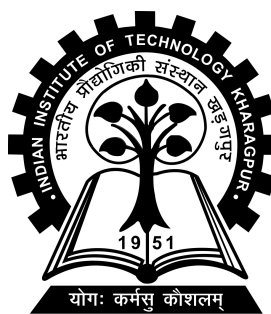


Simulating quantum circuits using the Qiskit library

Project-I (CS47005) report submitted to
Indian Institute of Technology Kharagpur
in partial fulfilment for the award of the degree of
Bachelor of Technology
in
Computer Science and Engineering

by
Hardik Pravin Soni
(20CS30023)

Under the supervision of
Professor Indranil Sen Gupta



Department of Computer Science and Engineering

Indian Institute of Technology Kharagpur

Autumn Semester, 2023-24

November 7, 2023

DECLARATION

I certify that

- (a) The work contained in this report has been done by me under the guidance of my supervisor.
- (b) The work has not been submitted to any other Institute for any degree or diploma.
- (c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.
- (d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

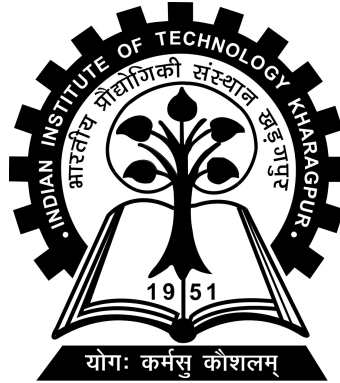
Date: November 7, 2023

Place: Kharagpur

(Hardik Pravin Soni)

(20CS30023)

DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
KHARAGPUR - 721302, INDIA



CERTIFICATE

This is to certify that the project report entitled “Simulating quantum circuits using the Qiskit library” submitted by Hardik Pravin Soni (Roll No. 20CS30023) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2023-24.

Indranil Sen Gupta

Professor Indranil Sen Gupta

Department of Computer Science and

Engineering

Indian Institute of Technology Kharagpur

Kharagpur - 721302, India

Date: November 7, 2023

Place: Kharagpur

Abstract

Name of the student: **Hardik Pravin Soni**

Roll No: **20CS30023**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Computer Science and Engineering**

Thesis title: **Simulating quantum circuits using the Qiskit library**

Thesis supervisor: **Professor Indranil Sen Gupta**

Month and year of thesis submission: **November 7, 2023**

Simulating quantum circuits using the Qiskit library

Simulating quantum circuits is an essential tool for developing and testing quantum algorithms and applications. Qiskit is a popular open-source library for quantum computing that provides a variety of tools for simulating quantum circuits, including:

- A quantum circuit simulator that can simulate circuits up to millions of qubits.
- A noise model simulator that can simulate the effects of noise on quantum circuits.
- A statevector simulator that can simulate the state of a quantum system after applying a circuit.
- A density matrix simulator that can simulate the evolution of a quantum system after applying a circuit.

Qiskit's simulation tools are easy to use and provide a variety of features, such as the ability to:

- Simulate circuits with parameterized gates.
- Simulate circuits with classical control.
- Measure the expectation values of observables.
- Visualize the state of a quantum system.

Qiskit's simulation tools are used by researchers and practitioners around the world to develop and test quantum algorithms and applications. For example, Qiskit has been used to simulate quantum circuits for quantum machine learning, quantum chemistry, and quantum cryptography.

In this abstract, we provide a brief overview of how to simulate quantum circuits using the Qiskit library. We also discuss some of the key features of Qiskit's simulation tools and provide examples of how they can be used to simulate different types of quantum circuits.

Keywords: Qiskit, quantum simulation, quantum circuits, quantum algorithms, quantum machine learning, quantum chemistry, quantum cryptography

Acknowledgements

I express my sincere gratitude to my supervisor, Professor Indranil Sengupta, for granting me the opportunity to work under his mentorship. Throughout my B.Tech project, he provided insightful guidance and constructive feedback. He was consistently supportive and understanding.

I am grateful for the chance to have worked on this project and to have learned from the people I have met along the way. This project has been challenging but rewarding, and I am proud of what I have achieved. I would also like to extend my warmest gratitude to my family and friends at IIT Kharagpur for their unwavering support and motivation throughout my journey.

