QUANTUM RANDOM WALK ON A NUMBER LINE IMPLEMENTATION ON DIFFERENT SIMULATORS WITH WALL-TIME BENCHMARKING

MR. WARAT PUENGTAMBOL

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF ENGINEERING
(COMPUTER ENGINEERING)
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI
2020

Quantum Random Walk on A Number Line Implementation on Different Simulators with Wall-Time Benchmarking

Mr. Warat Puengtambol B.Eng. (Computer Engineering)

A Thesis Submitted in Partial Fulfillment
of The Requirements for
the Degree of Master of Engineering (Computer Engineering)
Faculty of Engineering
King Mongkut's University of Technology Thonburi
2020

Thesis Committee

.... Chairman of Thesis Committee
(Asst.Prof. Sujin Suwanna, Ph.D.)

.... Member and Thesis Advisor
(Lect. Unchalisa Taetragool, Ph.D.)

J. Harnsomburana
.... Member
(Lect. Jaturon Harnsomburana, Ph.D.)

.... Member
(Lect. Pruet Kalasuwan, Ph.D.)

.... Member
(Lect. Tanapat Deesuwan, Ph.D.)

Copyright reserved

Thesis Title	Quantum Random Walk on A Number Line Implementation on Different Simulators with Wall-Time Benchmarking
Thesis Credits	12
Candidate	Mr. Warat Puengtambol
Thesis Advisor	Dr. Unchalisa Taetragool
Program	Master of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2020

Abstract

Since a classical computer has a limitation to solve a large size of data and complex problems, researchers have been trying to find new solutions. Quantum computing is one of them. A number of quantum algorithms have been invented. Some of them were theoretically proved that they could solve problems faster than their classical versions. According to the literature review, a quantum random walk has various advantages. Moreover, it can be a tool to construct other quantum algorithms. In this thesis, the one-dimensional quantum random walk is studied. Then, the quantum random walk circuit is implemented. Forest from Rigetti and Qiskit from IBM, which are quantum programming platforms, are explored. Two experiments are conducted. First, the circuits are examined on quantum computer simulators from IBM and Rigetti. The sizes of the circuits are 4, 5, 6, and 7-qubits position states. For each size, the circuit is constructed with 5 different types of shift operators. The results show that computing wall-time from IBM's quantum computer simulator is significantly lower than Rigetti's quantum computer simulator for the same size and circuit type. Moreover, on the same circuit type, a positive linear relationship between the computing wall-time and the circuit depth is presented. Second, the same circuit sizes and types from the first experiment were deployed on an actual IBM quantum computer, named *ibmq_16_melbourne*. However, the returned result is only for the 4-qubit position state. The other sizes of position states return error due to the capacity of the quantum computer. Furthermore, the returned measurement results are disturbed by the noise from the operation of the actual quantum computer.

Keywords: Circuit Depth/ Computing Wall-Time/ Quantum Random Walk/ Quantum Computer Simulator

หัวข้อวิทยานิพนธ์	การนำการเคลื่อนที่แบบสุ่มเชิงควอนตัมบนเส้นจำนวนไปปฏิบัติการบนเครื่องจำลองต่าง ๆ โดยมีการเปรียบเทียบเวลาในการประมวลผล
หน่วยกิต	12
ผู้เขียน	นายวรัตถ์ พึ่งตำบล
อาจารย์ที่ปรึกษา	ดร.อัญชลิสา แต้ตระกูล
หลักสูตร	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	วิศวกรรมศาสตร์
ปีการศึกษา	2563

บทคัดย่อ

จากข้อจำกัดของการประมวลผลข้อมูลที่มีขนาดใหญ่ รวมถึงปัญหาที่มีความสลับซับซ้อนมากบนเครื่องคอมพิวเตอร์ในยุคปัจจุบันทำให้นักวิจัยต้องค้นหาวิธีการใหม่ ๆ เพื่อให้ได้มาซึ่งประสิทธิภาพการประมวลผลที่เพิ่มมากขึ้น การประมวลผลเชิงควอนตัมเป็นหนทางหนึ่งที่จะทำให้ได้การประมวลผลที่มีประสิทธิภาพสูงขึ้น ดังนั้นอัลกอริทึมเพื่อสนับสนุนการประมวลผลด้วยเครื่องควอนตัมคอมพิวเตอร์จึงได้ถูกค้นคว้าและพัฒนา ควอนตัมอัลกอริทึมได้ถูกพิสูจน์แล้วว่ามีประสิทธิภาพในการทำงานดีกว่าอัลกอริทึมสำหรับแก้ปัญหาแบบที่เคยมีอยู่เดิมซึ่งหนึ่งในนั้นคือ การเคลื่อนที่แบบสุ่มเชิงควอนตัม อัลกอริทึมดังกล่าวมีประสิทธิภาพในการแก้ไขปัญหาที่เกี่ยวข้องกับกราฟ ซึ่งปัญหาสำคัญหลาย ๆ ปัญหาสามารถจำลองรูปแบบของปัญหาให้อยู่ในรูปของกราฟได้ ในวิทยานิพนธ์ฉบับนี้ได้ทำการศึกษาอัลกอริทึมการเคลื่อนที่แบบสุ่มเชิงควอนตัมบนพื้นที่หนึ่งมิติ และพัฒนาวงจรให้สามารถประมวลผลด้วยเครื่องจำลองควอนตัมคอมพิวเตอร์ วิทยาพนธ์ฉบับนี้ประกอบไปด้วย 2 การทดลอง การทดลองแรกเพื่อหาความแตกต่างของการประมวลผลด้วยเครื่องจำลองควอนตัมคอมพิวเตอร์จากทั้ง 2 บริษัทคือ IBM และ Rigetti โดยการทดลองได้มีการออกแบบวงจรการเคลื่อนที่แบบสุ่มเชิงควอนตัมออกมาจำนวน 5 ขนาดคือ 4 5 6 และ 7 โดยขนาดที่เพิ่มขึ้นคือขนาดของคิวบิตสำหรับแสดงตำแหน่งของการเคลื่อนที่บนพื้นที่หนึ่งมิติ นอกจากความแตกต่างจากขนาดของวงจร ในการทดลองยังมีการเพิ่มความแตกต่างของวงจรเปลี่ยนตำแหน่งในแต่ละขนาดอีก 5

ประเภท สำหรับผลจากการประมวลผลจากวงจรทั้งหมดพบว่าในวงจรประเภทเดียวกันเวลาที่ใช้ในการประมวลผลผ่านเครื่องจำลองควอนตัมคอมพิวเตอร์จาก IBM มีระยะเวลาน้อยกว่า Rigetti และสำหรับการทดลองด้วยวงจรขนาดเดียวกันแต่มีความแตกต่างที่ประเภทของวงจรสลับตำแหน่งพบว่าระยะเวลาที่ใช้ในการประมวลผลแปรผันตามความลึกของวงจร สำหรับการทดลองที่ 2 วงจรที่ถูกออกแบบสำหรับทดลองในการทดลองที่ 1 จะถูกนำไปประมวลผลบนเครื่องควอนตัมคอมพิวเตอร์จาก IBM ชื่อ *ibmq_16_melbourne* จากผลการทดลองพบว่าเฉพาะการประมวลผลโดยใช้วงจรที่มีการเก็บตำแหน่งขนาด 4 คิวบิตเท่านั้นที่สามารถได้รับผลจากการประมวลผล สำหรับวงจรขนาดอื่นๆมีการส่งค่าผิดพลาดกลับมา อย่างไรก็ตามผลการคำนวณพบว่าผลลัพธ์ที่ได้กลับมาจากเครื่องควอนตัมคอมพิวเตอร์มีค่าผิดพลาดไปจากทฤษฎี (noise) ร่วมอยู่

คำสำคัญ : การเคลื่อนที่แบบสุ่มเชิงควอนตัม/ ความลึกของวงจร/ เครื่องจำลองควอนตัมคอมพิวเตอร์/
ระยะเวลาที่ใช้ในการประมวลผล

CONTENTS

PAGE

ENGLISH ABSTRACT ii
THAI ABSTRACT iii
CONTENTS v
LIST OF TABLES vii
LIST OF FIGURES viii

CHAPTER

1. INTRODUCTION 1
- 1.1 Statement of the Problem 1
- 1.2 Objectives 5
- 1.3 Scope 3

2. BACKGROUND STUDY ON QUANTUM COMPUTING 4
- 2.1 Quantum Mechanics 4
- 2.2 Quantum Circuits 5
- 2.3 Quantum Processing Unit and Programming 7
- 2.4 IBM Single Qubit Gate and Multi-Control Toffoli Function 13

3. QUANTUM RANDOM WALKS 16
- 3.1 Quantum Random Walk 16

4. SHIFT OPERATOR FOR QUANTUM RANDOM WALK 19
- 4.1 Incrementer Circuits 19
- 4.2 A Design of Shift Operator for Quantum Random Walk 22
- 4.3 Shift Operator Circuit Verification 24

5. EXPERIMENT RESULTS AND DISCUSSION 28
- 5.1 A Benchmarking of Quantum Computer Simulators 28
- 5.2 A Quantum Random Walk Computing on Quantum Computer 50

CONTENTS (Cont'd)

PAGE

6. CONCLUSION **53**

REFERENCES **55**

CURRICULUM VITAE **58**

LIST OF TABLES

TABLE **PAGE**

2.1 Commonly used quantum gates 7

4.1 The truth table of the three-qubit incrementer circuit 20

4.2 The comparison among the notation 22

4.3 The comparison between incrementer and incrementer in this thesis 23

5.1 The analysis of wall-time from running the quantum random walk on four-qubit position state on the number line between IBM’s Qasm_simulator and Rigetti’s QVM 31

5.2 The analysis of wall-time from running the quantum random walk on five-qubit position state on the number line between IBM’s *Qasm_simulator* and Rigetti’s *QVM* 34

5.3 The analysis of wall-time from running the quantum random walk on six-qubit position state on the number line between IBM’s *Qasm_simulator* and Rigetti’s *QVM* 36

5.4 The analysis of wall-time from running the quantum random walk on seven-qubit position state on the number line between IBM’s *Qasm_simulator* and Rigetti’s *QVM* 39

5.5 The increasing rate of the computing wall-time from the circuit depth in Figure 5.32 50

LIST OF FIGURES

FIGURE **PAGE**

1.1 Example quantum circuit 2

2.1 Quantum circuit for creating a maximally entangled two-qubit state 7

2.2 Coupling map of IBM 16-qubits 8

2.3 Coupling map of IBM 5-qubits 8

2.4 Quantum circuit for the establishment of entanglement between 2 qubits with Qiskit 10

2.5 The connectivity (topology) of 16 qubits Aspen-1 QPU 10

2.6 Quantum circuit for the establishment of entanglement between 2 qubits with PyQuil 12

2.7 MCT-basic with $q0_0, q0_1, and\ q0_2$ as control qubits, $q0_3$ as an ancilla, and $q0_4$ as a target qubit 14

2.8 MCT- basic-dirty-ancilla with $q1_0, q1_1, and\ q1_2$ as control qubits, $q1_3$ as an ancilla, and $q1_4$ as a target qubit (the second part of the circuit is continued from the first part) 14

2.9 MCT-advanced with $q2_0, q2_1, and\ q2_2$ as control qubits, $q2_3$ as an ancilla, and $q2_4$ as a target qubit 14

2.10 MCT- no-ancilla with $q3_0, q3_1, and\ q3_2$ as control qubits, $q3_3$ as an ancilla, and $q3_4$ as a target qubit(the second part of the circuit is continued from the first part) 15

3.1 One-dimensional quantum random walk 17

4.1 n-qubit incrementer 19

4.2 A three-qubit incrementer gate 20

4.3 Four-qubits incrementer with notation $(n: 0)$ 20

4.4 Four-qubits incrementer circuit reconstruction with Toffoli, CNOT and NOT gates 21

LIST OF FIGURES (Cont'd)

FIGURE **PAGE**

4.5 Four-qubits incrementer circuit with notation (n:n-1:RE) after reducing two equivalent adjacent quantum gates 21

4.6 Four-qubits incrementer circuit with notation (n:n-1: RD) 21

4.7 n-qubit decrementer circuit 22

4.8 Four-qubits incrementer circuit 23

4.9 Four-qubits decrementer circuit 24

4.10 The conditional shift operator for four-qubit state 24

4.11 The probability distribution of one-dimensional quantum random walk with Hadamard coin after 100 walks starting at position 0 25

4.12 The number line with 2^8 possible vertices 25

4.13 The initial set up of walking with an eight-position state and one coin state 26

4.14 The coin and condition shift operator for walking on the number line with eight-position states. The vertical dashed line separates the coin and the shift operator 26

4.15 The probability distribution of one-dimensional quantum random walk with Hadamard coin after 100 walks starting from the 0 position 27

5.1 The number line of a one-dimension quantum random walk with four qubits position state 29

5.2 The initial set up of walking with a four-position state and one coin state 29

5.3 The conditional shift operator for four-qubit state and the position of MCT modification (in the square) 30

5.4 The probability distributions of four-qubit position state one-dimensional quantum random walk with 7 walking steps 30

5.5 The number line of a one-dimension quantum random walk with five qubits position state 32

LIST OF FIGURES (Cont'd)

FIGURE **PAGE**

5.6 The initial set up of walking with a five-position state and one coin state 32

5.7 The coin and condition shift operator for walking on the number line with five-position states; the vertical dashed line separates the coin and the shift operator 32

5.8 The probability distributions of five-qubit position state one-dimensional quantum random walk with 15 walking steps 33

5.9 The number line of a one-dimension quantum random walk with six qubits position state 34

5.10 The initial set up of walking with a six-position state and one coin-state 35

5.11 The coin and condition shift operator for walking on the number line with six-position states. The vertical dash line separated between coin and shift operator 35

5.12 The probability distributions of six-qubit position state one-dimensional quantum random walk with 31 walking steps 36

5.13 The number line of a one-dimension quantum random walk with seven qubits position state 37

5.14 The initial set up of walking with a seven-position state and one coin-state 37

5.15 The coin and condition shift operator for walking on the number line with seven-position states. The vertical dash line separates the coin from the shift operator 38

5.16 The probability distributions of seven-qubit position state one-dimensional quantum random walk with 63 walking steps 38

5.17 The peak of position -43 from seven-qubit position state walk 39

5.18 The computing wall-time from Qiskit and Rigetti quantum computer simulator 40

LIST OF FIGURES (Cont’d)

FIGURE **PAGE**

5.19 The circuit depth and computing wall-time of 4-qubit position state for each type of conditional shift operator 41

5.20 The circuit depth and computing wall-time of 5-qubit position state for each type of conditional shift operator 42

5.21 The circuit depth and computing wall-time of 6-qubit position state for each type of conditional shift operator 42

5.22 The circuit depth and computing wall-time of 7-qubit position state for each type of conditional shift operator 43

5.23 The circuit depth and computing wall-time of 4-qubit position state for each type of conditional shift operator after unrolling the circuit 44

5.24 The circuit depth and computing wall-time of 5-qubit position state for each type of conditional shift operator after unrolling the circuit 44

5.25 The circuit depth and computing wall-time of 6-qubit position state for each type of conditional shift operator after unrolling the circuit 45

5.26 The circuit depth and computing wall-time of 7-qubit position state for each type of conditional shift operator after unrolling the circuit 45

5.27 The circuit depth and computing wall-time of Toffili-CNOT-NOT conditional shift operator for each size of qubit position state 46

5.28 The circuit depth and computing wall-time of MCT-Basic conditional shift operator for each size of qubit position state 47

5.29 The circuit depth and computing wall-time of MCT-Basic-Dirty-Ancilla conditional shift operator for each size of qubit position state 47

5.30 The circuit depth and computing wall-time of MCT-Advanced conditional shift operator for each size of qubit position state 48

5.31 The circuit depth and computing wall-time of MCT-NoAncilla conditional shift operator for each size of qubit position state 48

LIST OF FIGURES (Cont'd)

FIGURE **PAGE**

5.32 The conclusion of a relation between circuit depth and computing wall-time for each type of conditional shift operator 49

5.33 The probability distribution of four-qubits position state from each computation 51

CHAPTER 1 INTRODUCTION

In this chapter, we give an introduction to the thesis. First, a brief description of the thesis, including an addressed issue, related works from a quantum random walk, the significance of the study, and an experimental method, is described. Second, four thesis objectives are illustrated. Last, we clarify the scope of study, including a selected algorithm, quantum computing platforms, and significant parameters.

1.1 Statement of the Problem

According to Moore's Law, the number of transistors on an integrated circuit will doubly increase every two years [1]. The trend seemed to be consistent with the law until several years ago. The computing power from classical computers is reaching its limit because the size of transistors integrated into a circuit is close to the atomic size of silicon. Some researchers have been trying to solve this problem. Hassan, et al. [2] stated that there are other computing technologies, including optical computing / computers, molecular computing / computers, DNA computing / computers, and quantum computing / computers. Since 1998, several quantum computers have been developed using various qubit technologies such as Superconducting qubit, Trapped ion qubit, and Photonic qubit [3]. According to Quantum Computing Report [3], the superconducting qubit is the most widely adopted qubit technology. More importantly, the superconducting qubit is suitable for large scale integration of the quantum computer [4]. The superconducting material exhibits superconductivity properties at very low temperatures. With the benefit of superconducting qubit technologies, a variety of companies and organizations maintained the quantum computing devices based on this technology. Both IBM and Rigetti construct their quantum computers using the superconducting qubits technology. Both of them provide a quantum computer simulator and a channel to interact with their quantum computers. However, different quantum computer simulators and provided frameworks might affect experimental results and other variables of the computation.

Among the numerous quantum computer technologies that have been researched, quantum algorithms have been proposed to perform quantum computation. The "Quantum Algorithm Zoo" [5] website presents a collection of quantum algorithms in multiple categories. Many quantum algorithms have been proven to have better performance than classical algorithms [6]. The Quantum random walk is an interesting tool that can also be used to construct other quantum algorithms [7]. For instance, to find k equal numbers from the list of given M elements, the element distinctness algorithm [8] uses the quantum random walk to move among the formulated Hamming graph vertices and perform a check algorithm. Quantum random walk can be categorized into discrete and continuous quantum random walks. Both types have been broadly adopted by several researchers. The discrete quantum random walk algorithms can be applied to various kinds of problems. Roy, et al. [9] developed a quantum algorithm using the discrete quantum random walk to solve a clustering problem. The results show that the the proposed algorithm speeds up exponentially over existing classical algorithms. In addition, the continuous quantum walk can be employed in multiple algorithms. Muklen, et al. [10] constructed an algorithm that travels among the vertices on a one-dimensional ring with additional bounds, using the continuous quantum walk. They showed that the traveling among the vertices can be faster than classical algorithm when provided appropriated additional bounds.

According to Venegas-Andraca [11], a book described the quantum random walk from the computer scientist's perspective. Also, it demonstrated the advantage of the study on a one-dimensional quantum random walk on the line with the following three reasons. Firstly, the one-dimensional quantum random walk on the line can be used to formulate a walk on more complex graphs such as a circle or general graph. Secondly, the one-dimensional quantum random walk on the line can be a simple model to study the quantum random walk properties, which will have a positive impact on a quantum algorithm construction. Thirdly, the one-dimensional quantum random walk on the line can be a quantumness testing tool for quantum computing devices.

In this thesis, we study a model of a one-dimensional quantum random walk. Quantum computing circuits are implemented and tested on quantum computer simulators from IBM and Rigetti. Then, the results and computing wall-time (which is the time spent only in the quantum simulator computation excluding the communication and SDK's compile time) of the two quantum computer simulators are compared. The effect of the different architectures is also inspected. Moreover, the relationship between the circuit depth and the computing wall-time is studied. In conducting this study, it was assumed that the computing wall-time depends on the circuit depth. For IBM's quantum computer simulator, computing wall-time is extracted from the return value after execution. For the Rigetti, computing wall-time is acquired from a simulator's log. The circuit depth can be computed by counting the highest number of gates that apply on the same qubit and counting the gate that has to perform before. For example, the function will return 9 for the circuit in figure 1.1. However, the circuit depth can be acquired from IBM's *depth()* function.

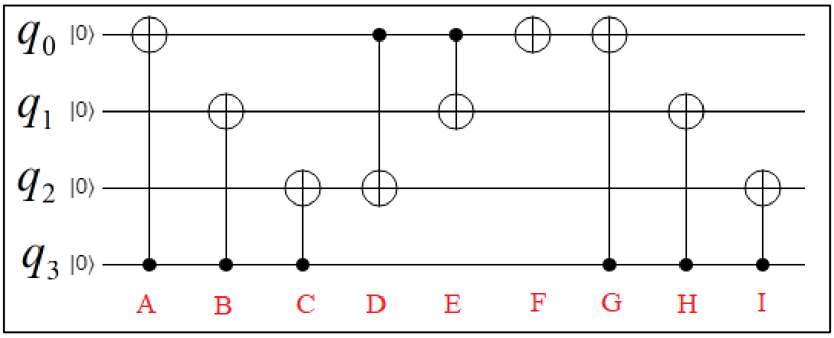


Figure 1.1 Example quantum circuit

Experiments on the real quantum computer are also conducted. To find the real quantum computer's capability, similar circuits from the simulators will be computed on the real quantum computer from IBM as Rigetti's quantum computer cannot be accessed. The probability distributions are then collected.

1.2 Objectives

1. To study the discrete quantum random walk.
2. To study programming on different quantum computers.
3. To study the effect and performance of the different quantum computer architectures upon implementation of a one-dimensional quantum random walk.
4. To study the relationship between computing wall-time and circuit depth of one-dimensional quantum random walk circuits.

1.3 Scope

In this thesis, the one-dimensional quantum random walk is studied. The quantum circuits are developed and implemented on the quantum computer and quantum computer simulators using software development kits provided by IBM and Rigetti. The computing wall-time and circuit depth of the quantum circuits tested on the different quantum computers simulators are scrutinized.

