] **Shende, V. V., Prasad, A. K., Markov, I. L., & Hayes, J. P.** (2003). Synthesis of Reversible Logic Circuits. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 22(6), 710-722.

[33] **Wille R., Soeken M., Przigoda N., Drechsler R.,** (2012) Exact synthesis of Toffoli gate circuits with negative control lines. Multiple-Valued Logic (ISMVL), 2012 42nd IEEE International Symposium on. IEEE, pp. 69- 74.

## APPENDICES

**APPENDIX A:** Toffoli Realizations

**APPENDIX B:** Test Program

This program was developed and used during my thesis studies.

## APPENDIX A

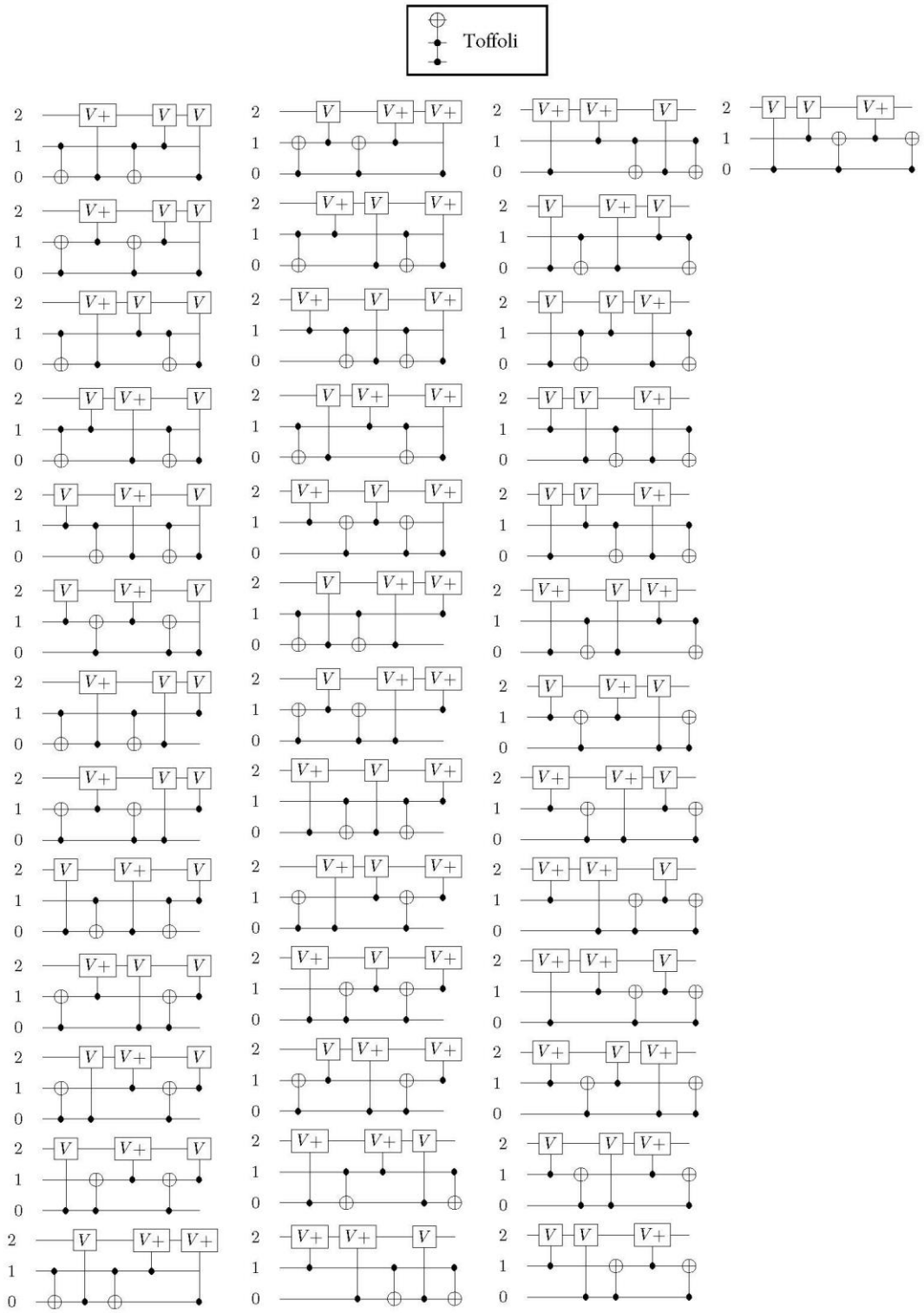



**Figure A.1 :** Subgates of Toffoli with positive control lines.

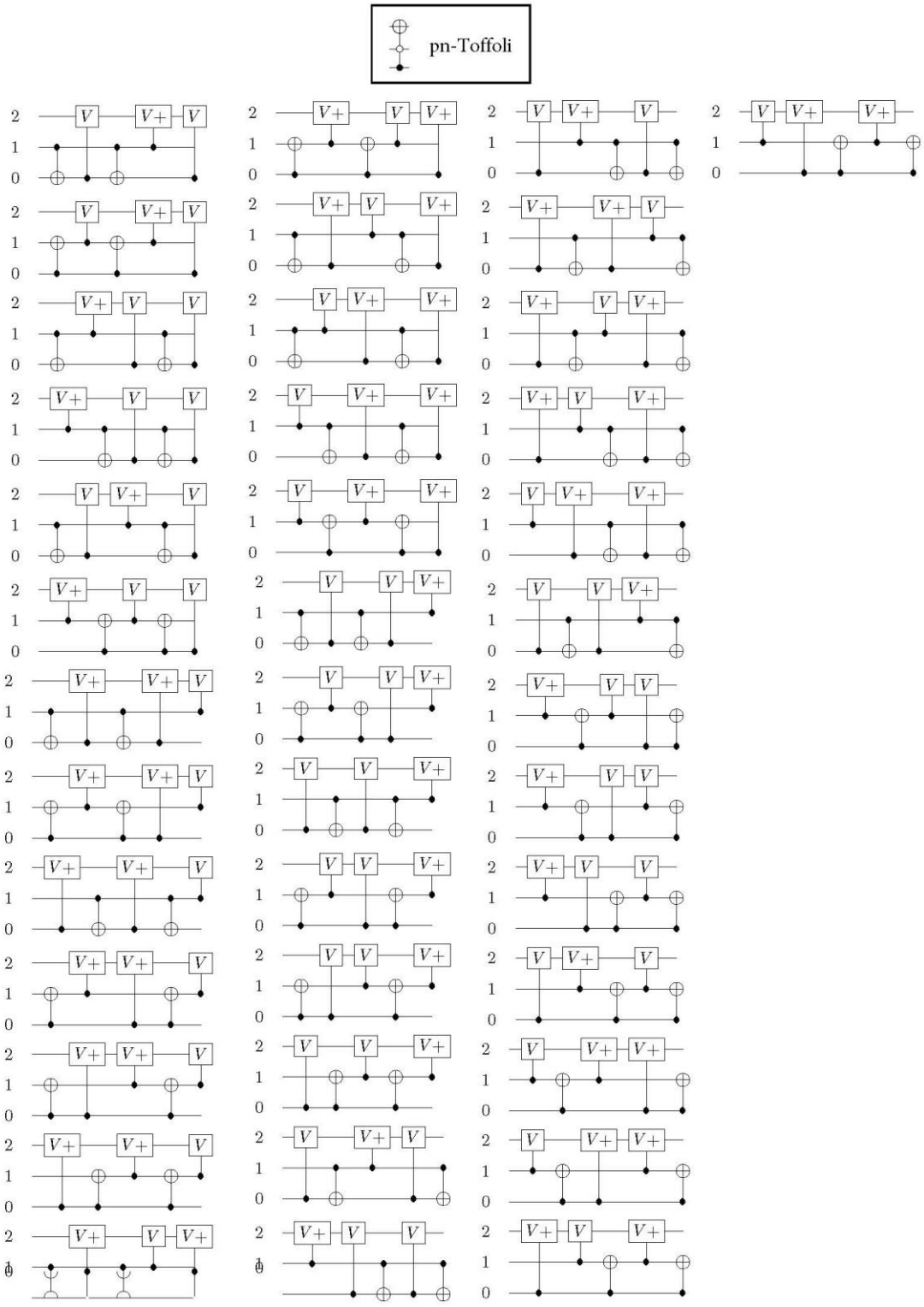



**Figure A.2 :** Subgates of Toffoli with positive and negative control lines.

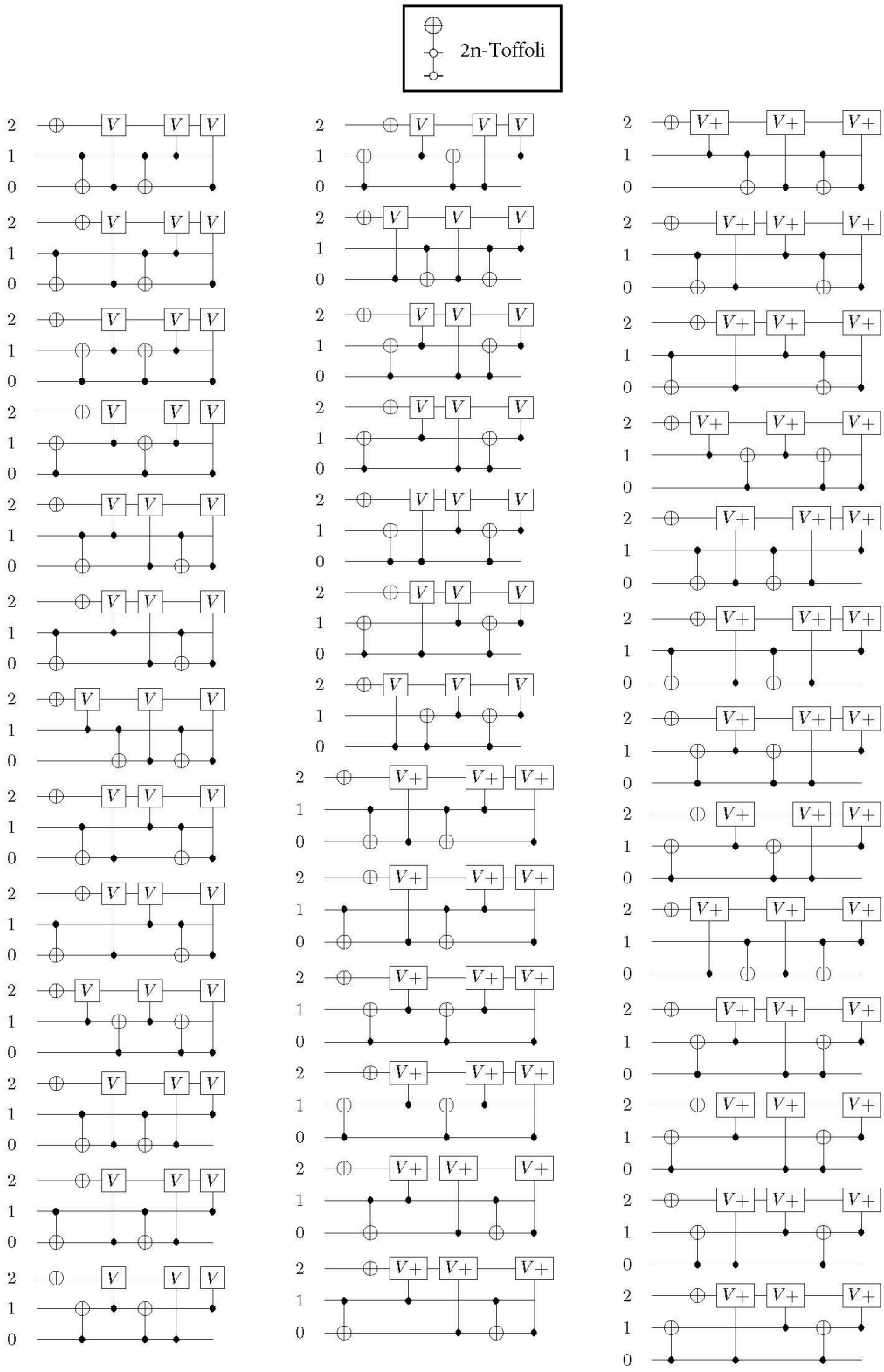