Contents

Declaration	i
Certificate	ii
Abstract	iii
Acknowledgements	v
Contents	vi
List of Figures	viii
List of Tables	ix
Contents	x
Abbreviations	xi
Symbols	xii
1 Introduction	1
1.1 Introduction	1
1.2 Qiskit Basics	2
1.2.1 Qiskit	2
What is a quantum circuit ?	2
How to build a quantum circuit in Qiskit ?	3
How to simulate a quantum circuit in Qiskit ?	3
1.2.2 Applications of Quantum Circuit Simulation	3
1.2.2.1 Testing and developing quantum algorithms and circuits	4
1.2.2.2 Studying quantum phenomena	4
1.2.2.3 Designing quantum computers	4
1.3 A Note on Quantum Gates	5

2	Methodology	11
2.1	Qubit	11
2.2	Superposition State	12
2.3	Quantum Circuit and Measurement	15
2.4	Quantum Gate and Reversibility	16
	No-Cloning Theorem and Entangled State	19
2.5	Quantum Circuits and Algorithms Developed	20
2.5.1	Increment	20
2.5.2	Decrement	21
2.5.3	Build BELL State	22
2.5.4	Build GHZ State	23
2.5.5	Quantum Fourier Transform	24
	Results and Observations	26
2.6	Increment	26
2.7	Decrement	28
2.8	Build BELL State	30
2.9	Build GHZ State	32
2.10	Quantum Fourier Transform	34
	Summary	37
	Future Work	38

List of Figures

1.1	Quantum Circuit Development Process	4
1.2	Hadamard Gate (H)	5
1.3	Pauli X gate (X)	6
1.4	Pauli Y gate (Y)	6
1.5	Pauli Z gate (Z)	8
1.6	Swap gate gate (SWAP)	9
1.7	Toffoli gate	10
2.1	All Photons are detected in DV	13
2.2	The half of photons are found at DV, and the rest are at DH	13
2.3	n-qubit quantum circuit	15
2.4	Controlled-Not gate behaves like a classical XOR	16
2.5	Controlled-Controlled-NOT (Toffoli gate)	18
2.6	Quantum Circuit Diagram of Increment	21
2.7	Quantum Circuit Diagram of Decrement	22
2.8	Quantum Circuit Diagram of Build BELL State	23
2.9	Quantum Circuit Diagram of Build GHZ State	24
2.10	Quantum Circuit Diagram of Quantum Fourier Transform	25
2.11	Probabilities for High Result VS. For Quantum State 001	27
2.12	Probabilities for Low Result VS. For Quantum State 111	28
2.13	Probability to into Respective Quantum State after Operation for Each Quantum State of 00, 01, 10 and 11	30
2.14	Plot State qsphere for Bell State 10 qsphere	32
2.15	Plot State qsphere for Bell State 11 qsphere	32
2.16	Probabilities for GHZ State	34
2.17	Histogram Plot for QFT in 5 - QB-its	35
2.18	Plot State qsphere for Computational basis state qsphere	36
2.19	Plot State qsphere for Fourier basis state qsphere	36

List of Tables

2.1	A truth table for the quantum XOR gate with two inputs and two outputs	16
2.2	A truth table for the classical NAND gate with two inputs	17
2.3	Truth table for Quantum NAND gate	18
2.4	Truth table for Controlled-Controlled-NOT	18
2.5	Truth table for Quantum Increment Operation	21
2.6	Truth table for Quantum Decrement Operation	22

Listings

1.1	The <code>h()</code> method applies a Hadamard gate to qubit 0	3
1.2	The <code>run()</code> method executes the quantum circuit on the simulator and returns a <code>Result</code> object.	3
1.3	Working of a SWAP Gate	9

Abbreviations

QC	Quantum Circuit
QB	Quantum Bits

Symbols

ψ	Quantum Variable
$ \rangle$	Quantum Bit
H	Hadamard Gate
X	Pauli X Gate
Y	Pauli Y Gate
Z	Pauli Z Gate
$SWAP$	Swap Quantum Gate
$CNOT$	Controlled-NOT Gate
\mathbb{C}	Complex Number
\oplus	Tensor Product

Chapter 1

Introduction

1.1 Introduction

Quantum computers could revolutionize many fields, including science, engineering, and medicine. However, they are still in development because of the complexity of quantum mechanics. Quantum computers are difficult to build and operate because their **Qubits**, the basic units of information, are extremely fragile and susceptible to errors. Simulation is a useful tool for developing quantum algorithms and circuits because it allows researchers to test and refine their ideas before they are implemented on real quantum hardware. Simulations can help to identify and overcome challenges such as Qubit decoherence, noise, and scalability.