The rest of this thesis is organized as follows: Chapter 2 presents the basic of quantum computing and quantum computer programming from IBM and Rigetti. Chapter 3 gives a simple explanation of the classical random walk and describes the one-dimensional quantum random walk. The demonstration of the shift operator and circuit implementation are provided in Chapter 4. Chapter 5 contains experiments on our quantum circuits, including 5 sizes of one-dimensional quantum random walk and the investigation on the Multi-Control Toffoli function. Lastly, we offer our conclusion in Chapter 6.

CHAPTER 2 BACKGROUND STUDY ON QUANTUM COMPUTING

In this chapter, we introduce three subjects of background knowledge. First, we briefly introduce quantum mechanics, including qubits, quantum entanglement, and quantum measurement. Second, we describe quantum gates and circuits. Last, we explain two quantum computers (from IBM and Rigetti) including their architecture and resources for programming.

2.1 Quantum Mechanics

While data in classical computing is represented by a series of bits, a set of qubits is used in quantum computing. A classical bit can represent only one of two possible values at a time (either 0 or 1), but a qubit can represent the two values at the same time.

A state of a qubit can be denoted by a Ket notation, “$|\quad\rangle$”. Therefore, the qubit in state 0 and 1 is represented by $|0\rangle$ and $|1\rangle$, respectively. The state that a qubit contains two possible values at the same time, called “superposition”, is shown in (2.1)

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

where Ψ denotes any qubit state. α and β can be arbitrary complex numbers, but they must satisfy $|\alpha|^2 + |\beta|^2 = 1$. The state of the qubit can also be represented by matrix notation as follows:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix} \text{ and } |1\rangle = \begin{bmatrix}0\\1\end{bmatrix} \tag{2.2}$$

Information in a qubit can be processed by a unitary operator that performs a unitary transformation. The unitary operator U must satisfy the following property:

$$UU^{\dagger} = U^{\dagger}U = I, \tag{2.3}$$

where I is an identity matrix and $U^{\dagger}$ is a transpose complex conjugate of U.

Equation 2.5 presents a unitary transformation on a qubit, where U is a unitary matrix of the one-qubit quantum operator.

$$|\Psi\rangle \rightarrow U|\Psi\rangle = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} U_{00}\alpha + U_{01}\beta \\ U_{10}\alpha + U_{11}\beta \end{pmatrix} \tag{2.5}$$

To operate more than one qubit, dimensions of the unitary matrix can be increased by a tensor product as shown in (2.6).

$$|\Psi_1\Psi_2\rangle = |\Psi_1\rangle \otimes |\Psi_2\rangle = \alpha_1\alpha_2|00\rangle + \beta_1\alpha_2|10\rangle + \alpha_1\beta_2|01\rangle + \beta_1\beta_2|00\rangle \tag{2.6}$$

Quantum entanglement is one of the most significant quantum phenomena. When two or more qubits are entangled, the state of qubits cannot be specified separately. However, the quantum entanglement property is not used in this thesis. Equation 2.7 shows an example of the two entangled qubits:

$$\begin{aligned} |\Psi_A\rangle &= \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \\ |\Psi_B\rangle &= \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \\ |\Psi_c\rangle &= \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \\ |\Psi_D\rangle &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \end{aligned} \tag{2.7}$$

Quantum measurement is a process of readout information from the quantum state. When a qubit, in (2.1), is measured, for example, the superposition state $|\Psi\rangle$ will be collapsed to either 0 with probability $|\alpha|^2$or 1 with probability $|\beta|^2$. The result of the measurement is then a classical bit of information, 0 or 1. In measurement, $\{|0\rangle, |1\rangle\}$ is used as a measurement basis. To measure multiple qubits, the measurement basis needed to be defined. If $\{|00\rangle, |10\rangle, |01\rangle, |11\rangle\}$ is our measurement basis for the two joint qubits, four possible outcomes will be given with probabilities $|\alpha_1\alpha_2|^2$, $|\beta_1\alpha_2|^2$, $|\alpha_1\beta_2|^2$ and $|\beta_1\beta_2|^2$, respectively. Therefore, the measurement of n- multiple qubits will result in 2^n possible outcomes.

2.2 Quantum Circuits

Classical gates such as *AND*, *OR*, and *NOT* are used to implement a classical computation circuit. Similarly, quantum gates are also used to construct a quantum circuit. The quantum gates are unitary transformation U that can be described by unitary matrices.

The Hadamard gate corresponds to Hadamard transformation that produces the superposition state from the computational basis. If the computational basis is $\{|0\rangle, |1\rangle\}$, applying Hadamard to $|0\rangle$ and $|1\rangle$ will transform the state to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, respectively.

The set of Pauli operators includes Pauli-I, Pauli-X, Pauli-Y, and Pauli-Z. Pauli-I is the identity transformation represented by the identity matrix. Pauli-X or a bit-flip operator will map $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$. Pauli-Z or a phase flip operator will change the phase of the qubit state from $|1\rangle$ to -$|1\rangle$, but it does nothing to $|0\rangle$. Pauli-Y is a combination of Pauli-X and Pauli-Z. Then, Pauli-Y performs a bit-flip and phase-shift that maps $|0\rangle$ to $i|1\rangle$ and $|1\rangle$ to $-i|0\rangle$.

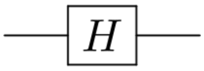
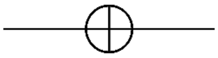
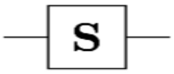
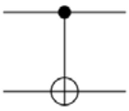
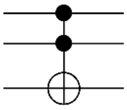
The rotation operators are a set of parameterized single-qubit operators. The set of rotation operators includes $R_x(\theta)$, $R_y(\theta)$, and $R_z(\theta)$ that rotates around the x, y, and z axis, respectively, on a Bloch sphere. The angle θ must be given to specify a rotation angle. It can be positive or negative but must be in radians.

Regarding two-qubit gates, the *CNOT* or *controlled-NOT* gate operates on two qubits. One qubit, called a control qubit, controls the operation, while another is a target qubit. If the state of the control qubit is $|1\rangle$, the target qubit will be flipped. The control phase gate is a two-qubit gate that the target qubit makes a change when the control qubit is $|1\rangle$; the operating is that done to the target qubit is a phase manipulation. The swap gate is also a two-qubit gate that swaps the state of the two qubits. The gate maps $|01\rangle$ to $|10\rangle$ and $|10\rangle$ to $|01\rangle$.

The Toffoli gate is a 3-qubit quantum gate. Like *CNOT*, the 3 operated qubits can be separated into control and target qubits. Two qubits are the control qubits and the other is the target qubit. The target qubit is flipped when all control qubits are in the state $|1\rangle$. Otherwise, the Toffoli gate does nothing.

Quantum gate can be represented in graphical notation and matrix, as illustrated in Table 2.1. The quantum circuit can be constructed by assigning any quantum gate to qubits. As shown in figure 2.1, *Hadamard* and *CNOT* are assigned to 2 qubits. The circuit creates entanglement between the two-qubits.

Table 2.1 Commonly used quantum gates

Gate name	Graphical notation	Unitary Matrix representation
Hadamard gate		$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$
Pauli-X or NOT gate		$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
S gate		$\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$
CNOT Gate		$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\\0 & 1 & 0 & 0\end{bmatrix}$
Toffoli Gate		$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\\0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}$

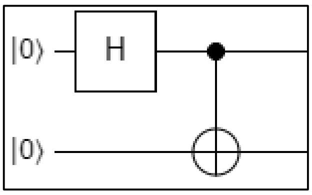


Figure 2.1 Quantum circuit for creating a maximally entangled two-qubit state

2.3 Quantum Processing Unit and Programming

Several techniques can be used to construct a quantum computer, such as Optical[12], trapped ions[13-15], and superconducting qubit. Superconducting qubit technology is widely adopted by many large companies such as Intel, Google, IBM, and Rigetti. Some of them have published documents and have made computing services available for public use and research. Among the available superconducting qubit quantum computers, IBM and Rigetti were selected to be studied in this thesis for 3 main reasons. Firstly, both companies have actual quantum computers. Secondly, the same quantum circuit can compute on simulators and real quantum computers of both companies. Thirdly, their quantum computing platforms are provided with all pertinent documents.

2.3.1 IBM

IBM is one of the major companies that has been intensively involved in the field of quantum computing. According to quantum computing devices [16], IBM has five quantum computers and one quantum simulator. Both can be operated with open quantum assembly language (openQASM)[17]. Their 4 quantum computers that are open to access contain 5 and 16 qubits. The other 20-qubit quantum computer is only for IBM's clients. In superconducting qubit technology, the qubit is controlled by conducting microwave pulses. The connection of qubits is subjected to a coplanar waveguide (CPW) resonator for readout and control. CPW is a transmission line that can transfer microwave-frequency signals.

To apply a circuit of quantum gates, the system has a limitation on a physical quantum computer. As shown in Figures 2.2 and 2.3, IBM [18-19] the arrows connected between qubit nodes indicate the frequency signal of a qubit. An arrow is pointing from a higher frequency qubit to a lower frequency qubit. The difference in frequency produces the limitation of deploying quantum gates. IBM suggested that using a higher frequency bit as a controlled bit can prevent the inverse of the operation. For example, if we want to deploy the CNOT gate to the quantum computer in Figure 2.3, the qubit at position 0 should be used as a control qubit while the qubit at position 1 can be a target. However, the quantum circuit setup will be managed and optimized by the framework.

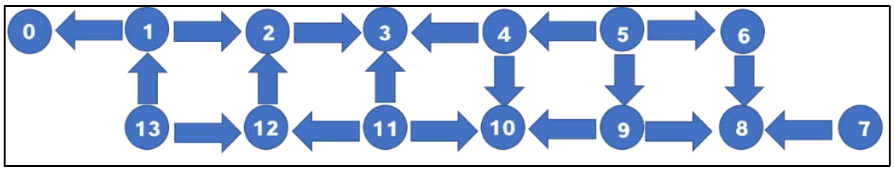


Figure 2.2 coupling map of IBM 16-qubits

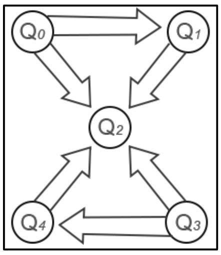


Figure 2.3 coupling map of IBM 5-qubits

IBM provides a software development kit (SDK) called Quantum Information Software Kit (Qiskit). Qiskit is an open-source framework that communicates between quantum computers and quantum programs. The programming language used for communicating with Qiskit can be Python, JavaScript, and swift. In this thesis, we use Python language.

Qiskit consists of four parts: Terra, Aqua, Ignis, and Aer. Each of them is responsible for different jobs as follows.

1. Terra is the foundation of the framework. The input of Terra is in a level of circuit and pulses. Terra will optimize the input due to the constraint of the physical quantum processor and will also manage the execution.
2. Aqua is an element that contains a library of several quantum algorithms and multiple extensions such as Qiskit Optimization, Qiskit Chemistry, and Qiskit Finance.
3. Ignis is designed to understand and mitigate noise in quantum circuits and devices. It also has tools to measure noise parameters, and to generate and analyze the circuits.
4. Aer is the high-performance simulator framework. It helps Terra to optimize and comply with circuits. It can also perform a realistic noise simulation from quantum computer noise models.

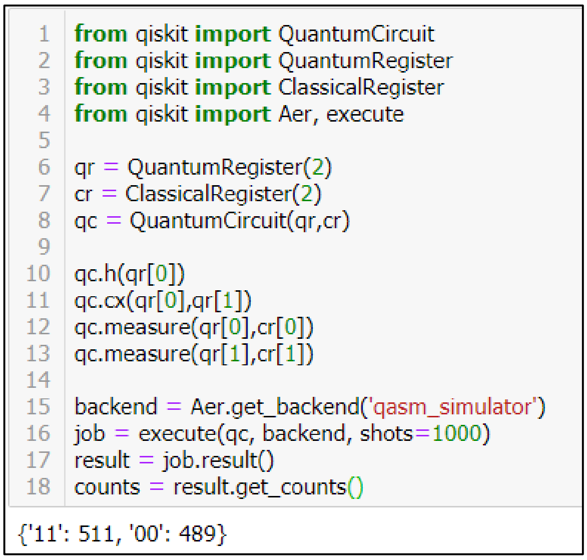
In order to use Python with Qiskit (which is similar to writing a regular Python program) the related libraries have to be imported. The following list presents the primary Qiskit libraries.

1. *QuantumRegister* defines a number of the required qubits.
2. *ClassicalRegister* defines a number of the required classical bits.
3. *QuantumCircuit* is for a circuit construction with the parameters from *QuantumRegister* and *ClassicalRegister*. The collections of quantum gates are bundled with *QuantumCircuit*.
4. *Aer* uses for a quantum computer simulator selection. 3 simulators, namely *Qasm_simulator*, *Statevector*, and *Unitary* are available.
    - *Qasm_simulator* simulates multiple shots of quantum computation on the circuit and returns counts. The noise model of the quantum computer backend can be applied to the *Qasm_simulator* for the computation under the specific noise behavior.
    - *Statevector* provides a single-shot execution of the circuit and returns a final state vector.
    - *Unitary* also provides a single-shot execution. The result of the *Unitary* simulator is the unitary matrix of the circuit.
5. *Execute* is a function for quantum circuit execution.

Figure 2.4 presents an example source code of the quantum circuit implementation with Qiskit. After importing the libraries (lines 1-4), a quantum circuit can be constructed by adding quantum gates to the *QuantumCircuit* variables (lines 10-13). To compute the circuit, a simulator has to be selected. In this example, *Qasm_simulator* is chosen to simulate a quantum computation.

2.3.2 Rigetti

Rigetti, which is another company involved in the field of quantum computing, aims to provide quantum cloud services. Currently, Rigetti has two operating quantum devices, 8-qubit and 16-qubit [20]. Similar to IBM, Rigetti’s devices are also constructed with superconducting technology, but are different in the topology as shown in Figure 2.5.



```
from qiskit import QuantumCircuit
from qiskit import QuantumRegister
from qiskit import ClassicalRegister
from qiskit import Aer, execute

qr = QuantumRegister(2)
cr = ClassicalRegister(2)
qc = QuantumCircuit(qr,cr)

qc.h(qr[0])
qc.cx(qr[0],qr[1])
qc.measure(qr[0],cr[0])
qc.measure(qr[1],cr[1])

backend = Aer.get_backend('qasm_simulator')
job = execute(qc, backend, shots=1000)
result = job.result()
counts = result.get_counts()
```

```
{'11': 511, '00': 489}
```

Figure 2.4 Quantum circuit for the establishment of entanglement between 2 qubits with Qiskit

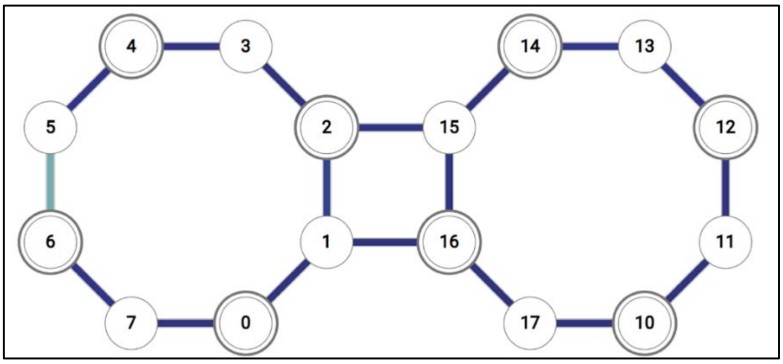


Figure 2.5 The connectivity (topology) of 16 qubits Aspen-1 QPU

Forest is Rigetti’s SDK for the communication between the quantum processing unit (QPU) and Quil [21]. Different from QISKit, Rigetti adopts Quil programming language. The Forest SDK did not separate into elements like QISKit, but it has three essential components.

1. PyQuil is a Python library that helps to construct a Python program with Quil programming language.
2. Quil compiler is responsible for translating Quil into the instruction code that can operate with any Rigetti quantum machine.
3. The quantum virtual machine can simulate a quantum computer and execute Quil on a classical computer.

To program with Forest, the major libraries, including *Program, get_qc,* and *set of quantum gates*, have to be imported.

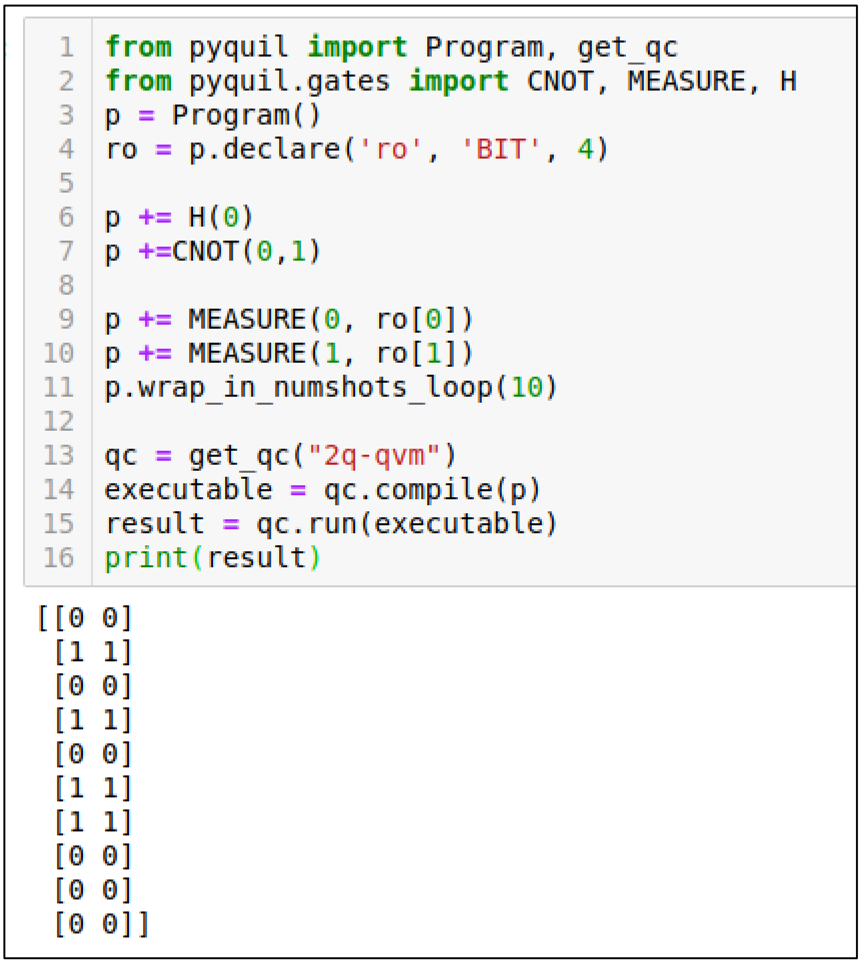
1. *Program* contains a list of instructions to compose a quantum program such as a quantum gate concatenation, classical bit declaration, a measurement, and running shot specification.
2. The *get_qc* function is responsible for selecting a quantum computer or specifying the size and topology of the quantum computer simulator.
3. For a *set of quantum gates,* unlike Qiskit, the required quantum gates should be specified separately. A quantum circuit can be constructed by concatenating quantum gates into the *Program*'s variables.

To compute a quantum circuit, Quil compiler (Quilc) and quantum virtual machine (QVM) have to start. Quil compiler and QVM operate as a computing server for their task. Because of the quantum gate limitation on the quantum computer, the set of quantum gates in the circuit has to be complied with the gate that able to operate on a quantum computer or simulator. QVM, then, computes the compiled version of the quantum circuit from Quilc and returns the measurement result of each shot to the program. Figure 2.6 presents the source code of a quantum circuit implementation with PyQuil. After importing the libraries (lines 1-2), a variable *p* is declared as a space for quantum gates (line 3). A classical register is then declared (line 4). The quantum circuit is constructed by adding quantum gates (lines 6 – 10). The number of measurement shots is set to 10 (line 11). Since this program requires 2 qubits, the 2-qubit quantum computer simulator *2q-qvm* is selected (line 13).

Both IBM and Rigetti quantum computers support a common quantum gate set as follows:

1. Pauli gates including Pauli-I, Pauli-X, Pauli-Y, and Pauli-Z
2. Hadamard gate
3. Toffoli gate
4. Phase gates
5. Control phase gate
6. Controlled X gate
7. Swap gate

As mentioned above, both IBM and Rigetti provide a quantum computer simulator. However, the operating techniques of the two platforms are different. *Qasm_simulator* operates on C++, but *QVM* operates on LISP. In addition to their default configuration, multiple elements in *Qasm_simulator* and *QVM* are configurable. In this thesis, only the default configuration is used.



```
from pyquil import Program, get_qc
from pyquil.gates import CNOT, MEASURE, H
p = Program()
ro = p.declare('ro', 'BIT', 4)

p += H(0)
p +=CNOT(0,1)

p += MEASURE(0, ro[0])
p += MEASURE(1, ro[1])
p.wrap_in_numshots_loop(10)

qc = get_qc("2q-qvm")
executable = qc.compile(p)
result = qc.run(executable)
print(result)
```

```
[[0 0]
 [1 1]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [0 0]]
```

Figure 2.6 Quantum circuit for the establishment of entanglement between 2 qubits with PyQuil

LaRose [22] compared the quantum computing resources from big companies in the quantum computing area, including Microsoft, Rigetti, IBM, and ProjectQ. His work described the software library support, quantum hardware, quantum compiler, and simulator performance. The list of quantum algorithms, which were added to each company's software library, was as well presented. *IBMQX5* from IBM and *Agave* from Rigetti, which were the largest publicly available quantum computers at that time, were selected for discussion. Regarding the actual quantum computers, each quantum computer has a basic set of quantum gates and connectivity. The quantum compiler is an operator that compiles program source code into a quantum circuit for a specific quantum computer and simulator. Again, only compilers from IBM and Rigetti were discussed. The discussion indicated that the two platforms have a different basic set of gates. IBM's quantum computer supports U_1, U_2, U_3, and *CNOT* while Rigettti's quantum computer supports R_x, R_z, and *Controlled-Z (CZ)*. Finally, this research compared the performance of quantum computer simulators from IBM and ProjectQ. A state vector simulator from IBM and C++ simulator from ProjectQ were chosen. The comparison described the performance of circuit depth, the number of qubits, and computing wall-time. Unlike that work, this thesis compares circuit depth and computing wall-time between *Qasm_simulator* from IBM and *QVM* from Rigetti. The quantum circuits depend on the type and size of the one-dimensional quantum random walk.