**Figure A.3 :** Subgates of Toffoli with negative control lines.

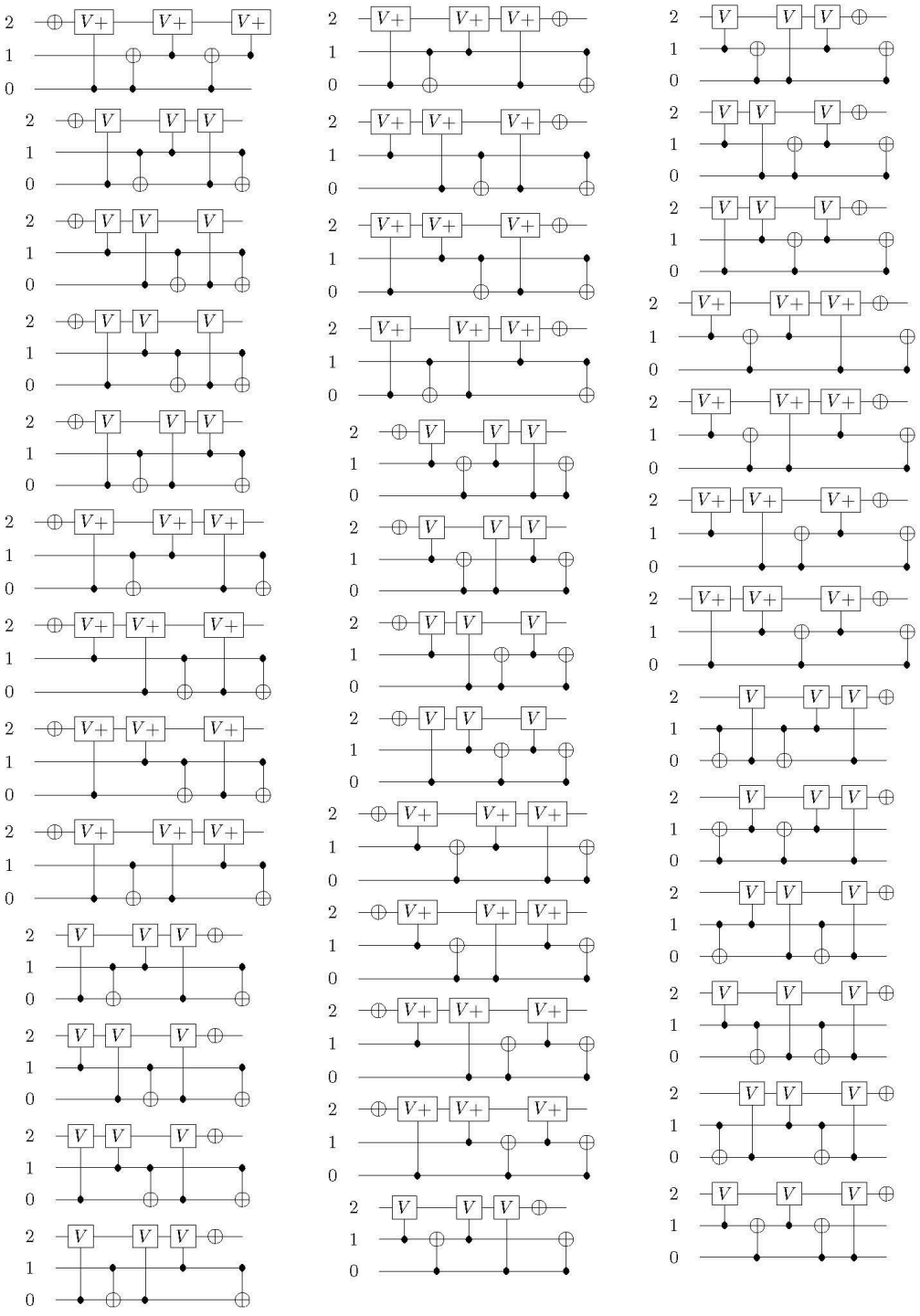


**Figure A.4 :** Subgates of Toffoli with negative control lines.

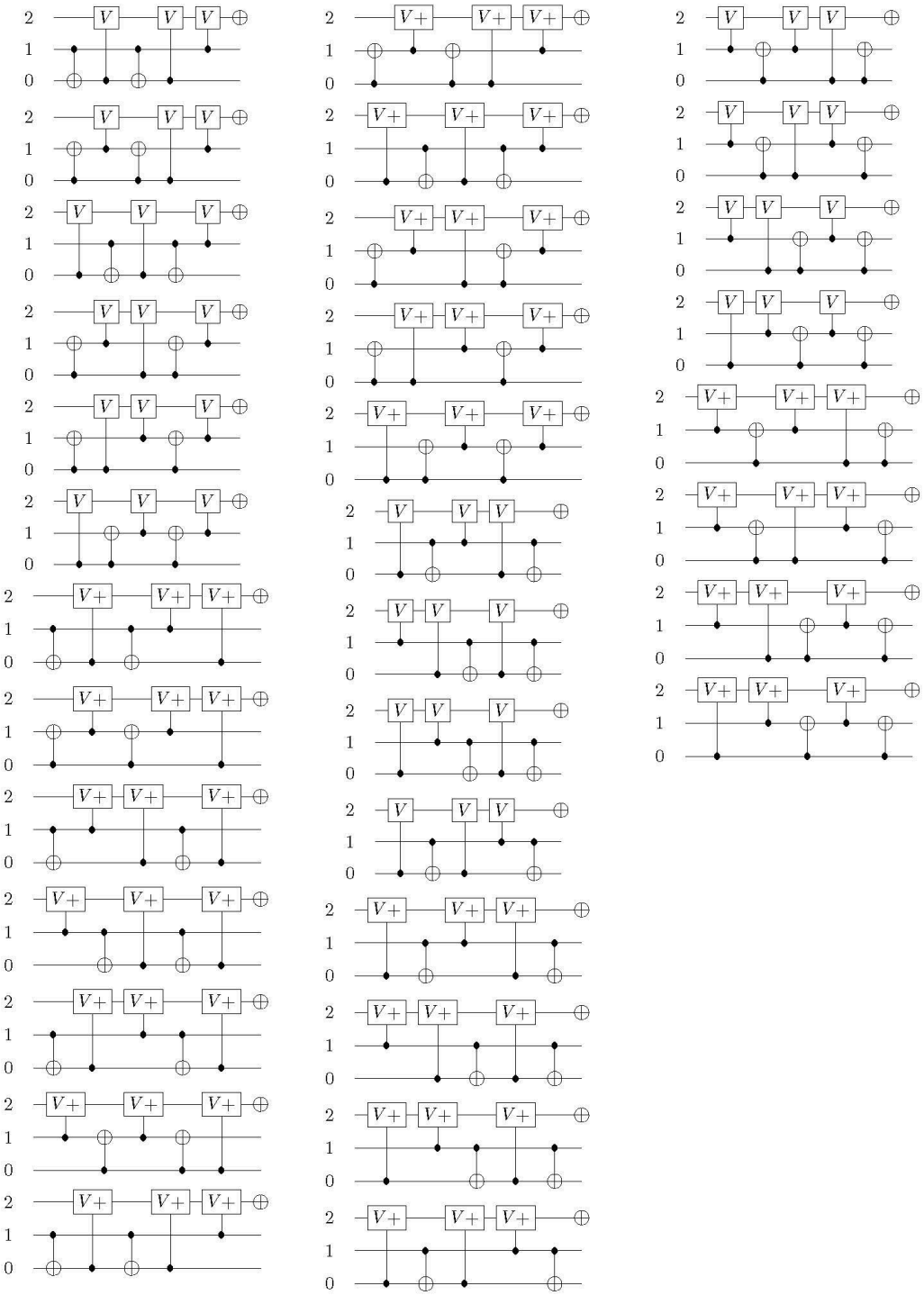


**Figure A.5 :** Subgates of Toffoli with negative control lines.

## APPENDIX B


**Figure A.6 :** Instructions are shown at strat screen.

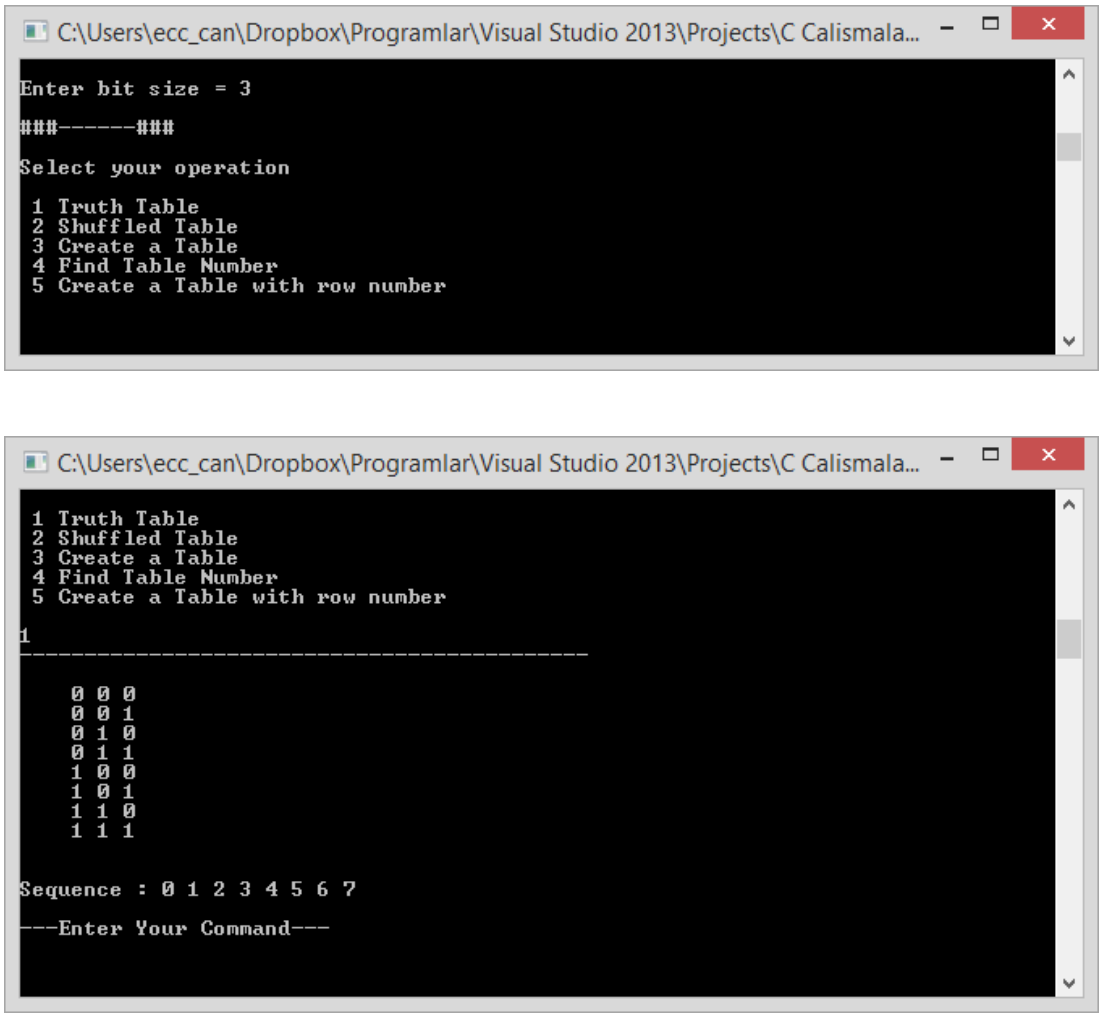