Elaboration of the description:

- **Qubit decoherence:** Qubits are extremely sensitive to their environment, and even small disturbances can cause them to lose their quantum properties, a phenomenon known as decoherence. This is a major challenge for quantum computing, as it can lead to errors in calculations. Simulations can help researchers to understand and mitigate decoherence.

- **Noise:** Quantum computers are also susceptible to noise from their environment. This can cause errors in calculations, making it difficult to obtain reliable results. Simulations can help researchers to identify and reduce noise sources.
- **Scalability:** Building quantum computers with a large number of qubits is a major challenge. Simulations can help researchers to design and test scalable quantum algorithms and circuits. Overall, simulation is a valuable tool for developing quantum computers and algorithms. It allows researchers to test and refine their ideas before they are implemented on real quantum hardware, which can save time and resources.

1.2 Qiskit Basics

1.2.1 Qiskit

is an open-source Software Development kit (SDK) for quantum computing. It provides a comprehensive set of tools for building, simulating, and running quantum circuits. Qiskit also provides access to a variety of quantum hardware platforms, including cloud-based quantum computers and simulators. Here I will Introduce the Basic of Quantum Circuit Simulation via Qiskit. We will discuss the following topics:

What is a quantum circuit ? A quantum circuit is a graphical representation of a quantum algorithm. It is made up of a series of quantum gates, which represent operations that can be performed on quantum bits (qubits). Qubits are the basic unit of information in quantum computing, and they may be in a superposition of states.

How to build a quantum circuit in Qiskit ? To build a quantum circuit in Qiskit, you can use the `QuantumCircuit` class (QC Class). This class provides a variety of methods for adding quantum gates and other elements to the circuit. For Example:-

```
1 from qiskit import QuantumCircuit
2 qc = QuantumCircuit(1)
3 qc.h(0)
```

LISTING 1.1: The `h()` method applies a Hadamard gate to qubit 0

How to simulate a quantum circuit in Qiskit ? To simulate a quantum circuit in Qiskit, you can use the `AerSimulator` class. This class provides a variety of simulation backends, which can be used to simulate quantum circuits of different sizes and complexity. For example, the following code shows how to simulate the quantum circuit from the previous example using the `AerSimulator` class:

```
1 from qiskit import QuantumCircuit
2 qc = QuantumCircuit(1)
3 qc.h(0)
4 simulator = AerSimulator()
5 result = simulator.run(qc).result()
```

LISTING 1.2: The `run()` method executes the quantum circuit on the simulator and returns a `Result` object.

The `Result` object contains the results of the simulation, such as the state of the qubits at the end of the circuit.

1.2.2 Applications of Quantum Circuit Simulation

Quantum circuit simulation can be used for a variety of purposes, including:

1.2.2.1 Testing and developing quantum algorithms and circuits

Quantum circuit simulation can be used to test and debug quantum algorithms and circuits before they are implemented on real quantum hardware. This can help to identify and fix errors in the algorithms and circuits, and it can also help to optimize the algorithms and circuits for performance.

1.2.2.2 Studying quantum phenomena

Quantum circuit simulation can be used to study quantum phenomena that are difficult or impossible to study experimentally. For example, quantum circuit simulation can be used to study the behavior of quantum entanglement and decoherence.

1.2.2.3 Designing quantum computers

Quantum circuit simulation can be used to design new quantum computer architectures and to evaluate the performance of different quantum computer designs.