2.4 IBM Single Qubit Gate and Multi-Control Toffoli Function

According to Y. Aharonov [23], the IBM quantum computing platform operates on the implementation of a Controlled-NOT gate and three parameterized single-qubit gates, including U_1, U_2 and U_3. The U_1 gate is a one-parameter gate. The parameter can be assigned to the gate by $U_1(\lambda)$. An operation of U_1 is equivalent to the rotation gate $R_z(\lambda)$. U_2 is a two-parameter gate. The parameters can be assigned to the gate by $U_2(\phi,\lambda)$. An operation of U_2 is equivalent to the set of rotation gate $R_z(\phi+\frac{\pi}{2})R_x(\frac{\pi}{2})R_z(\lambda-\frac{\pi}{2})$. U_3 is a three-parameter gate. The parameters for U_3 includes θ, ϕ, and λ that can be assigned to the gate by $U_3(\theta,\phi,\lambda)$. An operation of U_3 gate is equivalent to the set of rotation gate $R_z(\phi+3\pi)R_x(\frac{\pi}{2})R_z(\theta+\pi)R_x(\frac{\pi}{2})R_z(\lambda)$. A major difference among these three gates is how the microwave pulse is applied. U_1 can change a phase of qubit without applying any pulses. The U_2 gate applies single $\frac{\pi}{2}$ pulse. For U_3, two $\frac{\pi}{2}$ pulses are applied for making an operation. However, both of $U_1$and U_2 can be converted into the U_3 gate as follows.

$$U_1(\lambda) := U_3(0,0,\lambda) \tag{2.8}$$

$$U_2(\phi,\lambda) := U_3(\frac{\pi}{2},\phi,\lambda) \tag{2.9}$$

Besides, any single-qubit gate can also be transformed into U_3. For example:

$$Pauli-X \ := U_3(\pi,0,\pi) \tag{2.10}$$

$$Hadamard \ := U_3(\frac{\pi}{2},0,\pi) \tag{2.11}$$

$$S\ gate \ := U_3(0,0,\frac{\pi}{2}) \tag{2.12}$$

Moreover, Qiskit offers a useful function named multi-controlled-Toffoli (MCT). The MCT function works as applying the Toffoli operation with multiple control qubits. MCT consists of four main parameters: a list of control qubits, the target qubit, a list of ancilla, and mode of use. The modes of use in the MCT function are basic, basic-dirty-ancilla, advanced, and no-ancilla. The different modes require different numbers of ancilla. After applying the function, the set of quantum operations will be generated based on the selected mode and input parameters.

For the basic mode, four main parameters must be determined. A member of ancilla's list must be $|0\rangle$. The number of ancillae must be equal to $x-2$, where x is the number of control qubits. The MCT with basic mode will generate a circuit that consists of U_1, U_2, $CNOT$ and $Toffoli$ gate, as shown in Figure 2.7.

For the basic-dirty-ancilla mode, the required parameters are similar to the basic mode. However, a state of ancilla is not necessary to be $|0\rangle$. The number of required ancilla is still similar to the basic mode. The applying of MCT with basic mode will generate a circuit that consists of U_1, U_2, and $CNOT$ gate, as illustrated in Figure 2.8.

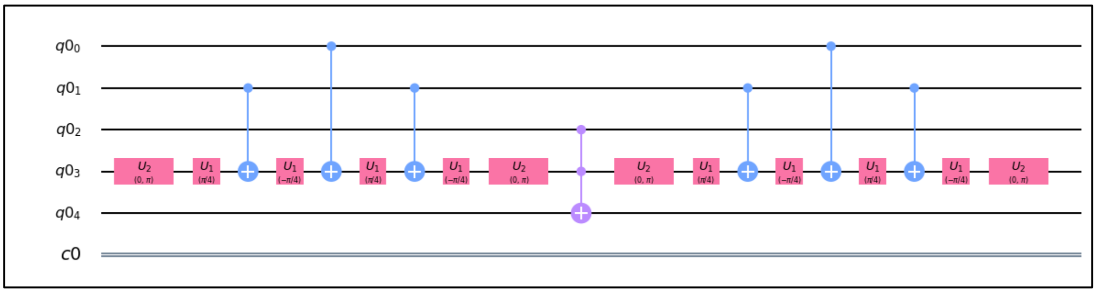


Figure 2.7 MCT-basic with $q0_0, q0_1, and\ q0_2$ as control qubits, $q0_3$ as an ancilla, and $q0_4$ as a target qubit

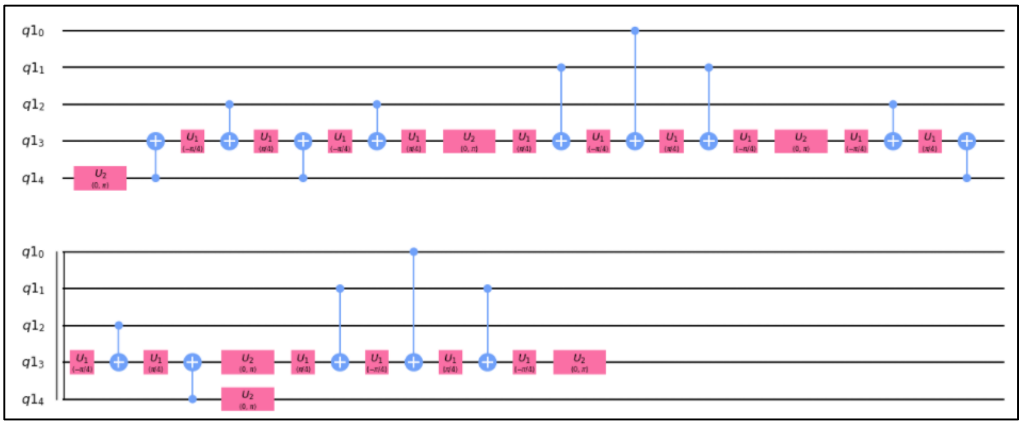


Figure 2.8 MCT- basic-dirty-ancilla with $q1_0, q1_1, and\ q1_2$ as control qubits, $q1_3$ as an ancilla, and $q1_4$ as a target qubit (the second part of the circuit is continued from the first part)

In the advanced mode, the required number of parameters remains four. However, the number of ancilla qubits will be $x - 4$, where x is the number of control qubits. Still, the state of ancilla doesn't have to be $|0\rangle$. The application of MCT with basic mode will generate a circuit that consists of $Hadamard$, $Controlled - U_1$, and CNOT gate, as displayed in Figure 2.9.

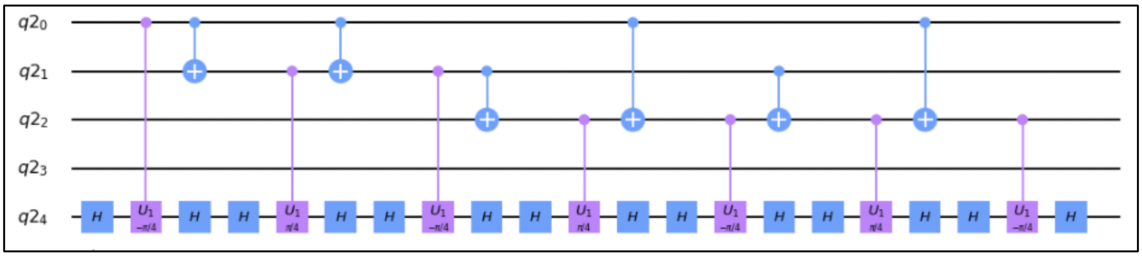


Figure 2.9 MCT-advanced with $q2_0, q2_1, and\ q2_2$ as control qubits, $q2_3$ as an ancilla, and $q2_4$ as a target qubit

With the no-ancilla mode, the required number of parameters decreases to three. As the name of the mode, this mode requires zero ancillae. Then, the remaining three parameters include a list of control qubits, the target qubit, and the mode of use. The (application /

applying) of MCT with basic mode will generate a circuit that consists of $Hadamard$, U_1, and CNOT gate, as shown in Figure 2.10.

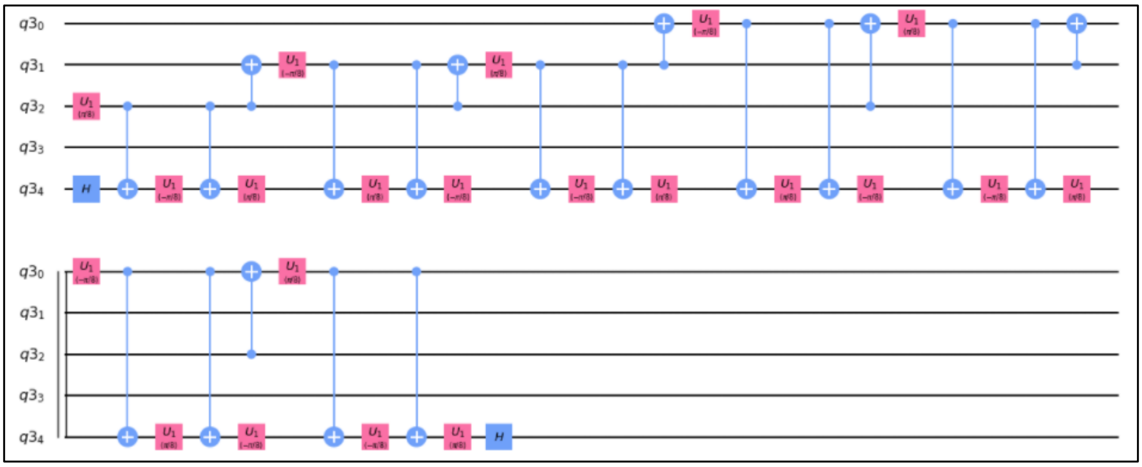


Figure 2.10 MCT- no-ancilla with $q3_0, q3_1, and\ q3_2$ as control qubits, $q3_3$ as an ancilla, and $q3_4$ as a target qubit (the second part of the circuit is continued from the first part)

CHAPTER 3 QUANTUM RANDOM WALKS

This chapter introduces the one-dimensional quantum random walk. The concept of quantum random walk and the relevant research literature are reviewed. Subsequently, the one-dimensional quantum random walk is focused. Main operators and quantum state evolution are lastly presented.

3.1 Quantum Random Walk

A quantum random walk is a quantum computing version of a classical random walk. The quantum random walk can be classified into two categories: discrete-time quantum random walk and continuous-time quantum random walk. According to Venegas-Andraca [11], the main difference between the two categories is the times at which evolution operators are applied. The evolution operator is a set of operators that applies change to the system of quantum random walk. Details of the evolution operator will later be described in this section. The evolution operators are applied in discrete time steps in a discrete quantum random walk, while it can be applied anytime in a continuous quantum random walk.

The concept of the quantum random walk was first introduced in 1993 by Aharonov, et al.[23]. They also demonstrated that, with the nature of quantum characteristics, the average length of the walking path on a line can be longer than the length a classical random walk can produce. This quantum random walk algorithm was later considered a discrete model. After this work, the quantum random walk has gained more attention. Kempe [24] overviewed the development of quantum walks for quantum computing. The work described both discrete and continuous quantum random walks and compared them to a classical random walk. Her work also presented the physical quantum random walk implementation point of view. Lovett, et al. [25] implemented a universal quantum computation gate from the discrete quantum random walk algorithm. They also proved that both continuous and discrete quantum random walks could be efficiently used to simulate other quantum algorithms.

This thesis focuses only on the discrete quantum random walk on a one-dimensional graph. The main structure of the discrete quantum random walk consists of a walker, a coin, and the evolution operators. Similar to the classical random walk, a walker walks on the graph. The possible walking paths are the concatenated vertices and edges. The coin is a state that is considered a decision-maker space. The next step of the walk to any of the concatenated vertices and edges can be determined by the coin state. The evolution operators can be separated into two parts, including coin and shift operators. The coin operator is a function that operates the coin state. Similarly, the shift operation operates the state of the walker. The shift operator considers the change of the walker state from the coin state that was manipulated by the coin operator. The two operations are repeatedly operated without performing a measurement. The measurement is only performed at the final position of the walking process. Figure 3.1 displays the flowchart of the discrete quantum random walk on the one-dimensional graph.

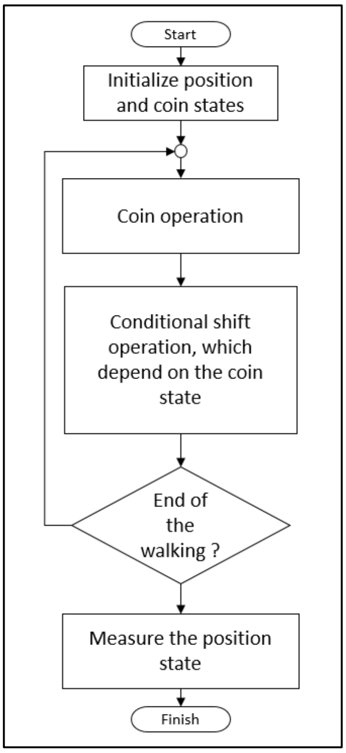

Start
Initialize position and coin states
Coin operation
Conditional shift operation, which depend on the coin state
End of the walking ?
Measure the position state
Finish

Figure 3.1 One-dimensional quantum random walk

The entire state of the quantum walk H consists of two Hilbert spaces, $H_p$and H_c, in the Hilbert space that $H = H_p \otimes H_c$, where H_p represents the Hilbert space of the walker's position state and H_c represents the Hilbert space for the coin state.

The walker is a quantum state that occupies a space H_p. The H_p spanned with basis $|i\rangle_p$ where i is a position on the one-dimensional graph (the number line). To initialize the position, the origin of the walker is usually at position zero on the number line that can be represented as $|0\rangle_p$. The coin state basis, however, can be spanned by $\{\ |0\rangle_c,\ |1\rangle_c\}$, in which the number of basis is equal to the number of edges connected to the vertex in the number line.

The application sequence of the evolution operators on each step of the walk is a coin operator followed by a conditional shift operator. In our experiments, the coin operator is performed using the Hadamard gate that transforms the coin state into a superposition of its state, as shown in the following equations.

$$C(\,|0\rangle_c) \rightarrow \frac{1}{\sqrt{2}}(\,|0\rangle_c + \,|1\rangle_c) \tag{3.1}$$

$$C(\,|1\rangle_c) \rightarrow \frac{1}{\sqrt{2}}(\,|0\rangle_c - \,|1\rangle_c) \tag{3.2}$$

where C is the coin operator.

Next, the conditional shift operator operates based on the state of coin space, which could be defined as:

$$S(\,|0\rangle_c \otimes |i\rangle_p) \longrightarrow \,|0\rangle_c \otimes |i+1\rangle_p \tag{3.3}$$

$$S(\,|1\rangle_c \otimes |i\rangle_p) \longrightarrow \,|1\rangle_c \otimes |i-1\rangle_p \tag{3.4}$$

where S is the conditional shift operator.

Accordingly, Equation 3.5 presents the evolution operators on one step of the walk.

$$U = S(C \otimes I_p) \tag{3.5}$$

where U is the evolution operators of the one-dimensional quantum random walk, S represents the conditional shift operator, C represents the coin operator, and I represents the vector space of the position state. The tensor operator $\otimes$ is used to state that C applies only the coin state without direct action to the position state. For example, if the walk start with $|\Psi\rangle_0 = |0\rangle_c \otimes |0\rangle_p$, two walking step with Hadamard as a coin operator can be derived as follows:

$$|\Psi\rangle_1 = U(|0\rangle_c \otimes |0\rangle_p) \tag{3.6}$$

$$|\Psi\rangle_1 = S\left(\frac{1}{\sqrt{2}}(\,|0\rangle_c + \,|1\rangle_c) \otimes |0\rangle_p\right) \tag{3.7}$$

$$|\Psi\rangle_1 = \frac{1}{\sqrt{2}}\,|0\rangle_c|1\rangle_p + \frac{1}{\sqrt{2}}\,|1\rangle_c|-1\rangle_p \tag{3.8}$$

$$|\Psi\rangle_2 = S\left(\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(\,|0\rangle_c + \,|1\rangle_c)\right)|1\rangle_p + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(\,|0\rangle_c - \,|1\rangle_c)\right)|-1\rangle_p\right) \tag{3.9}$$

$$|\Psi\rangle_2 = \frac{1}{2}\,|0\rangle_c|2\rangle_p + \frac{1}{2}\,|1\rangle_c|0\rangle_p + \frac{1}{2}\,|0\rangle_c|0\rangle_p - \frac{1}{2}\,|1\rangle_c|-2\rangle_p \tag{3.10}$$

CHAPTER 4 SHIFT OPERATOR FOR QUANTUM RANDOM WALK

In the one-dimensional quantum random walk, the shift operator is a key function that manipulates the walker's position state to go forward or backward according to the condition of the coin state. In this chapter, we firstly describe a family of quantum incrementer circuits derived by Li, et al. [26]. After that, we design a shift operator circuit based on the idea of Li. Lastly, we verify the implementation of our designed shift operator circuit.

4.1 Incrementer Circuits

Li, et al. [26] introduced the first quantum incrementer circuit with the notation $(n\colon 0)$ where n denotes the number of qubits, as shown in Figure 4.1. The operation of the incrementer is to increment the value of the input state by one. The gates presented in the figure are the control-flip gates where the "$\oplus$" signs represent the target qubit, and the black dots represent the control qubits. The target qubit will be flipped if every quantum state of the connected black-dot is $|1\rangle$.

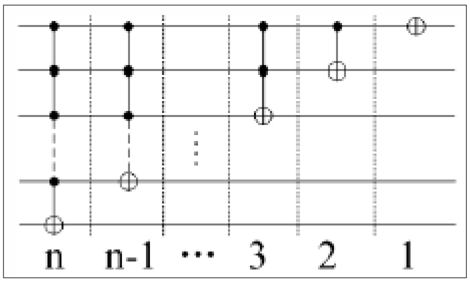


Figure 4.1 n-qubit incrementer

They also gave an example of a simple three-qubit incrementer circuit, as displayed in Figure 4.2. After passing through the circuit, the input qubits $|A_2A_1A_0\rangle$ become $|{A_2}^+{A_1}^+{A_0}^+\rangle$. The state of $|{A_2}^+{A_1}^+{A_0}^+\rangle$ is obtained from Equations (4.1)-(4.3) where "¬", "*", and "+" represent Boolean logic NOT, AND, and XOR gates, respectively. The results of all possible three-qubit states are displayed in Table 4.1.

$${A_2}^+ = A_2 + A_1 * A_0 \tag{4.1}$$

$${A_1}^+ = A_1 + A_0 \tag{4.2}$$

$${A_0}^+ = \neg A_0 \tag{4.3}$$

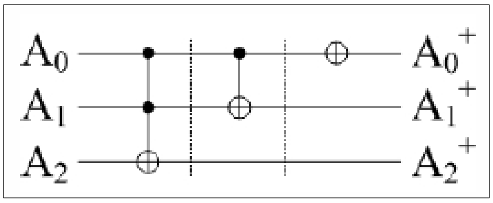


Figure 4.2 A three-qubit incrementer gate

Table 4.1 The truth table of the three-qubit incrementer circuit

Input	$\|000\rangle$	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$
Output	$\|001\rangle$	$\|010\rangle$	$\|011\rangle$	$\|100\rangle$	$\|101\rangle$	$\|110\rangle$	$\|111\rangle$	$\|000\rangle$

The quantum gate with more than two-control qubits, however, does not exist in the common quantum computing operators. Thus, they generalized the incrementer $(n{:}\,0)$ circuit into two circuit types, namely $(n{:}\,n-1{:}\,RE)$ and $(n{:}\,n-1{:}\,RD)$, that are constructed from the standard quantum computing gates, including NOT, CNOT, and Toffoli gates. Note that for the notion $(N{:}\,M{:}\,RE/RD)$: N denotes the number of qubits, M denotes the number of carrying bits, and RE/RD denotes the enabled/disabled reset carry qubit.

The $(n{:}\,n-1{:}\,RE)$ circuit is generalized from the $(n{:}\,0)$ circuit using the "full ancilla qubits" technique. The main idea of this technique is to add one ancilla qubit between every two control qubits. For the four-qubit incrementer circuit as an example, Figure 4.3 illustrates the $(n{:}\,0)$ circuit where $n = 4$ that consists of, from left to right, three-control-not, Toffoli, CNOT, and NOT gates. Figure 4.4 presents the reconstructions of the four-qubit incrementer circuit for the $(n{:}\,n-1{:}\,RE)$ notation using only Toffoli, CNOT, and NOT gates. It can be seen in the figure that the input qubits $|A_3A_2A_1A_0\rangle$ become "$|{A_3}^+{A_2}^+{A_1}^+{A_0}^+\rangle$" where C_i represents the ancilla qubit with $i = 0{,}1{,}2$. However, due to the unitary properties of quantum gates, the two adjacent mirror-images quantum gates can be canceled out. The final circuit after the cancelation is then shown in Figure 4.5.