**Figure A.7 :** Desired operation can be selected after enterin bit size.

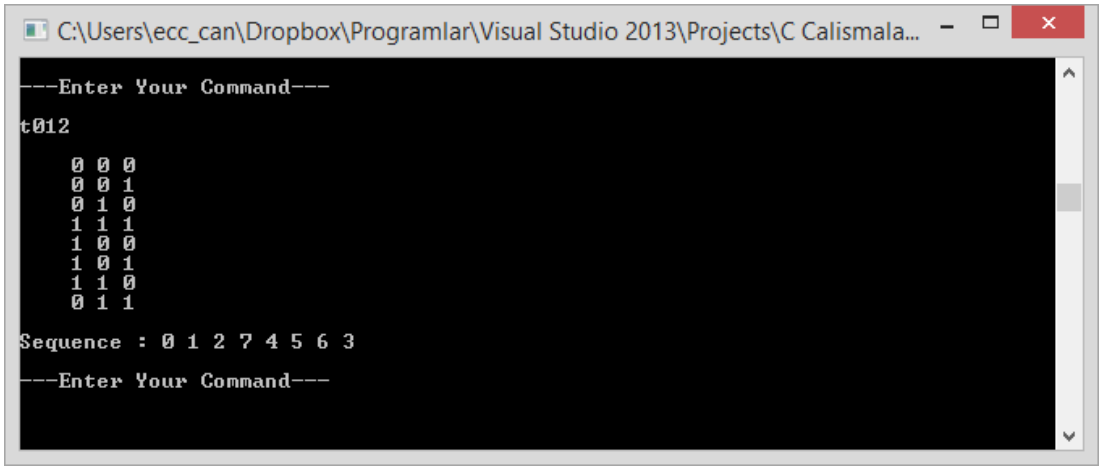
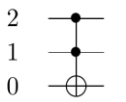



**Figure A.8 :** Circuits can be applied to the truth table by leaving blank space between gates, e.g “t012 c02 n1”.




Table

| 0 | 1 | 2 | 0′ | 1′ | 2′ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 |

**Figure A.9 :** After entering desired circuit, circuit layout can be exported in latex format, as shown.