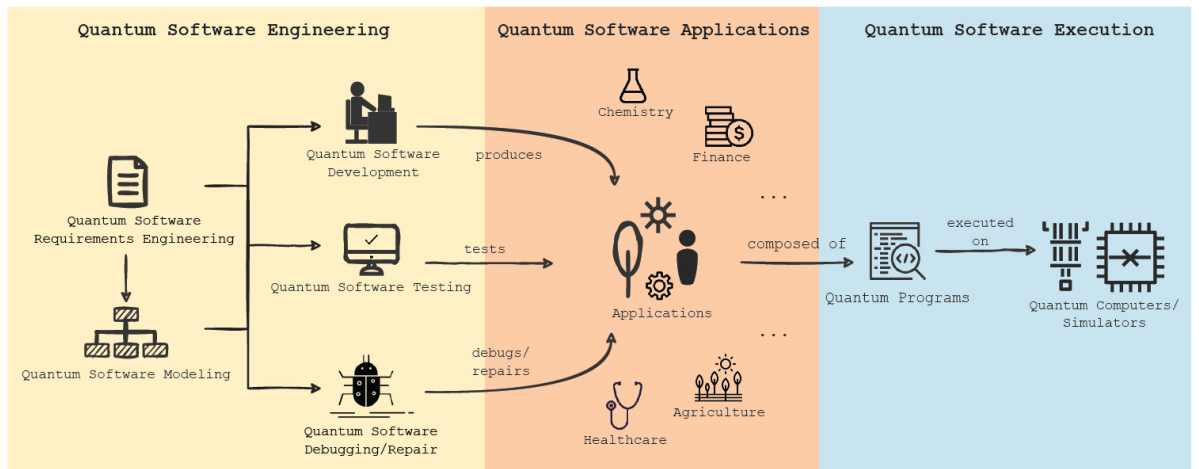


FIGURE 1.1: Quantum Circuit Development Process

1.3 A Note on Quantum Gates

Here are some of the most important quantum gates and a brief description of each:

- **Hadamard gate (H):** The Hadamard gate puts a qubit in an equal superposition of the $|0\rangle$ and $|1\rangle$ states. The Hadamard gate is used in many quantum algorithms, such as Shor's algorithm for factoring integers, Grover's algorithm for searching unsorted databases, and the Deutsch-Jozsa algorithm. It is also used in quantum teleportation and quantum entanglement swapping. Here is a brief explanation of how the Hadamard gate works:

- The Hadamard gate takes a qubit in the $|0\rangle$ state and puts it in a superposition of the $|0\rangle$ and $|1\rangle$ states with equal probability.
- The Hadamard gate takes a qubit in the $|1\rangle$ state and puts it in a superposition of the $|0\rangle$ and $|1\rangle$ states with opposite probability.

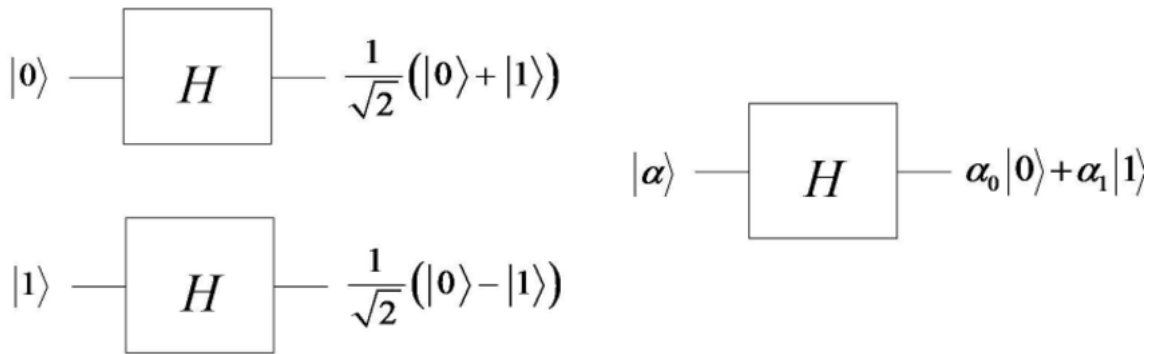
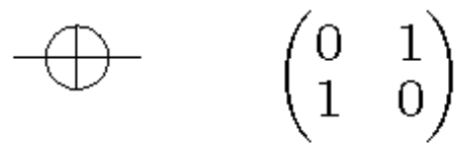


FIGURE 1.2: Hadamard Gate (H)

- **Pauli X gate (X):** The Pauli X gate flips the state of a qubit. To apply the Pauli X gate to a qubit, we simply multiply the state of the qubit by the matrix. The Pauli X gate is a universal gate, which means that it can be

Pauli-X gate



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

FIGURE 1.3: Pauli X gate (X)


used to implement any other quantum gate. For example, the following circuit shows how to implement the Hadamard gate using the Pauli X gate:

$$H = X * (\sqrt{Z}) * X$$

The Pauli X gate is also a key ingredient in many quantum algorithms, such as the Deutsch-Jozsa algorithm and the Grover's algorithm. The Pauli X gate is an essential tool for quantum computing. It is used in many quantum gates and algorithms, and it is a key ingredient for building quantum computers.