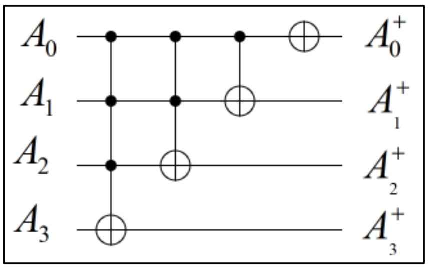


Figure 4.3 Four-qubits incrementer circuit with notation (n:0)

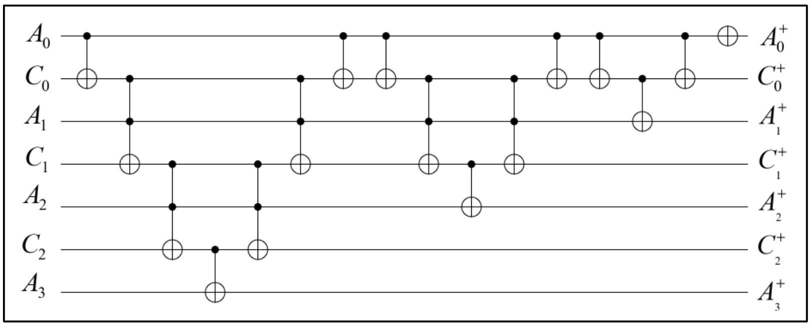


Figure 4.4 Four-qubits incrementer circuit reconstruction with Toffoli, CNOT and NOT gates

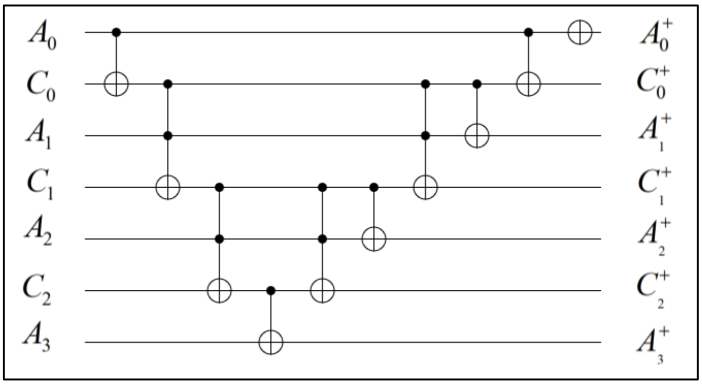


Figure 4.5 Four-qubits incrementer circuit with notation (n:n-1:RE) after reducing two equivalent adjacent quantum gates

The ancilla qubits in the $(n{:}\, n-1{:}\, RE)$ circuit will always be reset to $|0\rangle$ after being transformed by CNOT or Toffoli gates. However, the ancilla qubits in the $(n{:}\, n-1{:}\, RD)$ circuit will not be reset. The four-qubit incrementer circuit with the $(n{:}\, n-1{:}\, RD)$ notation is shown in Figure 4.6.

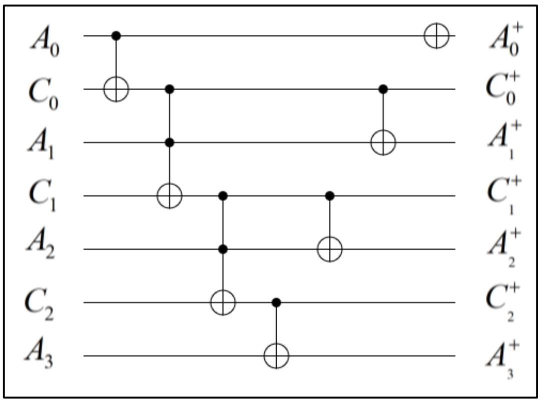


Figure 4.6 Four-qubits incrementer circuit with notation (n:n-1: RD)

According to X. Li, et al. [26], the comparison between the three notations was presented, as shown in Table 4.2.

Table 4.2 The comparison among the notation

Type of the circuits	$(n{:}0)$	$(n{:}n-1{:}RE)$		$(n{:}n-1{:}RD)$
		Before reduce	reduced	
Amount of gates	n	n^2	$3n-2$	$2n-1$
Number of carried(ancilla) qubits	0	$n-1$	$n-1$	$n-1$
Time Complexity	N/A	$O(n^2)$	$O(n)$	$O(n)$

In addition, Li, et al. [26] briefly mentioned a decrementer circuit that was constructed from the multiple-control-not gates as displayed in Figure 4.7. The target qubits at the "$\oplus$" sign will be flipped when the qubits state at all connected white dots is $|0\rangle$.

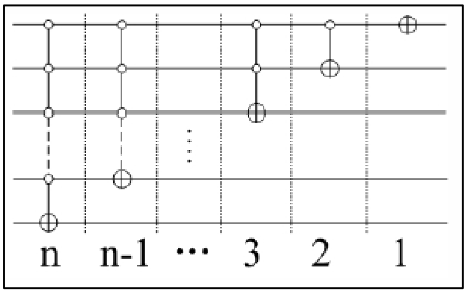


Figure 4.7 n-qubit decrementer circuit

4.2 A Design of Shift Operator for Quantum Random Walk

In accordance with the incrementer and decrementer design presented by X. Li, et al. [26], in this section we present the construction of the shift operator with the properties of forward, backward, and conditional operator.

In order to build the forward, our proposed incrementor is newly designed based on the idea of the incrementor with the notation $(n{:}n-1{:}RE)$. However, there are three main differences between our design and the incrementor with the notation $(n{:}n-1{:}RE)$. Firstly, in our design, the first 3 qubits can be placed next to each other without having an ancilla qubit in between. Secondly, the number of required ancilla will only be increased when the number of qubits is more than three. Thirdly, the number of required gates were less than the previous approach. The comparison between our approach and $(n{:}n-1{:}RE)$ is concluded in Table 4.3. The incrementor circuit can be constructed as shown in Figure 4.8.

Table 4.3 The comparison between incrementer and incrementer in this thesis

Type of the circuits	$(n{:}\, n-1{:}\, RE)$ reduced	In this thesis
Amount of gates	$3n-2$	$3n-6$
Number of carried (ancilla) qubits	$n-1$	$n-3; n \geq 4$ 0 otherwise
Time Complexity	$O(n)$	$O(n)$

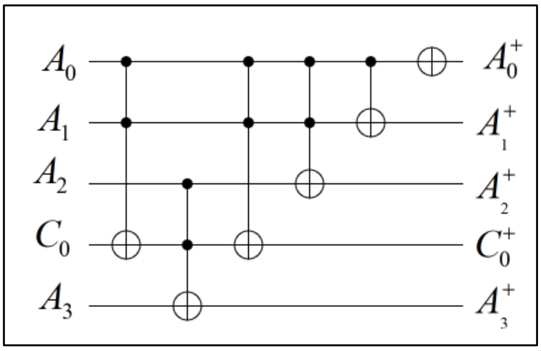


Figure 4.8 Four-qubits incrementer circuit

For the backward, we apply a decrementer circuit to the shift operator. As shown in Figure 4.7, the decrementer is the opposite of the incrementer circuit. However, adding the decrementer to the previous incrementer on the same circuits can generate a considerable depth of the circuit. The idea of the classical complement operator is then adopted.

The classical complement operator can be classified into one's complement and two's complement. The one's complement is the operation that inverts all bits in a bit string (turn 0s to 1s and vice versa). Two's complement is the operator that performs one's complement process and adds the output by one. Equations (4.4)-(4.9) describe an example of decrementing 0001 that should result in 0000:

X = 0001 (4.4)

X'1 complement = 1110 (4.5)

X'2 complement = X'1 complement +1 (4.6)

X'2 complement = 1110 +1 (4.7)

X'2 complement = 1111 (4.8)

If we one's complement all bits again, the result will be 0000. (4.9)

As the two's complement has an addition by one, it then can take advantage of our incrementor circuit. For each of complement, we used the NOT gate that applies on all qubits. Then, the decrementer circuit can be constructed as shown in Figure 4.9. The gates in the red boxes are responsible for the complement operations.

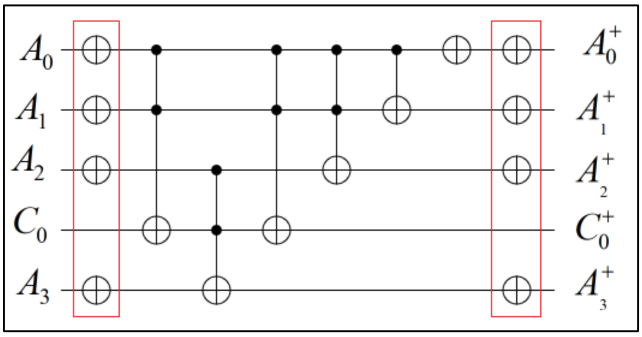


Figure 4.9 Four-qubits decrementer circuit

Further, the additional operation is applied by replacing the NOT gate in the complement operation with the CNOT gate where the coin qubit is the control qubit. As aforementioned, the shift operator of the quantum random walk will be operated based on the state of the coin. When the coin state is $|0\rangle$, the condition of CNOT will not be applied and the shift circuit will operate only the increment. When the coin state is $|1\rangle$, on the other hand, the circuit operates the decrementer by initially flipping the target qubits, performing the increment, and eventually flipping the qubits again. Our design of the shift operator circuit is displayed in Figure 4.10, where $Coin$ represents the coin state.

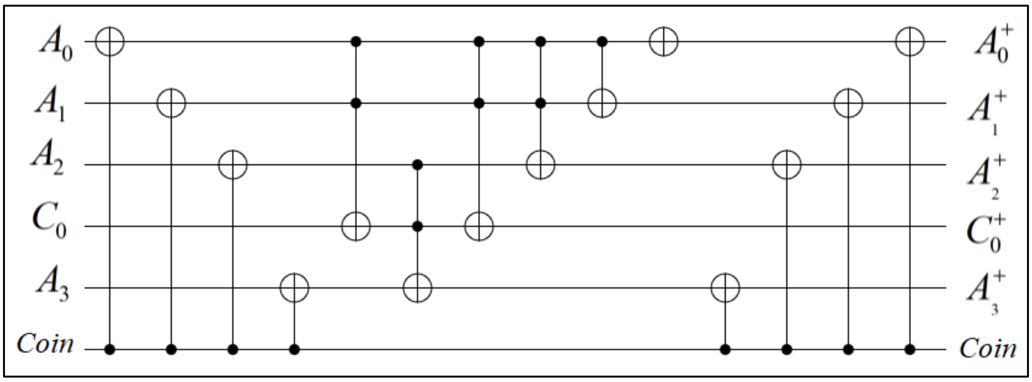


Figure 4.10 The conditional shift operator for four-qubit state

4.3 Shift Operator Circuit Verification

To verify our design of the conditional shift operator in the previous section, the quantum random walk on the number line is simulated. The probability distribution from the measured final position state will be compared with the probability distribution from [11] as shown in Figure 4.11.

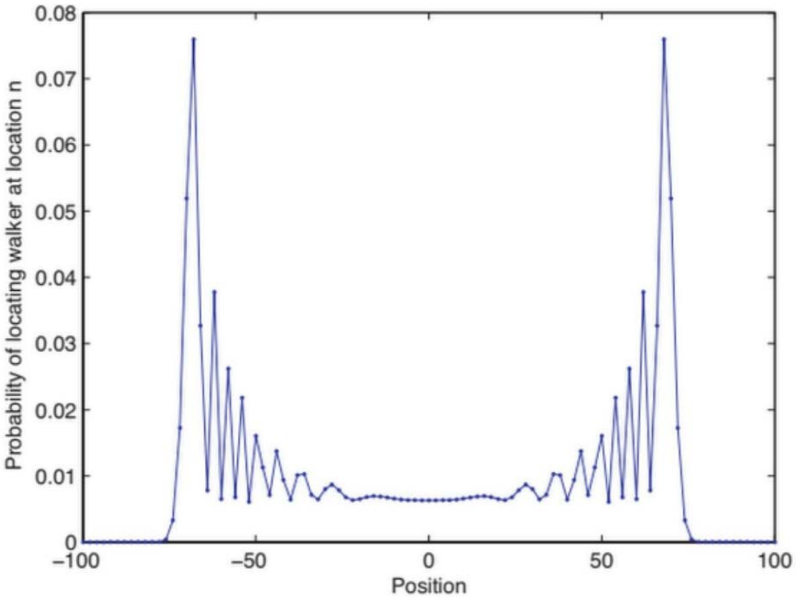


Figure 4.11 The probability distribution of one-dimensional quantum random walk with Hadamard coin after 100 walks, starts at position 0

As displayed in Figure 4.11, the walking space required at least 100 positions to the left and right of the starting point. The number line with possible 2^8 positions is then defined as shown in Figure 4.12. Thus, eight qubits of the position state are needed along with one qubit of the coin state. The experiment was run on the IBM's quantum computer simulator, "*Qasm_simulator*."

Initially, the walker is placed at position 0 on the number line, which was mapped to the $|01111111\rangle_p$ position state, as shown in Figure 4.12. Figure 4.13 presents the initial setup of the walk, where the $|01111111\rangle_p$ position state can be mapped to the qubits as $|q_7, q_6, q_5, q_4, q_3, q_2, q_1, q_0\rangle_p$. The set of NOT gates are applied on the specific qubits to initialize the starting point of the walker. The C_0, C_1, C_2, C_3, and C_4 in Figure 4.13 are the carried or ancilla qubits. $Coin$ represents space of coin state. The state of the coin is initialized with $\frac{1}{\sqrt{2}}(|0\rangle_c + i|1\rangle_c)$ because the compared distribution is symmetric, as demonstrated in Figure 4.11.

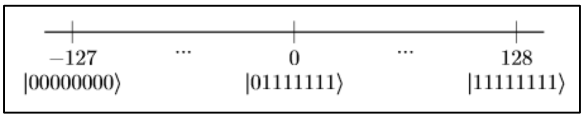


Figure 4.12 The number line with 2^8 possible vertices

The Hadamard gate is selected as an operation of a coin operator because of the two possible directions of the walk. The shift operator is constructed from the design in section 4.2. However, the shift operator was scaled up from the example in Figure 4.10

by adding more position states and ancilla. The circuit with Hadamard coin and shift operator is illustrated in Figure 4.14. During the execution, the coin and shift operators are repeatedly operated one-hundred times, which are equal to the number of walking steps.

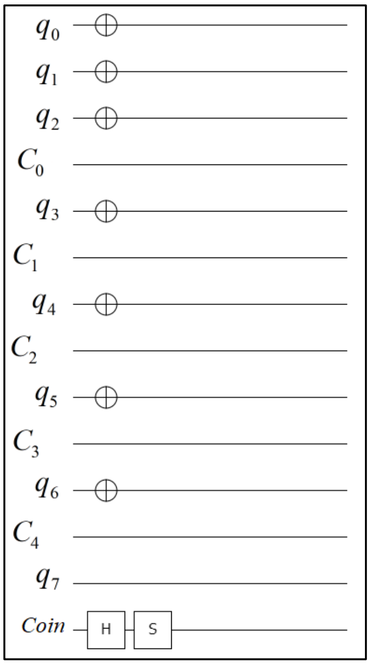


Figure 4.13 The initial set up of walking with an eight-position state and one coin state

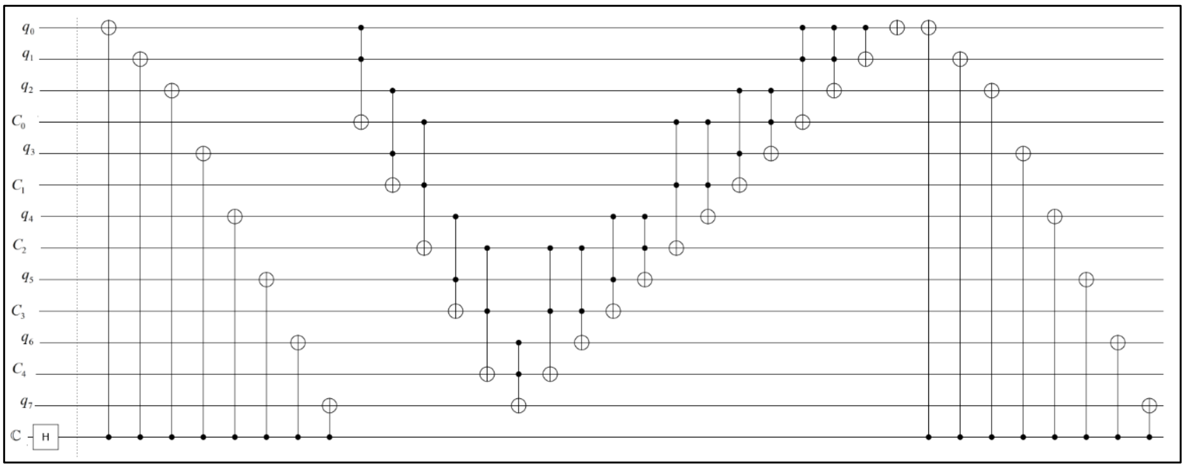


Figure 4.14 The coin and condition shift operator for walking on the number line with eight-position states. The vertical dashed line separates the coin and the shift operator.

Figure 4.15 presents the probability distribution of our circuit implementation. The probability distribution complies with the characteristic of probability distribution in the previous literature [11] as shown in Figure 4.11. Thus, we can conclude that our conditional shift operator circuit can be used to simulate the one-dimensional quantum random walk.

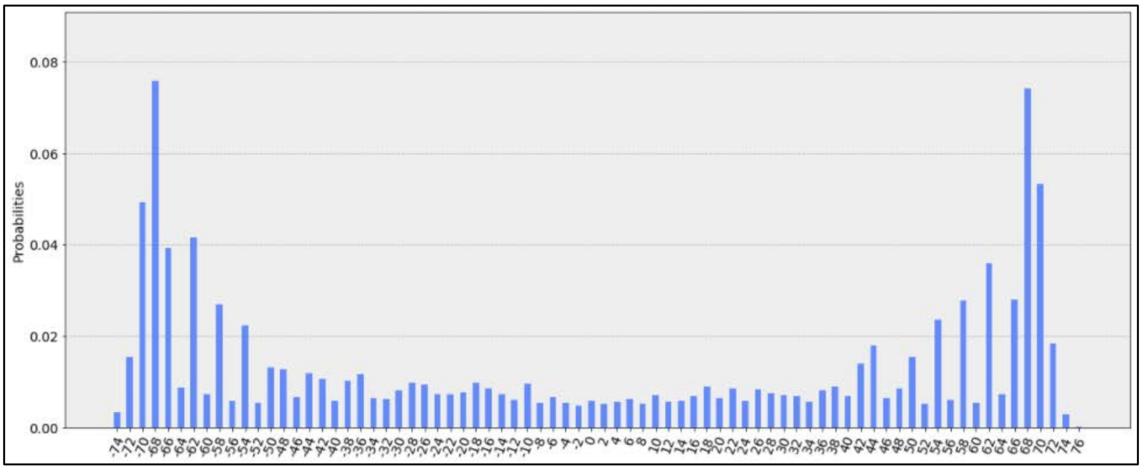


Figure 4.15 The probability distribution of one-dimensional quantum random walk with Hadamard coin after 100 walks starting from the 0 position.

CHAPTER 5 EXPERIMENTAL RESULT AND DISCUSSION

In this section, we present experiments of one-dimensional quantum random walk on quantum simulators and on the IBM's quantum computer. All of the experiments are conducted under the following environments:

- Rigetti Quil Compiler (quilc) version 1.12.1
- Rigetti Quantum Virtual Machine (QVM) version 1.12.0
- IBM Qiskit 0.13.0
  - Qiskit-terra 0.10.0
  - Qiskit-aer 0.3.2
- Python version 3.7.3
- Ubuntu version 19.04
- Virtual Machine Specification:
  - 3 CPU Cores
  - 4 GB of Memory
- Virtual Box Version 6.0.10
- Computer Specification:
  - CPU: Intel I5-6400, 2.70 GHz
  - Memory: 16 GB, DDR 4 1066 MHz

The one-dimensional quantum random walk circuits in the experiments consist of 3 main steps. Firstly, the initial walker and the coin state are set up. Secondly, the unitary operators, including coin and conditional shift operators, are placed with multiple trials depend on the number of walking steps. Lastly, the measurement operators are placed at the last operation of each qubit position state. For all of the experiments, the coin state is always initialized with $\frac{1}{\sqrt{2}}(|0\rangle_c + i|1\rangle_c)$ using the Hadamard and S gates.

Two main experiments have been executed. First, the benchmarking of quantum computer simulators between IBM's *Qasm_simulator* and Rigetti's *QVM* using a one-dimensional quantum random walk. Both of the quantum computer simulators operate under their default configuration. Second, the execution of a one-dimensional quantum random walk on an actual quantum computer.

5.1 A Benchmarking of Quantum Computer Simulators

The benchmarking is tested on the quantum computer simulators from IBM and Rigetti with 4 different sizes, including 4, 5, 6, and 7 qubits position state, of the one-dimensional quantum random walk. For each size, 6 circuits are tested. 2 circuits are made of the common quantum gates from Qiskit's qasm_simulator and Rigetti's *QVM*. The other 4 circuits are built using the 4 different modes of the Qiskit's Multi Control Toffoli function and compute only on *Qasm_simulator*.

5.1.1 A Benchmarking on Four-qubit Quantum Random Walk

For the four-qubit position state, a walker can walk along the $2^4 = 16$ possible positions on the number line as displayed in Figure 5.1. Initially, the walker is placed at position 0 on the number line, which was mapped to the $|0111\rangle_p$ position state. The walking step was set to 7 due to the capacity of the number line. 6 qubits and 4 classical bits are used in this experiment. The position state uses 4 qubits, while the coin state uses 1 qubit. The last qubit is used for an ancilla in the shift operator. As shown in Figure 5.2, q_0, q_1, q_2 and q_3 represent the qubit for the position states, $Coin$ represents the qubit for the coin, and C_0 represents an ancilla. For the initialization process, the NOT gates are applied to the $q_0, q_1,$ and q_2 qubits so that the initial position state becomes $|0111\rangle_p$. The Hadamard and S gates are applied to initial the coin state to $\frac{1}{\sqrt{2}}(|0\rangle_c + i|1\rangle_c)$. Figure. 5.3 illustrates the shift operator for walking on the four-qubit position state.