**CURRICULUM VITAE**

**Name Surname:** **Ömer Can Susam**

**Place and Date of Birth:** **Istanbul 24.10.1988**

**E-Mail:** **susam@itu.edu.tr**

**B.Sc.:** **Electrical Engineering - Kocaeli University - 2010**

**Professional Experience and Rewards:**

2014-2015 Istanbul Technical University Research Support Program

2014-2015 TUBITAK MSc Scholarship Program in Priority Areas

01.2012 – 04.2012 Ilkotek Automation

Designing Motion Control (servo motor – driver – controller), PLC and HMI projects, developing object oriented programs with C# (4 axis race simulator).

10.2011 – 01.2012 Teksan GenSet

Designing and setting up control panels, GenSet remote control systems, GenSet-Mains Synchronous panels and programing PLC-HMI systems.

**PUBLICATIONS/PRESENTATIONS ON THE THESIS**

- Susam Ö. C., Altun M., 2014: Kuantum Devre Sentezi ve Optimizasyonu İçin Verimli Bir Algoritma. Elektrik – Elektronik, Bilgisayar ve Biyomedikal Mühendisliği Sempozyumu, ELECO November 27-29, 2014 Bursa, Turkey.
- Susam Ö. C., Altun M., 2014: An Efficient Algorithm to Synthesize Quantum Circuits and Optimization – twenty-first IEEE International Conference on Electronics Circuits and Systems, ICECS December 7-10, 2014 Marseille, France.