- **Pauli Y gate (Y):** The Pauli Y gate rotates the state of a qubit by 90 degrees around the Y-axis of the Bloch sphere. The Pauli Y gate is one of the three Pauli gates, which are the most important single-qubit quantum gates. The other two Pauli gates are the Pauli X gate and the Pauli Z gate. The Pauli

Pauli-Y gate



$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

FIGURE 1.4: Pauli Y gate (Y)

Y gate is an essential part of many quantum algorithms, and it is one of the most important single-qubit quantum gates. For Example: Rotating the state of a qubit around the Y-axis of the Bloch sphere, Implementing the quantum Fourier transform, Implementing the Deutsch- Jozsa algorithm, Implementing the Shor's algorithm etc. Here is an example of how the Pauli Y gate can be used to rotate the state of a qubit: Suppose we have a qubit in the state $|0\rangle$. We can apply the Pauli Y gate to the qubit to rotate its state to the $|1\rangle$ state. The following equation shows how the Pauli Y gate will rotate the state of the qubit:

$$|0\rangle \xrightarrow{Y} Y|0\rangle = |1\rangle$$

We can also apply the Pauli Y gate to a qubit in the state $|1\rangle$ to rotate its state to the $|0\rangle$ state. The following equation shows how the Pauli Y gate will rotate the state of the qubit:

$$|1\rangle \xrightarrow{Y} Y|1\rangle = |0\rangle$$

- **Pauli Z gate (Z):** The Pauli Z gate flips the phase of a qubit. the Bloch sphere. The Pauli Z gate is an essential part of many quantum algorithms,

Pauli-Z gate

$$\text{---} \boxed{\text{Z}} \text{---} \quad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

FIGURE 1.5: Pauli Z gate (Z)

and it is one of the most important single-qubit quantum gates. The Pauli Z gate can be used to implement a variety of quantum operations, such as: Flipping the phase of a qubit, Measuring a qubit in the Z-basis, Implementing the quantum controlled-Z gate, Implementing the quantum controlled-NOT gate etc. Here is an example of how the Pauli Z gate can be used to flip the phase of a qubit: Suppose we have a qubit in the state $|0\rangle$. We can apply the Pauli Z gate to the qubit to flip its phase. The following equation shows how the Pauli Z gate will flip the phase of the qubit:

$$|0\rangle \xrightarrow{Z} Z|0\rangle = |0\rangle$$

We can also apply the Pauli Z gate to a qubit in the state $|1\rangle$ to flip its phase. The following equation shows how the Pauli Z gate will flip the phase of the qubit:

$$|1\rangle \xrightarrow{Z} Z|1\rangle = |1\rangle$$

The Pauli Z gate can be used to flip the phase of a qubit to any value. The value of the phase can be controlled by the phase of the Pauli Z gate.

- **Swap gate (SWAP):** The SWAP gate swaps the states of two qubits. The swap gate is an essential tool for quantum computing, and it is one of the most important two-qubit quantum gates. It can be used to implement a variety of

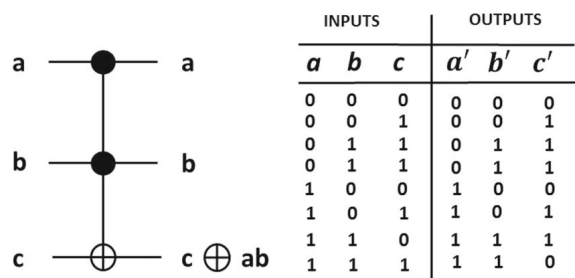


FIGURE 1.6: Swap gate gate (SWAP)

quantum algorithms, such as: Teleportation, Entanglement swapping, Quantum sorting, Quantum Fourier transform etc. Here is an example of how the swap gate can be used to teleport a qubit: Suppose we have two qubits, A and B, and we want to teleport the state of qubit A to qubit B. We can do this by applying the following quantum circuit:

```

1           H -- CNOT -- H
2           A      B      B
3

```

LISTING 1.3: Working of a SWAP Gate

The first Hadamard gate puts qubit A into a superposition of the $|0\rangle$ and $|1\rangle$ states. The CNOT gate then entangles qubits A and B. The second Hadamard gate then measures qubit A in the X-basis. If qubit A is in the $|0\rangle$ state, the measurement will collapse the state of qubit B to the $|0\rangle$ state. If qubit A is in the $|1\rangle$ state, the measurement will collapse the state of qubit B to the $|1\rangle$ state. The swap gate can also be used to implement entanglement swapping. Entanglement swapping is a process that allows two entangled pairs of qubits to be entangled with each other.