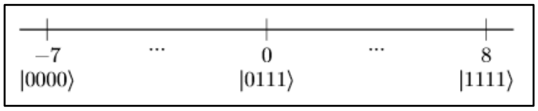


Figure 5.1 The number line of a one-dimension quantum random walk with four qubits position state

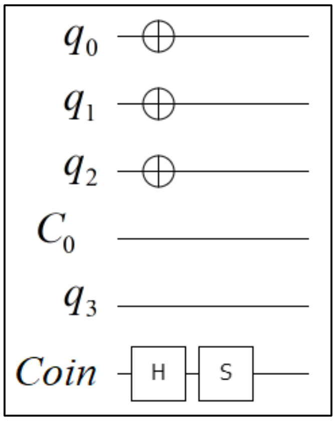


Figure 5.2 The initial set up of walking with a four-position state and one coin state

The one-dimensional quantum random walk circuit is composed of the initialization gates, seven (as the maximum walking steps) sequential sets of the coin and shift operators, and the measurement operators. The measurement operators are only applied to the q_0, q_1, q_2 and q_3 qubits at the very last step. In the modification with MCT, the MCT with 4 modes of operation will replace the circuit in the red square in Figure 5.3. The initialization and the measurement are still the same. To make a reasonable comparison, the walking process is performed 100,000 times on each quantum computer simulator. The computing wall-time, which is the time spent for the computation, is collected at every 1,000 measurement shots. Therefore, 100 computing wall-times are kept.

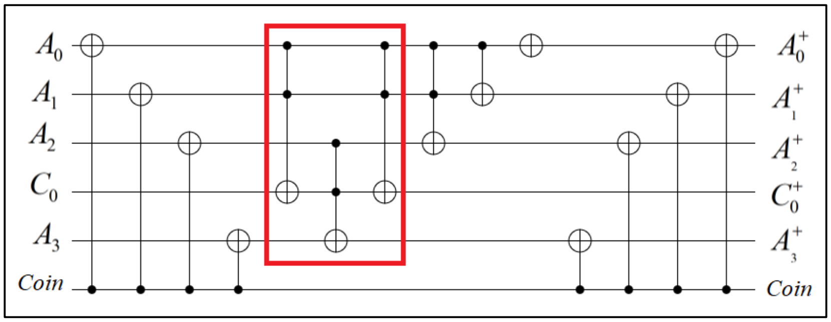


Figure 5.3 The conditional shift operator for four-qubit state and the position of MCT modification (in the square)

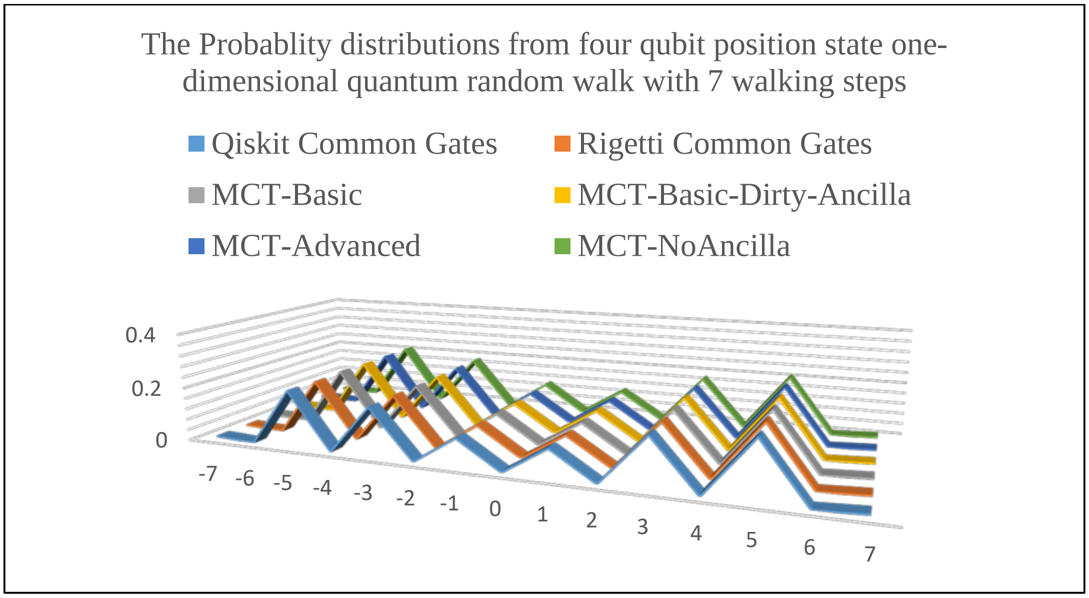


Figure 5.4 The probability distributions of four-qubit position state one-dimensional quantum random walk with 7 walking steps

Figures 5.4 display the probability distributions of the measured results from the 100,000 walking processes performed on the Qiskit's *Qasm_simulator* and Rigetti's *QVM* using the quantum circuits with common gates and and Qiskit's with MCT functions. The results from the measurements show that the probability distributions from the two quantum computer simulators and four-mode of MCT function are similar. Moreover, both probability distributions agree with the theory that the distribution should be symmetrically spread for this experimental setting. However, the computing wall-times on the two quantum computer simulators are significantly different, as shown in Table 5.1. The computing wall-time from the *Qasm_simulator* is remarkably less than the wall-time from the *QVM*. The discussion on this issue will be given in Section 5.1.5.2.

Table 5.1 The analysis of wall-time from running the quantum random walk on four-qubit position state on the number line between IBM's *Qasm_simulator* and Rigetti's *QVM*

	Common quantum gates		MCT			
	Qasm_simulator	QVM	Basic	Basic-Dirty-Ancilla	Advanced	NoAncilla
Mean	**0.093**	**11.395**	**0.166**	**0.239**	**0.207**	**0.291**
S.D.	0.004	0.856	0.012	0.0096	0.007	0.0094
Median	0.092	11.051	0.162	0.237	0.206	0.288
Circuit Depth	99	99	175	316	225	323
Ancilla	1	1	1	1	0	0

5.1.2 A Benchmarking on Five-qubit Quantum Random Walk

In this experiment, the size of the position state is extended to five qubits. Figure 5.5 displays the number line with $2^5 = 32$ walking positions. The maximum number of walking steps is increased to 15 because of the number line expansion. The walker is, again, initialized at position 0 represented by the $|01111\rangle_p$ position state. The coin state at initializing is $\frac{1}{\sqrt{2}}(|0\rangle_c + i|1\rangle_c)$. This experiment requires 8 qubits, as shown in Figure 5.6, where q_0, q_1, q_2, q_3 and q_4 represent the position state, $Coin$ represents the coin state, and C_0, C_1 represent two ancilla qubits. The extra ancilla is required as the shift operator for the five-qubit position state has to be redesigned. The shift operator for the four-qubit position state is expanded by adding one ancilla and multiple Toffoli gates, as demonstrated in Figure 5.7. In this experiment, the one-dimensional quantum random walk circuit for the five-qubit position state is composed of the initialization gates, fifteen duplicated sets of the coin and shift operators, and the measurement operator at the end of the circuit. Again, the MCT with 4 modes of operation will replace the circuit in the red square in Figure 5.7. The initialization and the measurement are still the same. In a similar manner, the walking process is executed 100,000 times, and the computing wall-time is collected at every 1,000 measurement shots.

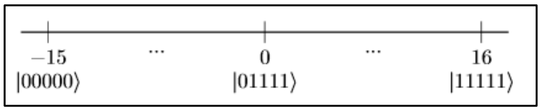


Figure 5.5 The number line of a one-dimension quantum random walk with five qubits position state

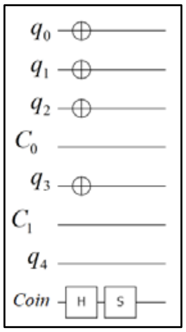


Figure 5.6 The initial set up of walking with a five-position state and one coin state

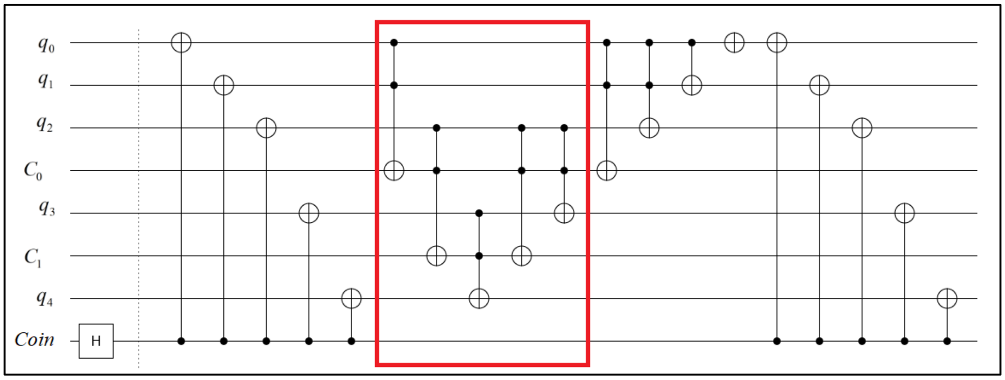


Figure 5.7 The coin and condition shift operator for walking on the number line with five-position states; the vertical dashed line separates the coin and the shift operator

Figures 5.8 displays the probability distributions of the measured results from the 100,000 walking processes performed on the Qiskit's *Qasm_simulator* and Rigetti's *QVM* using the quantum circuits with common gates and and Qiskit's with MCT functions. Same as the previous experiment, the probability distributions from the two quantum computer simulators and four-mode of MCT functions are similar and are consistent with the theory.

The computing wall-times on the two quantum computer simulators, however, are different as shown in Table 5.2. The computing wall-time from the *Qasm_simulator* is still less than the wall-time from the *QVM*. Moreover, according to further investigation into the relationship between circuit depth and computing wall-time, it was found that the circuit composed of the MCT function with Advanced mode spent more computation time than the circuit MCT function with Basic-Dirty-Ancilla mode although the Advanced mode has smaller circuit depth. More details on this issue will be discussed in Section 5.1.5.3.

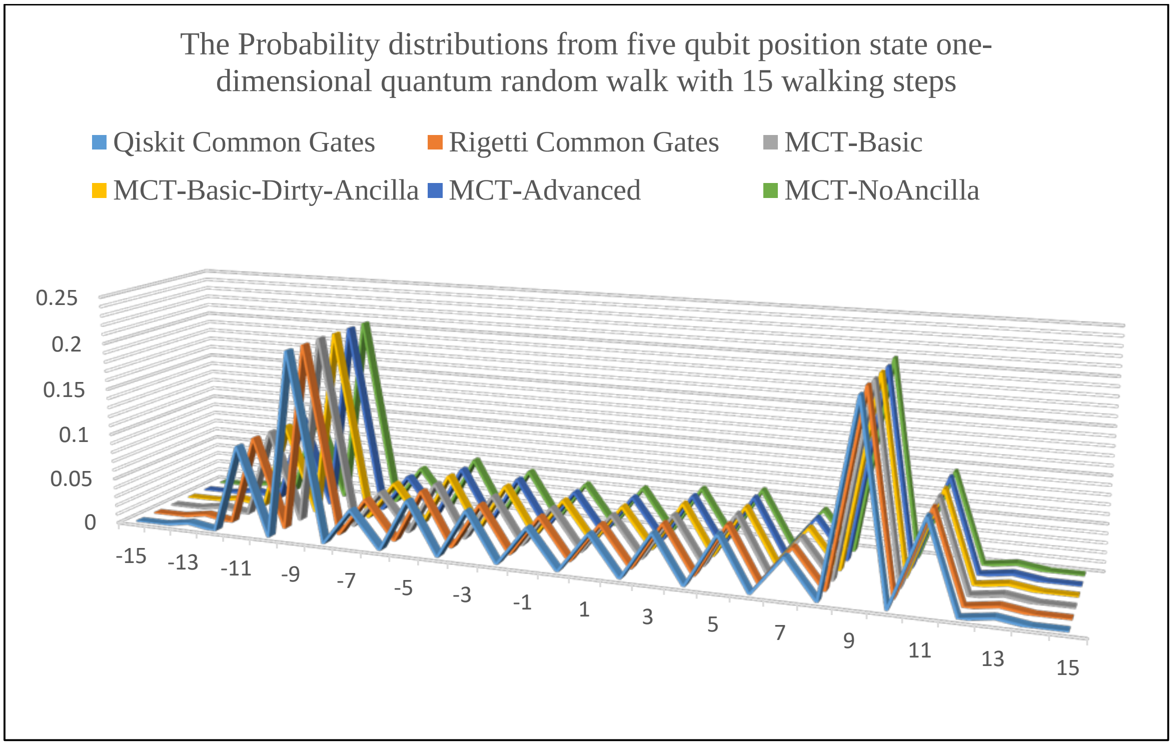


Figure 5.8 The probability distributions of five-qubit position state one-dimensional quantum random walk with 15 walking steps

Table 5.2 The analysis of wall-time from running the quantum random walk on five-qubit position state on the number line between IBM's *Qasm_simulator* and Rigetti's *QVM*

	Common quantum gates		MCT			
	Qasm_simulator	QVM	Basic	Basic-Dirty-Ancilla	Advanced	NoAncilla
Mean	**0.261**	**114.181**	**0.512**	**1.207**	**1.374**	**1.593**
S.D.	0.014	0.962	0.018	0.054	0.035	0.039
Median	0.26	114.074	0.510	1.190	1.367	1.584
Circuit Depth	270	270	885	1605	1380	1770
Ancilla	2	2	2	2	0	0

5.1.3 A Benchmarking on Six-qubit Quantum Random Walk

In this experiment, the position state is increased to six qubits. The space on the number line is also extended to 2^6 walking positions as shown in Figure 5.9. Then, The maximum number of walking steps is increased to 31. The walker starts on position 0 with the $|011111\rangle_p$ position state. The initialization of the coin state is still $\frac{1}{\sqrt{2}}(|0\rangle_c + i|1\rangle_c)$. The total number of a required qubit is 10, including 6 qubits for the position state, 3 qubits for the ancilla, and 1 qubit for the coin. The initialization of the circuit can be seen in Figure 5.10 where $q_0, q_1, q_2, q_3, , q_4$ and q_5 represent the position state, $Coin$ represents the coin state, and $C_0, C_1 and\ C_2$ represent 3 ancilla qubits. Again, the shift operator for the six-position state has to be redesigned by adding ancilla C_2 and multiple Toffoli gates as illustrated in Figure 5.11. Similar to the previous experiments, the MCT with 4 modes of operation will replace the circuit in the red square in Figure 5.11. The initialization and the measurement are the same as before.

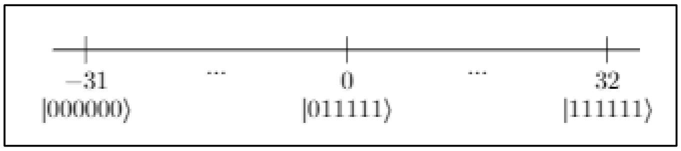


Figure 5.9 The number line of a one-dimension quantum random walk with six qubits position state

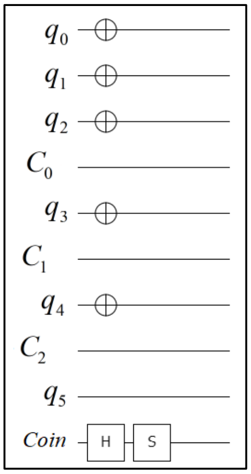


Figure 5.10 The initial set up of walking with a six-position state and one coin-state

Figure 5.12 illustrates the probability distributions of the measured results from the 100,000 walking processes performed on the Qiskit's *Qasm_simulator* and Rigetti's *QVM* using the quantum circuits with common gates and and Qiskit's with MCT functions. Unsurprisingly, the results from the measurements show that the probability distributions from the two quantum computer simulators and four modes of MCT function are similar. Moreover, both probability distributions are consistent with the theory. The computing wall-times on the two quantum computer simulators are still different, as shown in Table 5.3. The computing wall-time from the *Qasm_simulator* is also less than the wall-time from the *QVM*.

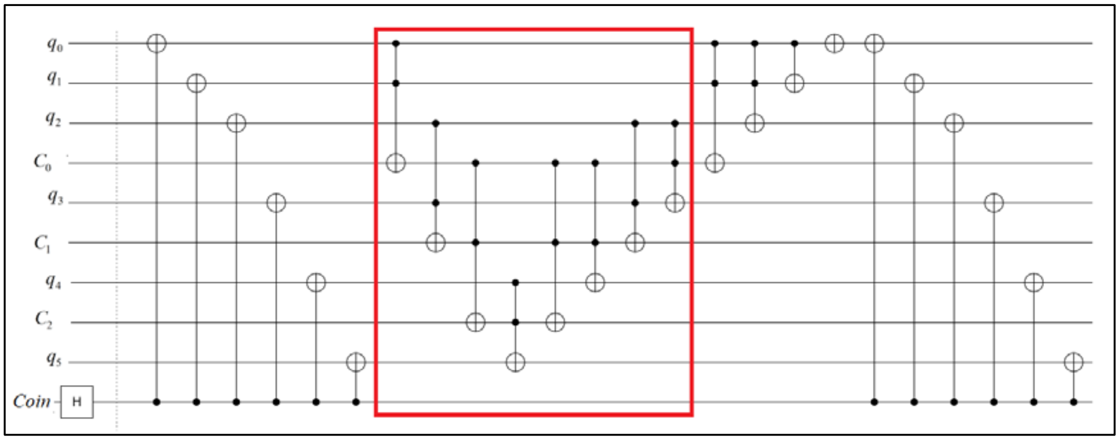


Figure 5.11 The coin and condition shift operator for walking on the number line with six-position states. The vertical dash line separated between coin and shift operator

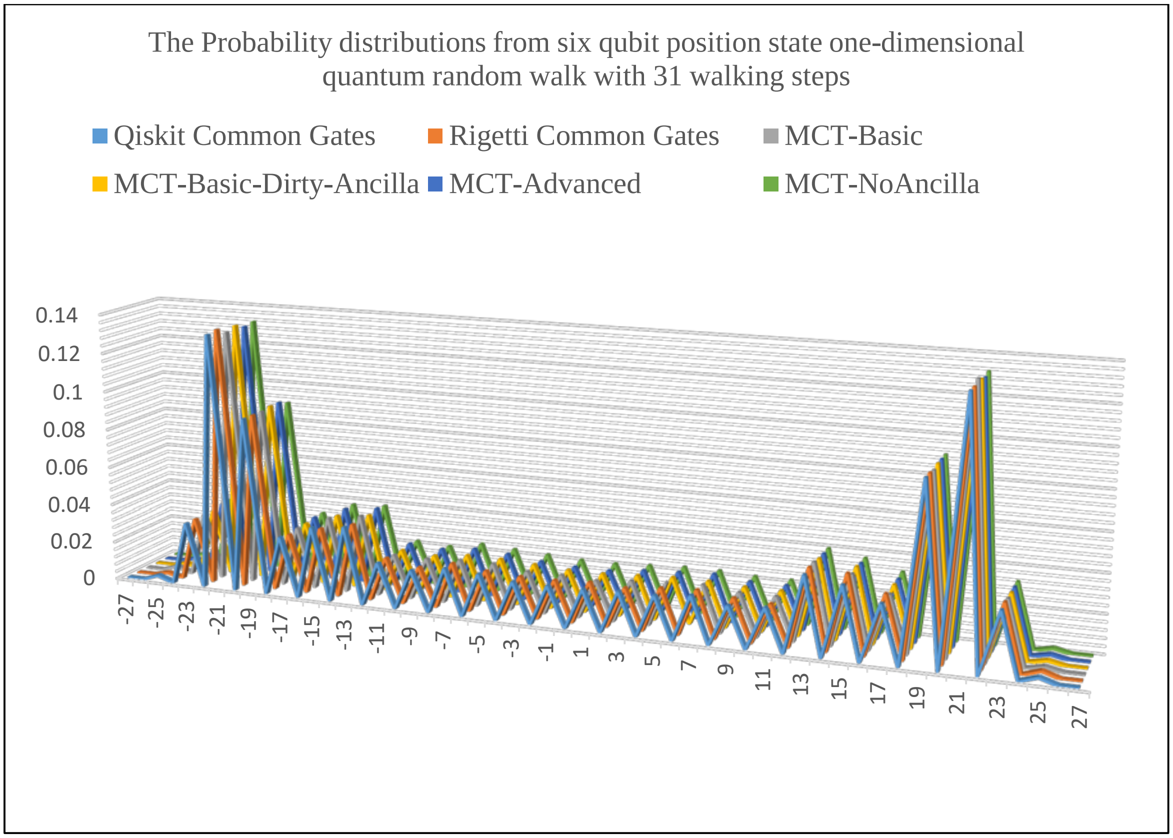


Figure 5.12 The probability distributions of six-qubit position state one-dimensional quantum random walk with 31 walking steps

Table 5.3 The analysis of wall-time from running the quantum random walk on six-qubit position state on the number line between IBM's *Qasm_simulator* and Rigetti's *QVM*

	Common quantum gates		MCT			
	Qasm_simulator	QVM	Basic	Basic-Dirty-Ancilla	Advanced	NoAncilla
Mean	**0.637**	**1617.956**	**1.369**	**4.234**	**5.107**	**7.473**
S.D.	0.021	38.484	0.064	0.162	0.198	0.186
Median	0.633	1596.643	1.345	4.148	5.016	7.442
Circuit Depth	682	682	3317	6108	4619	8370
Ancilla	3	3	3	3	1	0

5.1.4 A Benchmarking on Seven-qubit Quantum Random Walk

In order to achieve robust experimental results, the size of the examination circuit is increased to 7. With the extended size, the total position on the number line is escalated to 2^7, as displayed in Figure 5.13. The number of walking steps is 63. The walker starts at position 0 with the position state $|0111111\rangle_p$. The initial coin state is $\frac{1}{\sqrt{2}}(|0\rangle_c + i|1\rangle_c)$. The circuit is then constructed by 12 qubits, including 7 qubits-position-state, 4 qubits-ancilla, and 1 qubit-coin-state. The circuit initialization can be seen in Figure 5.14, $q_0, q_1, q_2, q_3, q_4, q_5$ and q_6 represent the position states; $Coin$ represents the coin state; and $C_0, C_1, C_2 and\ C_3$ represent 4 the Ancilla qubits. Again, 4 modes of MCT function will replac the quantum gate in the red rectangular as illustrated in Figure 5.15.

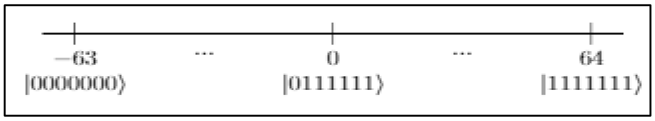


Figure 5.13 The number line of a one-dimension quantum random walk with seven qubits position state

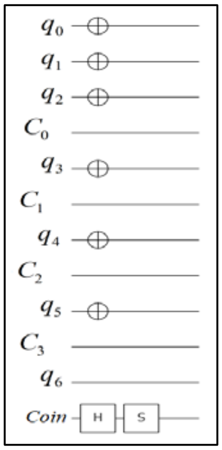


Figure 5.14 The initial set up of walking with a seven-position state and one coin-state

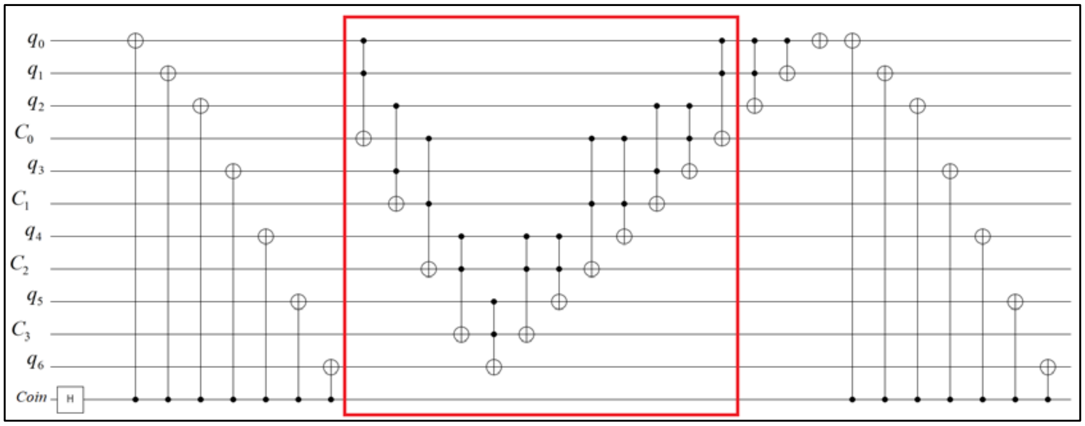


Figure 5.15 The coin and condition shift operator for walking on the number line with seven-position states. The vertical dash line separates the coin from the shift operator

Figure 5.16 displays the probability distributions of the measured results from the 100,000 walking processes performed on the Qiskit's *Qasm_simulator* and Rigetti's *QVM* using the quantum circuits with common gates and and Qiskit's with MCT functions. Like the previous 3 sizes, the probability distributions from the two quantum computer simulators and 4 MCT function modes are similar. Furthermore, the characteristic of the distribution is symmetrically spread from the starting point. Still, the computing wall-time between IBM and Rigetti quantum computer simulator is different.

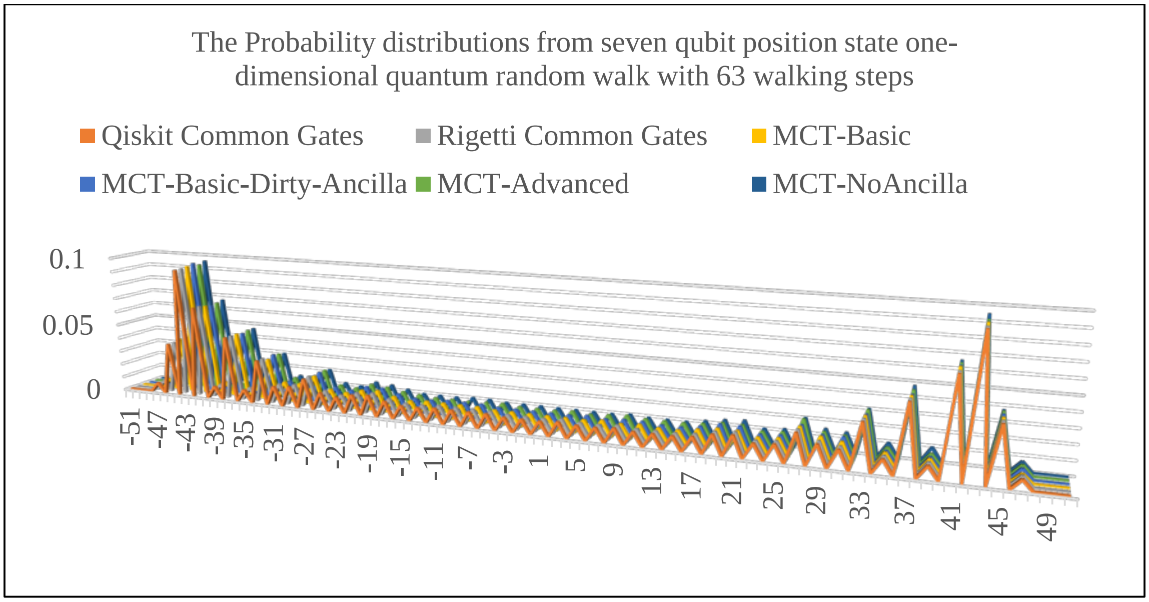


Figure 5.16 The probability distributions of seven-qubit position state one-dimensional quantum random walk with 63 walking steps

Table 5.4 The analysis of wall-time from running the quantum random walk on seven-qubit position state on the number line between IBM's *Qasm_simulator* and Rigetti's *QVM*

	Common quantum gates		MCT			
	Qasm_simulator	QVM	Basic	Basic-Dirty-Ancilla	Advanced	NoAncilla
Mean	**1.42**	**10908.271**	**3.268**	**12.65**	**19.171**	**30.537**
S.D.	0.029	36.558	0.038	0.194	0.244	0.25
Median	1.412	10913.062	3.261	12.605	19.155	30.535
Circuit Depth	1641	1641	10650	19848	18840	36669
Ancilla	4	4	4	4	2	0

5.1.5 Experimental Discussion

This section discusses the results from Section 5.1.1 - 5.1.4 based on 3 following points: the average probability distribution, the computing wall-time, and the relationship between the circuit depth and the computing wall-time.

5.1.5.1 Probability Distribution

Considering the position state with the same number of qubits, the probability distributions, as displayed in Figures 5.4, 5.8, 5.12, and 5.16, from different circuit implantations have similar trend. The probability distributions consistency proves that one-dimensional quantum random walk circuits, which are computed on Qiskit's *Qasm_simulator* and Rigetti's *QVM*, correspond to the theory in Section 3.2. However, peaks of the graph at the same position on the number line are slightly different, as shown in Figure 5.17. It might be assumed that the difference was occurred from a deviation from the average measurement values . The different trials of measurement are then conducted for a further analysis. Two additional experiments with 50 and 150-thousand shots are computed and measured. The results show that the difference among peaks still exists; the difference might due to the deviation.

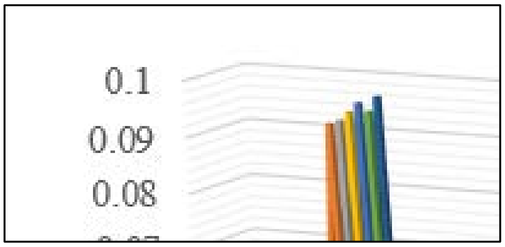


Figure 5.17 The peak of position -43 from seven-qubit position state walk

5.1.5.2 Computing wall-time (*Qasm_simulator* vs *QVM*)

For the identical circuits computed on IBM's *Qasm_simulator* and Rigetti's *QVM*, the computing wall-times are significantly distinct as illustrated in Figure 5.18. Note that the average computing wall-time from *Qasm_simulator* and *QVM* are shown, respectively, on the primary (left side) and secondary (right side) vertical axes.

Although the computing circuits are equivalent, the computing architectures of the two simulators are dissimilar. Both quantum computer simulators need to be analyzed. Unfortunately, both simulator are displayed with a binary file format, which is human unreadble file. Therefore, the detail of a circuit execution cannot be received by the source code analysis. We then try to study program documents from both simulators. However, the documents are not described in the detail of the circuit execution. Hence, resource utilization are inspected. Both simulators use 100% of single core CPU. For a memory usage, *QVM* takes 7.2% and *Qasm_simulator* takes 5.2% when computing the circuit. In the aspect of compiled programming language, moreover, the two simulators use different lanuguages. As described in Section 2.3, the python program for the *Qasm_simulator* is compiled into a Qasm instruction set. The Qasm program is constructed from C++ programming language to computes on the *Qasm_simulator*. Differently, the python program for the *QVM* is compiled into a native Quilc. The native Quil program is constructed from Lisp programming language to be computed on the *QVM*. As mentioned, both of the quantum computer simulators are computed with the default configuration. Then, with the difference of underlying programming language of the simulator, the computing wall-time from the identical circuit on different platforms can be different.

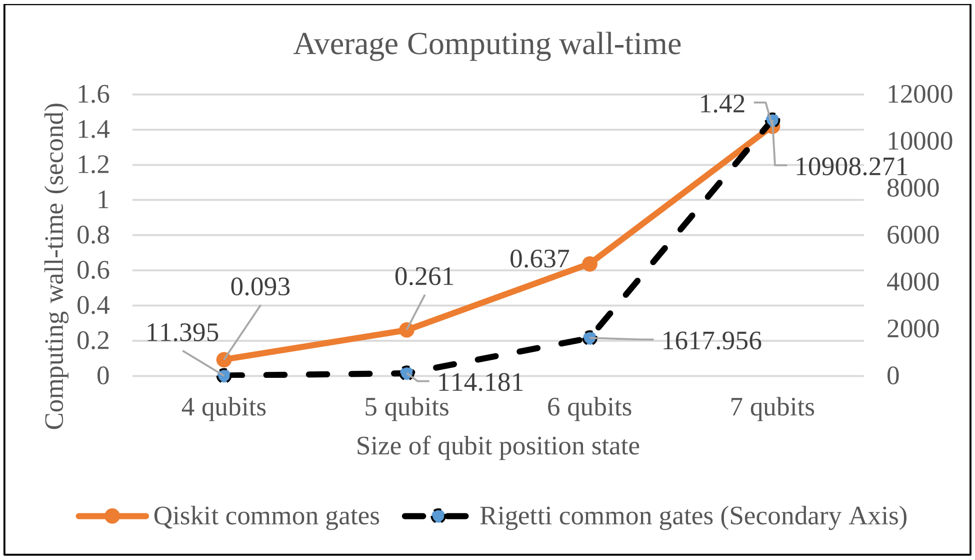


Figure 5.18 The computing wall-time from Qiskit and Rigetti quantum computer simulator

5.1.5.3 Circuit Types Vs Circuit Depth and Computing Wall-time

The circuit depths on each size of the one-dimensional quantum random walk are various as it depends on the depth of the conditional shift operator. In our experiments, 5 different types of conditional shift operators are used, including common quantum gates and 4 modes of MCT. However, the circuit depths of circuits that are made up of the common gates and run on the *Qasm_simulator* and *QVM* are equivalent because those circuits are identical. In this section, the discussion then focuses only on the results from *Qasm_simulator* because MCT is a special function of Qiskit.

Figures 5.19 – 5.22 show the computed circuit depth (on the primary vertical axis) and the computing wall-time (on the secondary vertical axis) of the 4-, 5-, 6-, and 7-qubit position state circuits, respectively. The circuit modes (on the horizontal axis) are sorted in an increasing order of the circuit depths, which are the same for all sizes of the position state, as: common gates (Toffoli-CNOT-NOT), MCT-Basic, MCT-Advanced, MCT-Basic-Dirty-Ancilla, and MCT-NoAncilla. It can be observed that the computing wall-time tends to increase when the circuit depth is growing. However, on the 5-, 6-, and 7-qubit position state in Figures 5.20-5.21, the pair of computing wall-time and circuit depth from MCT-Advanced and MCT-Basic-Dirty-Ancilla present a contradictory fashion. Even though the circuit depths of using MCT-Advance are smaller than those using the MCT-Basic-Dirty-Ancilla, the MCT-Advance's computing wall-times are longer than MCT-Basic-Dirty-Ancilla.

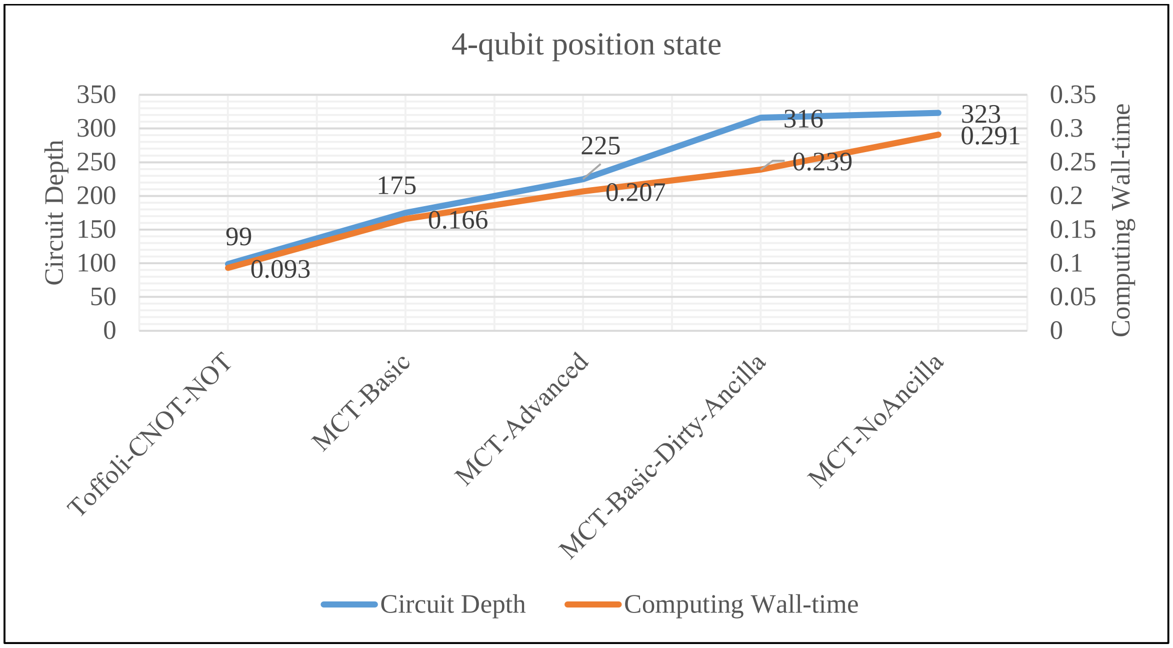


Figure 5.19 The circuit depth and computing wall-time of 4-qubit position state for each type of conditional shift operator

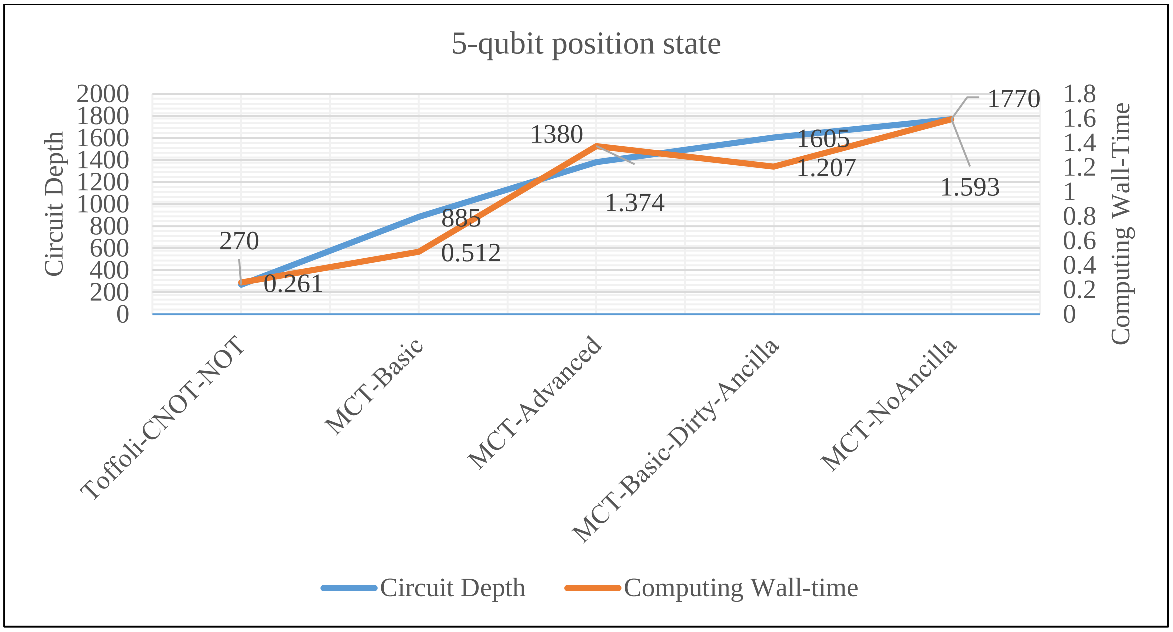


Figure 5.20 The circuit depth and computing wall-time of 5-qubit position state for each type of conditional shift operator

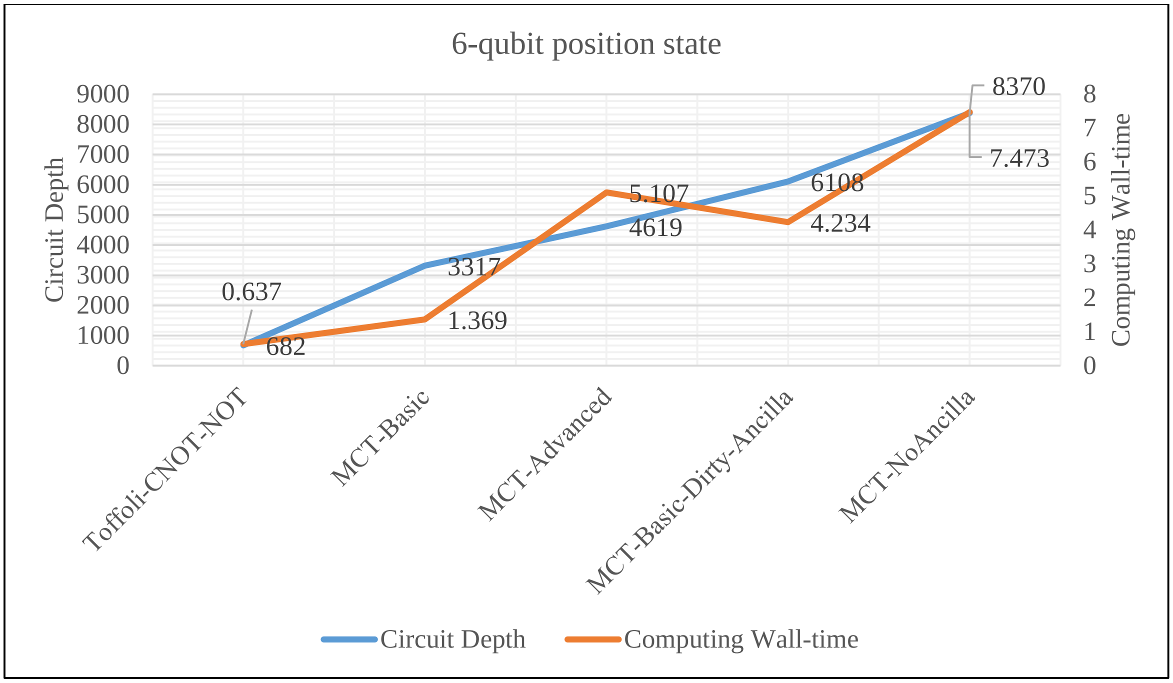


Figure 5.21 The circuit depth and computing wall-time of 6-qubit position state for each type of conditional shift operator

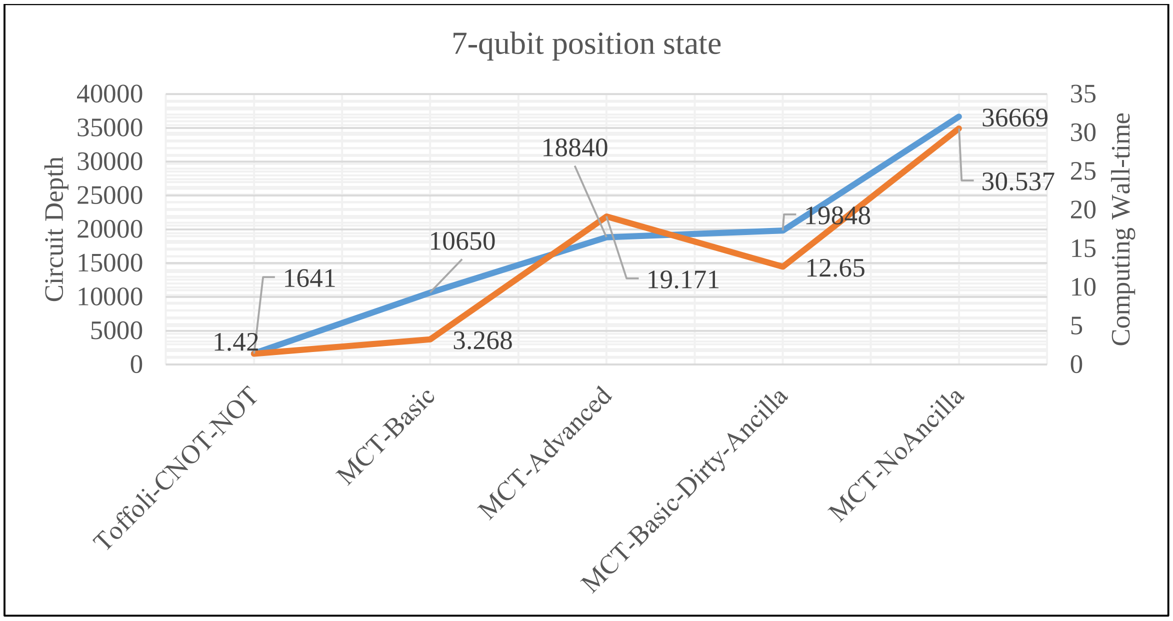


Figure 5.22 The circuit depth and computing wall-time of 7-qubit position state for each type of conditional shift operator

For further investigation, all the implemented circuits are transformed into a circuit that contains only a single-qubit (U_3) and two-qubits gate ($CNOT$). According to Kobayashi [27], the quantum circuit can be reassembled into a circuit that contains only those fundamental gates. The transformations are done using a function provided by IBM.

After the compilation and execution, the probability distribution from each circuit type is similar to its previous version. However, the computing wall-time seems not to depend on the circuit depth. For example, in the four-qubit position state as displayed in Figure 5.23, the MCT-Basic-Dirty-Ancilla has more depth than the common-quantum-gate circuit, but the MCT-Basic-Dirty-Ancilla circuit spends less time in computation. The inconsistency also occurs in comparing other types of the circuit with each size of qubit position state as shown in Figures 5.24 – 5.26. Note that the order of circuit modes on the horizontal axis is changed due to the circuit depth after unrolling the circuits.

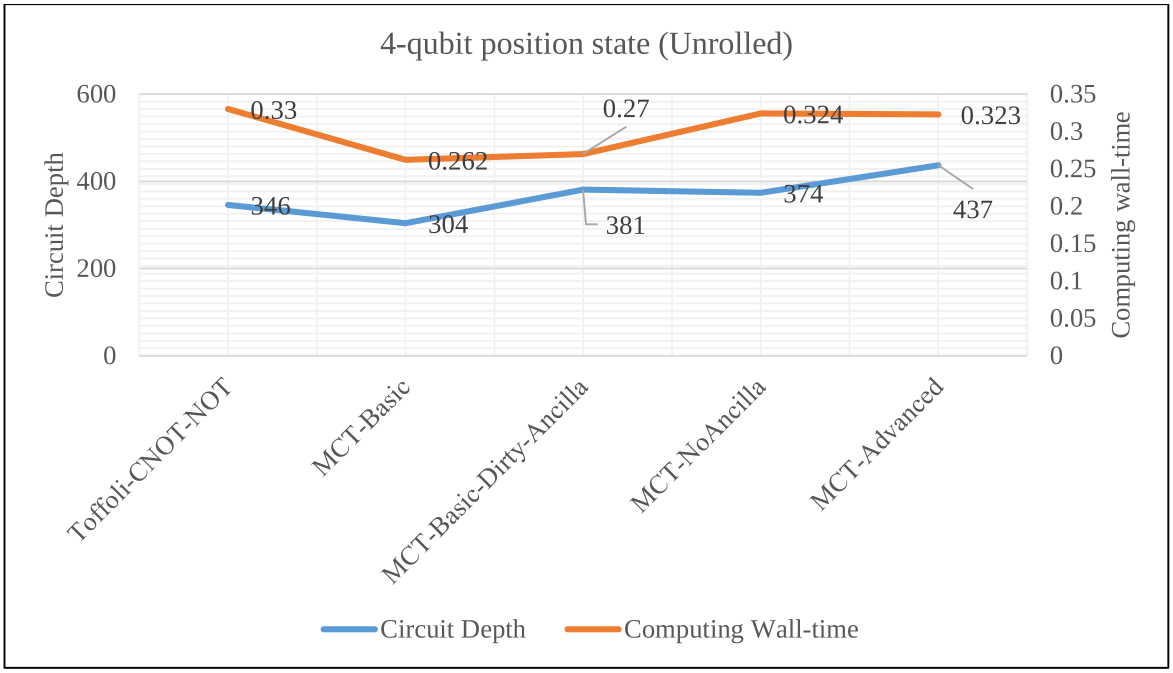


Figure 5.23 The circuit depth and computing wall-time of 4-qubit position state for each type of conditional shift operator after unrolling the circuit

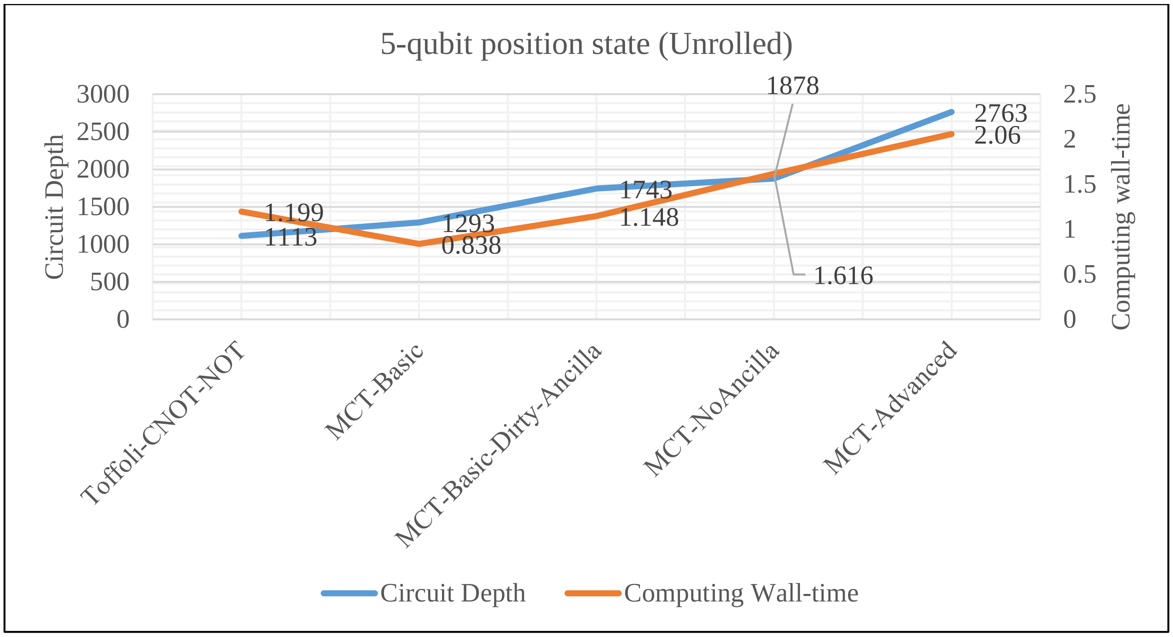


Figure 5.24 The circuit depth and computing wall-time of 5-qubit position state for each type of conditional shift operator after unrolling the circuit

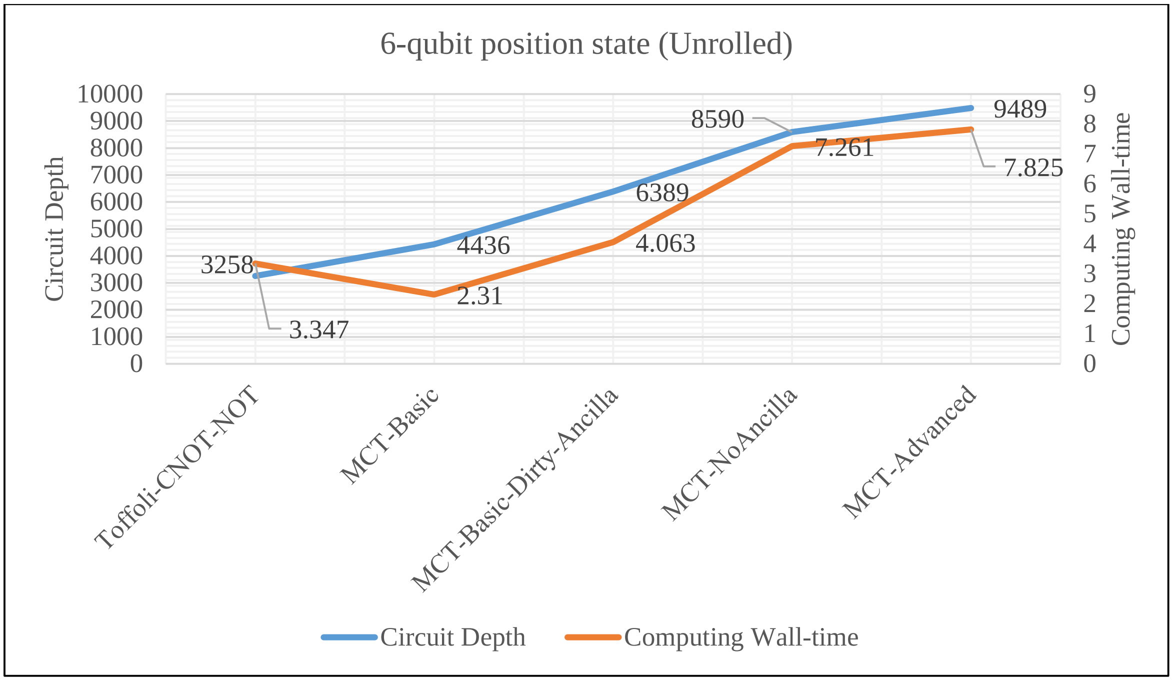


Figure 5.25 The circuit depth and computing wall-time of 6-qubit position state for each type of conditional shift operator after unrolling the circuit

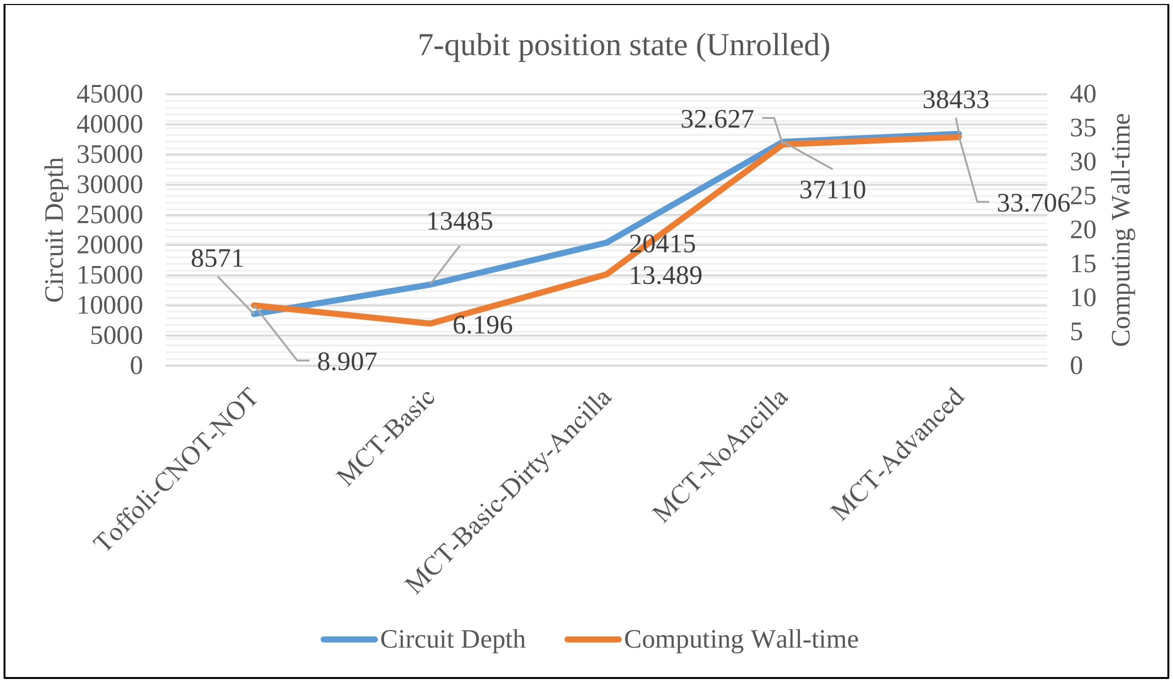


Figure 5.26 The circuit depth and computing wall-time of 7-qubit position state for each type of conditional shift operator after unrolling the circuit

5.1.5.4 Size of Qubit Position State Vs Circuit Depth and Computing Wall-time (on different circuit Types)

Instead of studying the relation between the circuit depth and the computing wall-time of the different circuit types, this section discusses another aspect of the that relation. The analysis of circuit depth and computing wall-time for the same type of conditional shift operator shows intriguing relation. For the same type of conditional shift operator, the computing wall-time increases as the circuit depth is increasing as demonstrated in Figures 5.27 – 5.31. A curve fitting for each graph is also presented.

From the graph fittings, both the computing wall-time and the circuit depth from the circuit that constructed from Toffoli-CNOT-NOT show a positive polynomial trend with the change in number of qubits position state. For the MCT-Advanced and MCT-NoAncilla, both the computing wall-time and the circuit depth present an exponential growth with the change in number of qubits position state. For the MCT-Basic and MCT-Basic-Dirty-Ancilla, the number of qubits position state has a positive polynomial trend with the computing wall-time, while having an increasing exponential trend with the circuit depth.

It can then be concluded that when the number of qubits position state increases, the circuit constructed from Toffoli-CNOT-NOT will have the lowest amount of both computing wall-time and circuit depth according to their polynomial trends.

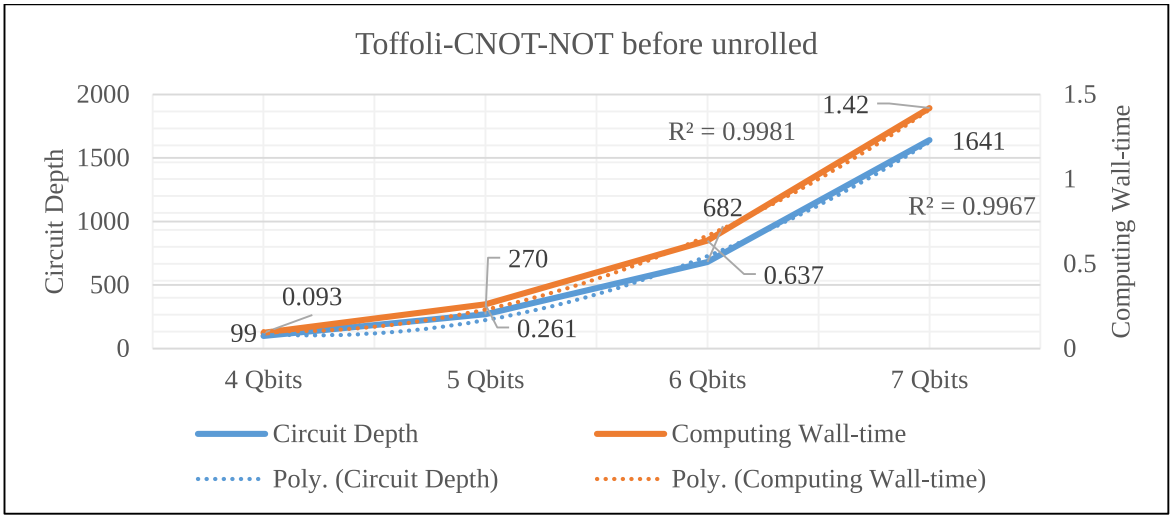


Figure 5.27 The circuit depth and computing wall-time of Toffili-CNOT-NOT conditional shift operator for each size of qubit position state

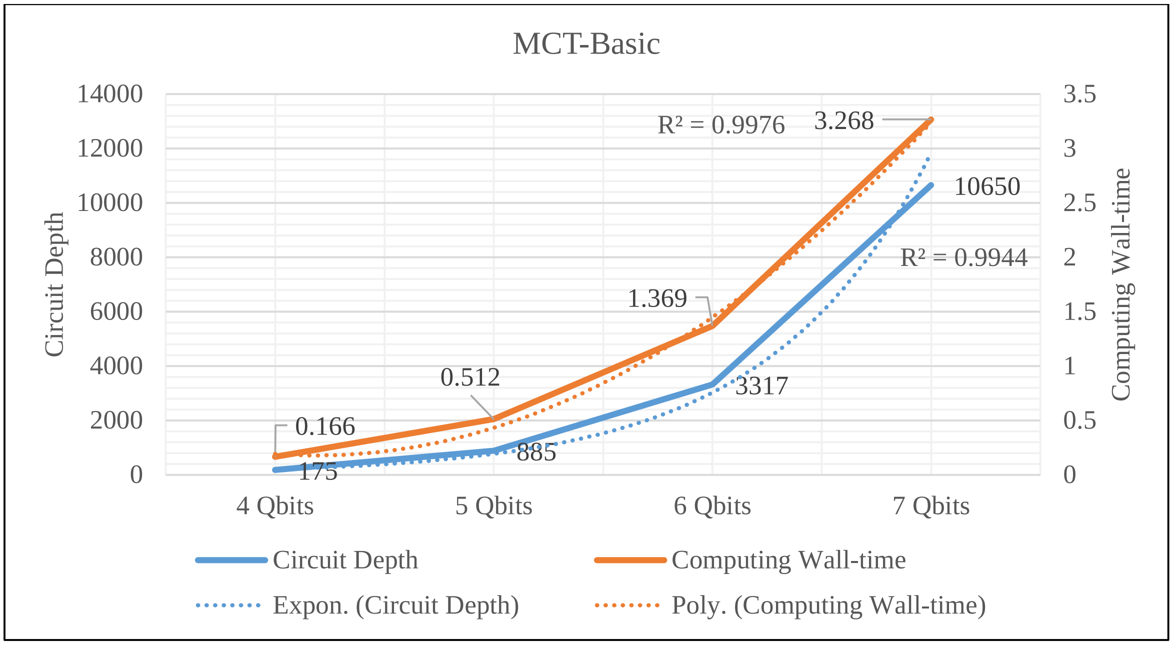


Figure 5.28 The circuit depth and computing wall-time of MCT-Basic conditional shift operator for each size of qubit position state

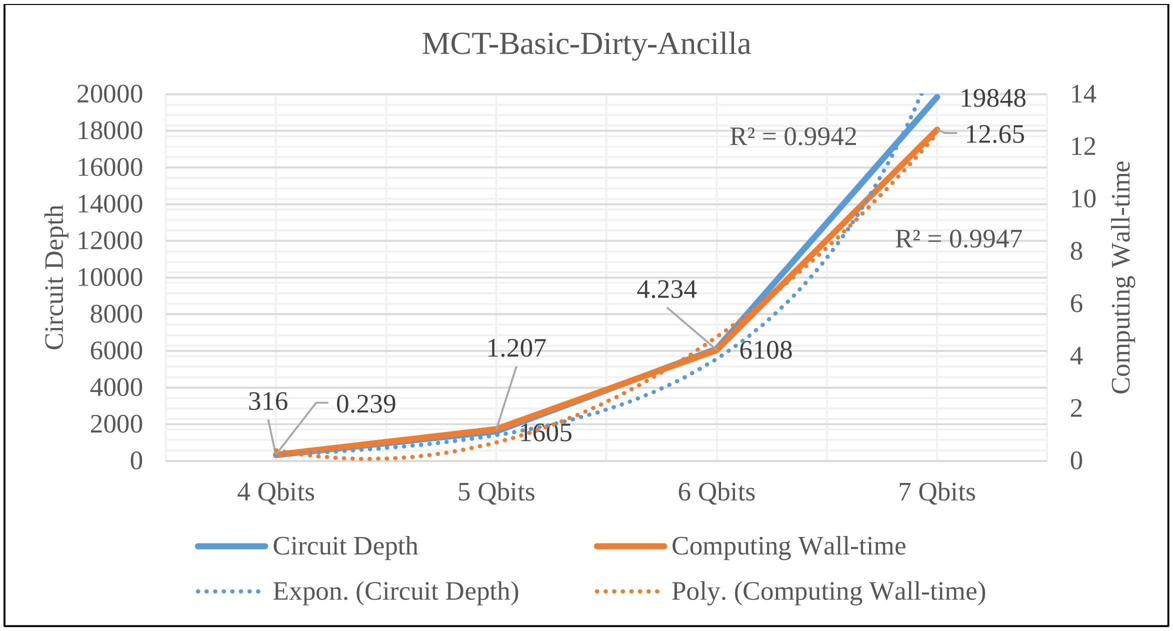


Figure 5.29 The circuit depth and computing wall-time of MCT-Basic-Dirty-Ancilla conditional shift operator for each size of qubit position state

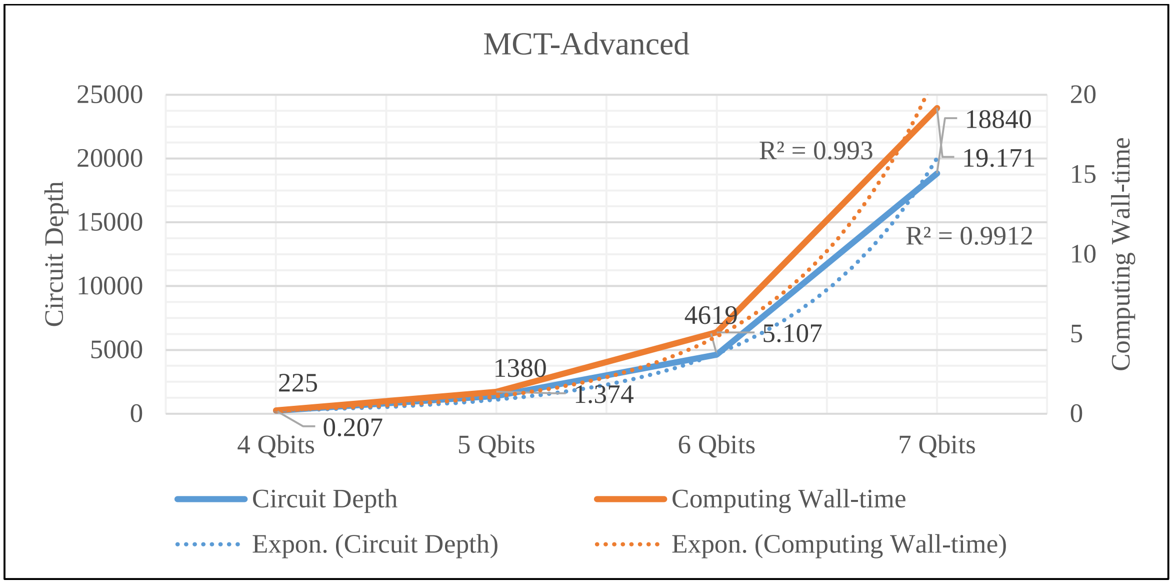


Figure 5.30 The circuit depth and computing wall-time of MCT-Advanced conditional shift operator for each size of qubit position state

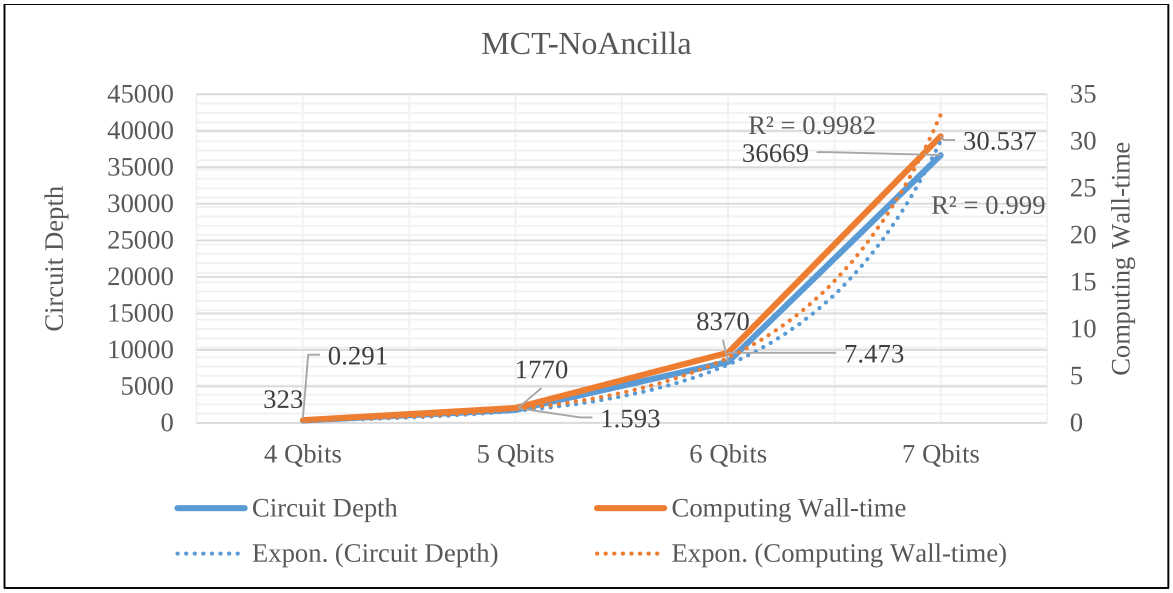


Figure 5.31 The circuit depth and computing wall-time of MCT-NoAncilla conditional shift operator for each size of qubit position state

5.1.5.5 The relation between computing wall-time and circuit depth

In this section, the relationships between circuit depth and computing wall-time are analyzed. As illustrated in Figure 5.32, it can be obviously seen that the computing wall-time and circuit depth have a linear relationship. The increasing rates, however, are different for each type of conditional shift operator. An increasing rate value presents the growth of the computing wall-time when the circuit depth is increasing, as shown in Table 5.5. For the same size of qubit position state, MCT-Advanced is the type that has more circuit depth and computing wall-time. However, the increasing rate is equal to the MCT-NoAncilla. The highest increasing rate among all types is the circuit constructed with Toffoli-CNOT-NOT. Nonetheless, for the same size of qubit position state, the circuit with Toffoli-CNOT-NOT, still takes less time than other type of conditional shift operator.

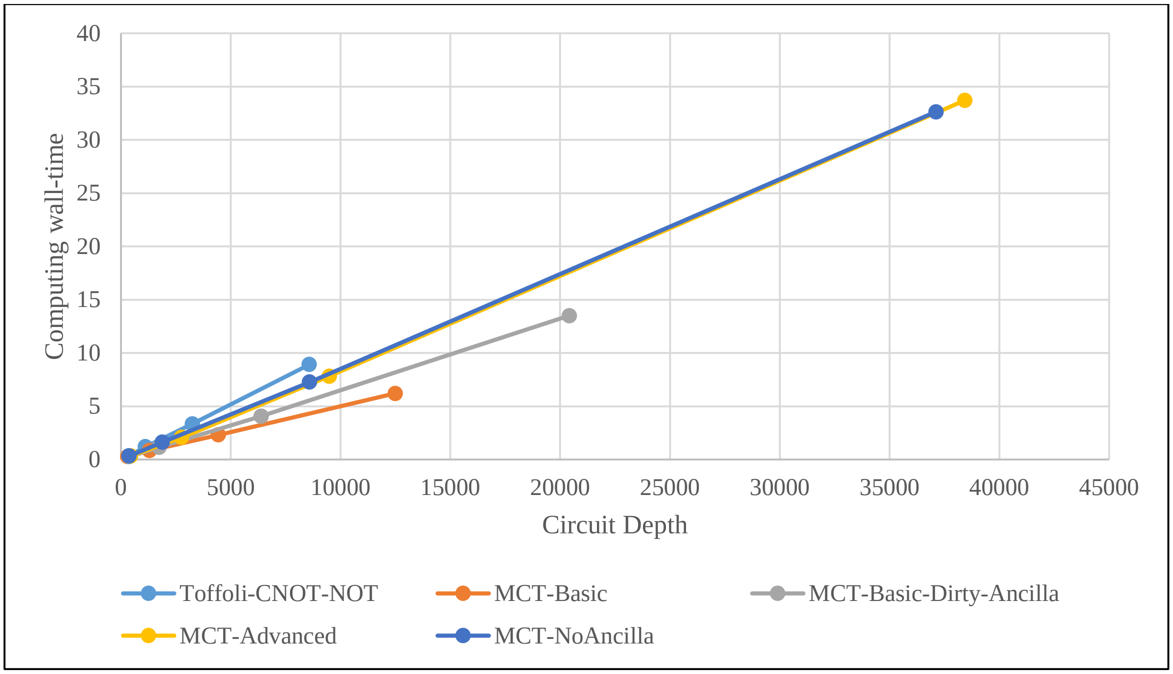


Figure 5.32 The conclusion of a relation between circuit depth and computing wall-time for each type of conditional shift operator

Table 5.5 The increasing rate of the computing wall-time from the circuit depth in Figure 5.32

Circuit types	Computing wall-time increasing rate from the Circuit depth
Toffoli-CNOT-NOT	0.001
MCT-Basic	0.0005
MCT-Basic-Dirty-Ancilla	0.0007
MCT-Advanced	0.0009
MCT-NoAncilla	0.0009

5.2 A Quantum Random Walk Computing on Quantum Computer

In this experiment, 4 different sizes of quantum random walk circuits, which are including 4, 5, 6, and 7 qubits position state, are computed on the real IBM's quantum computer, named *IBM_Q_16_Melbourne*. The objective of this study is to find a possibility to compute quantum circuits on the actual quantum computer.

The circuits in this experiment are identical to those in Section 5.1. The 5 different shift operator settings including common quantum gate, MCT-basic, MCT-basic-dirty-ancilla, MCT-advanced, and MCT-noancilla, are adopted. The coin operator is the Hadamard operator. The measurement operators are placed at the last operator on each qubit position state. The measurement shots are 8192, which is the maximum number of ther measurement shots from *IBM_Q_16_Melbourne.* The walking steps for the circuit with 4, 5, 6, and 7 qubits position states are 7, 15, 31, and 63, respectively.

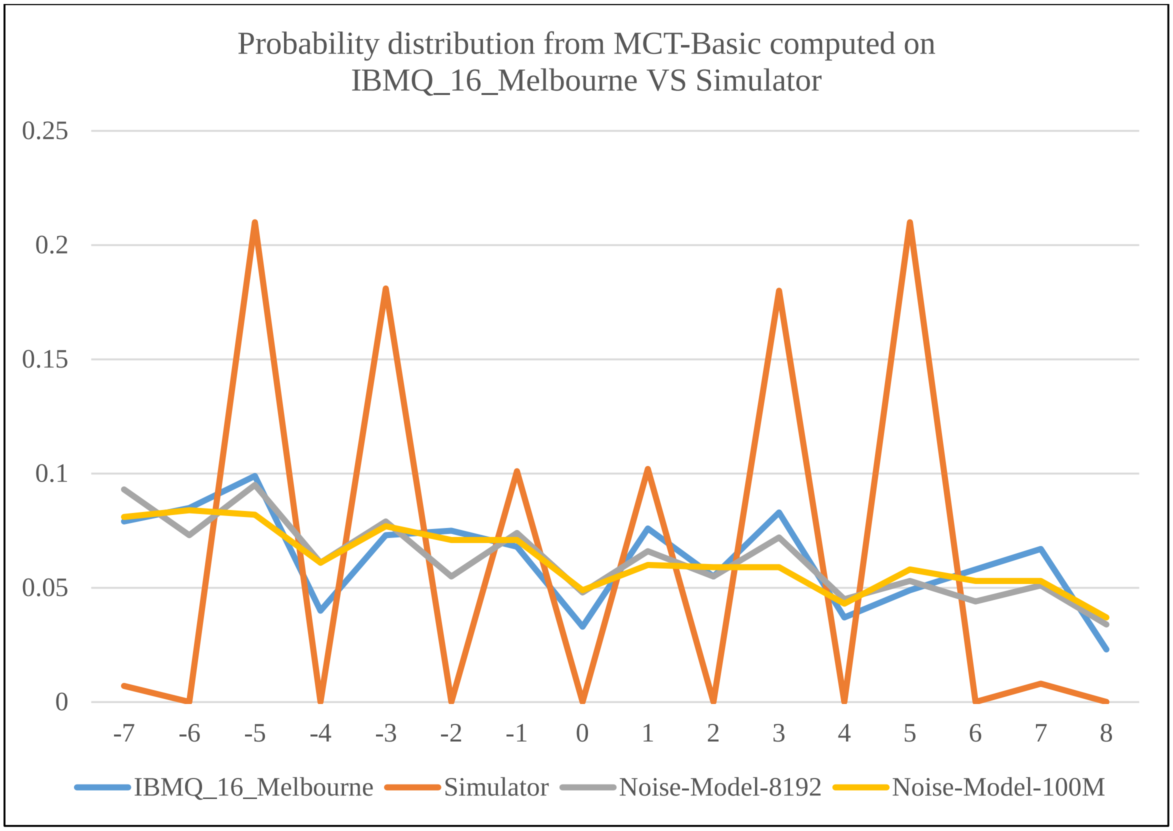


Figure 5.33 The probability distribution of four-qubits position state from each computation

To compute on *ibm_q_16_melbourne*, all circuits are submitted to the quantum computer by Qiskit's framework. The measurement results are then returned and the probability distribution from the measurement results was collected and displayed. As shown in Figure 5.33, *blue line* represents the probability distribution from *ibm_q_16_melbourne*, *red line* represents the probability distribution from quantum computer simulator without noise model, *grey line* represents the probability distribution from quantum computer simulator with noise model and 8192 measurement shots, *orange line* represents the probability distribution from quantum computer simulator with noise model and 100 million measurement shots. However, only the measurement results from the 4-qubits position state are returned from the computation, while the other circuit's size return the 8020 error code. According to Qiskit's document, the error 8020 is related to a longer circuit depth than the quantum computer's capability.

As illustrated in Figure 5.33, the returned probability distribution (the blue line) from *ibm_q_16_melbourne* is inconsistent with the results from the simulator (the red line). The inconsistency could come from the noise that occurred in the real quantum computer operation. To investigate further, the *ibm_q_16_melbourne*’s noise model was imported into *Qasm_simulator*. The circuits are computed again with 8,192 measurement shots. The probability distribution from the results (the grey line) is, however, still similar to the distribution from *ibm_q_16_melbourne*. Then, the measurement shots are increased to 100 million. The probability distribution is still in the same manner. Therefore, it might be concluded that noise is the variable that makes the probability distribution from the quantum computer different from the simulator.

CHAPTER 6 CONCLUSION

Quantum computing is one of the solutions that might break the limitation of computational power. Many quantum algorithms have been invented. Some of them have proved to be better than the classical algorithms when solving the same size and complexity of problems. In quantum computing, qubits are used to store information. A qubit can be in $|0\rangle$, $|1\rangle$, or the superposition state. Moreover, the multiple qubits can be joined together by a tensor product. The measurement operator can be applied to acquire the state of the qubit. After the measurement, the state of qubit will be projected to the measurement basis. To process qubit's information, a unitary transformation can be performed by employing quantum gates to qubits. For more complex computations, a collection of quantum gates can be used to construct a quantum circuit.

For quantum computing devices, several technologies can be used to construct a quantum processing unit. In this thesis, the computation are performed on the quantum devices using superconducting technology from IBM and Rigetti. These two companies have quantum processing units that are publicly available to researchers and also have complete documents. Moreover, the two companies provide an SDK to communicate between a programming language and the quantum processing unit. Qiskit and Forest are the quantum computing SDK provided by IBM and Rigetti respectively.

According to the literature, a quantum random walk is an analog to the classical random walk, which can be a tool to construct many other quantum algorithms. The shift operator is a significant part of the quantum random walk that operates to move the walker position. To construct a shift operator, the quantum circuit, which can operate as an incrementor and decrementer, is designed. The special function from IBM' Qiskit is explored as an alternative shift operator construction.

In this thesis, the one-dimensional quantum random walk algorithm is studied. The walking space, coin, and shift operators of the one-dimensional quantum random walk are analyzed. After that, the circuit of the one-dimensional quantum random walk is implemented. The shift operator is inspired by the incrementer circuit proposed by Li, et al. in 2013. Moreover, the different shift operators are constructed using Qiskit's MCT function with 4 distinct modes, namely Basic, Basic-Dirty-Ancilla, Advanced, and NoAncilla. Then, our shift operators are verified by setting up a quantum random walk calculation. The probability distributions obtained from testing the shift operators were consistent with one in the research paper by Kempe in 2003.

The benchmarking in IBM and Rigetti's quantum computer simulators are performed using the implemented quantum random walk circuit. The benchmark uses 4 sizes of the one-dimensional quantum random walk. For the equivalent position state size, the results show that they are not significantly different in the probability distributions measured from the two quantum computer simulators and four modes of the MCT function. For the computing wall-time, the results demonstrate that the computing wall-time depends on the circuit depth. However, even with the identical circuit depth, the computing wall-time on the different quantum computer simulators can be contrasted. From multiple trials, the results show that the computing wall-time of IBM's quantum computer simulator is significantly less than Rigetti's. Both quantum computer simulators are operated with the binary execution; therefore, the actual cause that accounts for the difference cannot be accurately defined.

The relationship between circuit depth and the number of qubits position state is observed on IBM's quantum computer simulator. The circuit depths are exponentially increasing except for the circuit constructed by Toffoli-CNOT-NOT that is rising in polynomial trend, when the number of qubits position state grows. Moreover, the relationship between computing wall-time and the number of qubit position state are investigated. The results show that the computing wall-times of Toffoli-CNOT-NOT, MCT-Basic and MCT-Basic-Dirty-Ancilla are increasing in a polynomial trend when the number of qubits position state is growing. Also, the circuit that has a larger depth spends a longer computing wall-time than a smaller one. Furthermore, the linear relationships between computing wall-time and circuit depth are examined. The circuit with MCT-Basic takes shorter computing wall-time when the circuit depth is increased. In contrast, the circuit with Toffoli-CNOT-NOT takes longer computing wall-time when the circuit depth is increased. However, for the same size of qubit position state, the circuit with Toffoli-CNOT-NOT, has the smallest depth and takes the least time than the other types of conditional shift operators.

Finally, another experiment is conducted to compare the circuit computed on *Qasm_simulator* and *ibmq_16_melbourne*. The acquired probability distributions are different, which could be the effect of the noise. The *ibmq_16_melbourne*'s noise model is imported to *Qasm_simulator* to confirmed that ibmq_16_melbourne is disturbed by the noise.

REFERENCES

1. Moore, G.E., 1965, "Cramming More Components onto Integrated Circuits", **Electronics**, Vol. 38, Issue 8.

2. Hassan, S., Humaira and Asghar, M., 2010, "Limitation of Silicon Based Computation and Future Prospects", **Proceedings of 2nd ICCSN International Conference on Communication Software and Networks**, Singapore, pp. 559-561.

3. Quantum Computing Report, 2019, **Qubit Technology**, [Online], Available:https://quantu mcomputingreport.com/scorecards/qubit-technology/ [2019, April 18].

4. Korotkov, A., 2009, "Special Issue on Quantum Computing with Superconducting Qubits", **Quantum Information Processing - QUANTUM INF PROCESS**, Vol. 8, Issue, pp. 51-54.

5. Jordan, S., 2020, **Quantum Algorithm Zoo**, [Online], Available: https://quantumalgorith mzoo.org/ [2020, April 22].

6. Raj, G., Singh, D. and Madaan, A., 2018, "Analysis of Classical and Quantum Computing Based on Grover and Shor Algorithm", **Smart Computing and Informatics**, pp. 413-423.

7. Ambainis, A., 2008, "Quantum Random Walks – New Method for Designing Quantum Algorithms", **SOFSEM 2008: Theory and Practice of Computer Science**, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1-4.

8. Ambainis, A., 2007, "Quantum Walk Algorithm for Element Distinctness", **SIAM Journal on Computing**, Vol. 37, Issue 1, pp. 210-239.

9. Roy, S.G. and Chakrabarti, A., 2017, "A Novel Graph Clustering Algorithm Based on Discrete-Time Quantum Random Walk", **Quantum Inspired Computational Intelligence**, pp. 361-389.

10. Mülken, O., Pernice, V. and Blumen, A., 2007, "Quantum Transport on Small-World Networks: A Continuous-Time Quantum Walk Approach", **Physical Review E**, Vol. 76, Issue 5, pp. 051125.

11. Venegas-Andraca, S., 2008, **Quantum Walks for Computer Scientists**, Morgan & Claypool Publishers, pp. 61-77.

12. O'Brien, J.L., 2007, "Optical Quantum Computing", **Science**,Vol. 318, pp. 1567-1570.

13. Cirac, J.I. and Zoller, P., 1995, "Quantum Computations with Cold Trapped Ions", **Physical Review Letters**.Vol. 74, Issue 20, pp. 4091-4094.

14. Kielpinski, D., Monroe, C. and Wineland, D.J., 2002, "Architecture for a Large-Scale Ion-Trap Quantum Computer", **Nature**, Vol. 417, Issue 6890, pp. 709-711.

15. Häffner, H., Roos, C.F. and Blatt, R., 2008, "Quantum Computing with Trapped Ions", **Physics Reports**, Vol. 469, Issue 4, pp. 155-203.

16. IBM, 2019, **List of Ibm Quantum Devices,** [Online], Available: https://quantumexperience.ng.bluemix.net/qx/devices [2019, March 18].

17. Cross, A., Bishop, L., Smolin, J. and Gambetta J., 2020, **Open Quantum Assembly Language**, [Online], Available: https://arxiv.org/abs/1707.03429 [2020, May 18].

18. IBM, 2019, **Backend Information of Ibm Quantum Devices,** [Online], Available: https://github.com/Qiskit/ibmq-device-information [2019, March 18].

19. IBM, 2019, **16-Qubit Backend: Ibm Q Team, "Ibm Q Melbourne Backend Specification V1.1.0,",** [Online], Available: https://github.com/Qiskit/ibmq-device-information/tree/master/backends/melbourne/V1 [2019, March 18].

20. Rigetti, 2019, **Qpu Specificatoins,** [Online], Available: https://www.rigetti.com/qpu [2019, March 18].

21. Smith, R.S., Curtis, M.J. and Zeng, W.J., 2016, **A Practical Quantum Instruction Set Architecture**, [Online], Available: https://arxiv.org/abs/1608.03355 [2020, May 18]

22. LaRose, R., 2018, "Overview and Comparison of Gate Level Quantum Software Platforms", **Quantum**,Vol. 3, p. 130.

23. Aharonov, Y., Davidovich, L. and Zagury, N., 1993, "Quantum Random Walks", **Physical Review**, Vol. 48, pp. 1687-1690.

24. Kempe, J., 2003, "Quantum Random Walks: An Introductory Overview", **Contemporary Physics**,Vol. 44, pp. 307-327.

25. Lovett, N.B., Cooper, S.,Everitt, M., Trevers, M. and Kendon, V, 2010, "Universal Quantum Computation Using the Discrete-Time Quantum Walk", **Physical Review A**, Vol. 81, Issue 4, p. 042330.

26. Xiaoyu, Li, Guowu, Y., Carlos, M. T. JR., Desheng, Z. and Kang, L. W., 2014, "A Class of Efficient Quantum Incrementer Gates for Quantum Circuit Synthesis", **International Journal of Modern Physics B**, Vol. 28, p 1350191.

27. Kobayashi, Y., 2019, **How to Calculate Quantum Costs Using an Unroller,** [Online], Available:https://github.com/quantum-challenge/2019/blob/master/ /problems/ week1/week1_en.ipynb [2020, May 18].

CURRICULUM VITAE

NAME	Mr. Warat Puengtambol
DATE OF BIRTH	22 April 1989

EDUCATIONAL RECORD

HIGH SCHOOL	High School Graduation Potisarnpittayakorn School, 2006
BACHELOR'S DEGREE	Bachelor of Engineering (Computer Engineering) King Mongkut's University of Technology Thonburi, 2010
MASTER'S DEGREE	Master of Engineering (Computer Engineering) King Mongkut's University of Technology Thonburi, 2020

EMPLOYMENT RECORD

- Programmer
  The Digital STM Limited Partnership 2021 – present

- Computerist
  King Mongkut's University of Technology Thonburi 2016 – 2020

- Network and System Administrator
  Department of Computer Engineering,
  Faculty of Engineering
  King Mongkut's University of Technology Thongburi 2014 – 2016

PUBLICATION	Puengtambol, W., Prechaprapranwong, P. and Taetragool, U., 2020, “Implementation of Quantum Random Walk on a Real Quantum Computer”, The 15th Siam Physics Congress SPC2020, 4-5 June 2020, Thailand.