- Toffoli gate (Toffoli):** The Toffoli gate flips the state of the target qubit if the first two control qubits are both in the $|1\rangle$ state. It takes three qubits as input and outputs three qubits. The first two qubits are the control qubits, and the third qubit is the target qubit. If the first two qubits are both in the $|1\rangle$ state, then the Toffoli gate flips the state of the target qubit. Otherwise,

the Toffoli gate does not change the state of the target qubit. The Toffoli gate

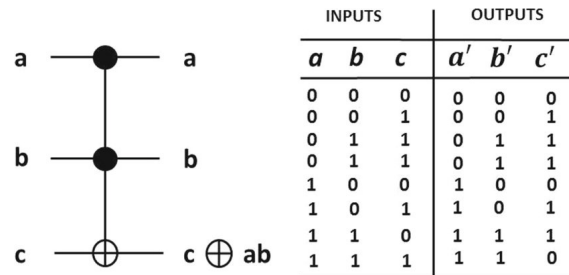


FIGURE 1.7: Toffoli gate

is an important tool for quantum computing. It is used in many quantum algorithms to implement a variety of quantum operations. Here is an example of how the Toffoli gate can be used to implement the controlled NOT gate: Suppose we have a qubit in the state $|0\rangle$ and a qubit in the state $|1\rangle$. We can apply the Toffoli gate to the two qubits to flip the state of the first qubit if the second qubit is in the $|1\rangle$ state. The following equation shows how the Toffoli gate will implement the controlled NOT gate:

$$\text{CNOT} \quad |0\rangle|1\rangle = Y|0\rangle|1\rangle = |1\rangle|1\rangle$$

The Toffoli gate can be used to implement any other quantum gate by using a combination of Toffoli gates and other quantum gates.

Chapter 2

Methodology

2.1 Qubit

Computation is a process of manipulating the states of a physical system to solve a problem. Quantum computing uses a microscopic object (e.g., electron, photon, ion) as the medium to store and transfer digital information. One-bit information (i.e., zero or one) can be encoded using two orthogonal states of a microscopic object. This quantum two-state system is called a quantum bit (or qubit). A quantum computer solves a problem by setting qubits in initial states and then manipulating the states so that an expected result appears on the qubits. In order to design such a quantum circuit, quantum mechanics is used to describe the states since those microscopic objects do not follow the rules of classical physics. The state of a qubit can be written as a vector $|\psi\rangle$.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \{\alpha, \beta \in \mathbb{C}\} \quad (2.1)$$

where α and β are complex numbers \mathbb{C} , called probability amplitude, and satisfy $|\alpha|^2 + |\beta|^2 = 1$. $|\alpha|^2$ is the probability of getting the state $|0\rangle$ as the result of the

measurement on the qubit $|\psi\rangle$ while $|\beta|^2$ is the probability of getting $|1\rangle$. “ $|\rangle$ ” is a standard notation for specifying states in quantum mechanics, called column vector or ket vector in the Dirac notation. The orthonormal basis $|0\rangle$ and $|1\rangle$ can be written as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (2.2)$$

Quantum states combine through the tensor product. For instance, two qubits state can be written as $|\psi_1\rangle \otimes |\psi_2\rangle$, where “ \otimes ” indicates a tensor product, or more compactly $|\psi_1\rangle|\psi_2\rangle$ or $|\psi_1\psi_2\rangle$. For example,

$$\begin{aligned} |\psi_1\rangle \otimes |\psi_2\rangle &= [\psi_{1,0}, \psi_{1,1}]^T \otimes [\psi_{2,0}, \psi_{2,1}]^T \\ &= [\psi_{1,0}, \psi_{2,0}, \psi_{1,0}, \psi_{2,1}, \psi_{1,1}, \psi_{2,0}, \psi_{1,1}, \psi_{2,1}]^T \end{aligned} \quad (2.3)$$

where $|\psi_i\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha[1, 0]^T + \beta[0, 1]^T = [\alpha, \beta]^T = [\psi_{i,0}, \psi_{i,1}]^T$.

Thus, an n-qubit state can be represented by a column vector with 2^n elements.

2.2 Superposition State

Superposition state is a key feature of quantum computing. In this section, we will use a photon as an example of a qubit for simplicity. A photon’s polarization (i.e., its geometric orientation) can represent one bit of information. A horizontally polarized photon represents classical bit 0, and a vertically polarized photon represents bit 1. In Figure 2.1, photons are fired at the emitter and go through the Filter-V, which only allows vertically polarized photons to pass through. We will assume that only one photon enters the Polarization Beam Splitter (PBS) at a time to simplify the example. The PBS transmits vertically polarized photons (which are then measured at the detector DV), while deflecting horizontally polarized photons

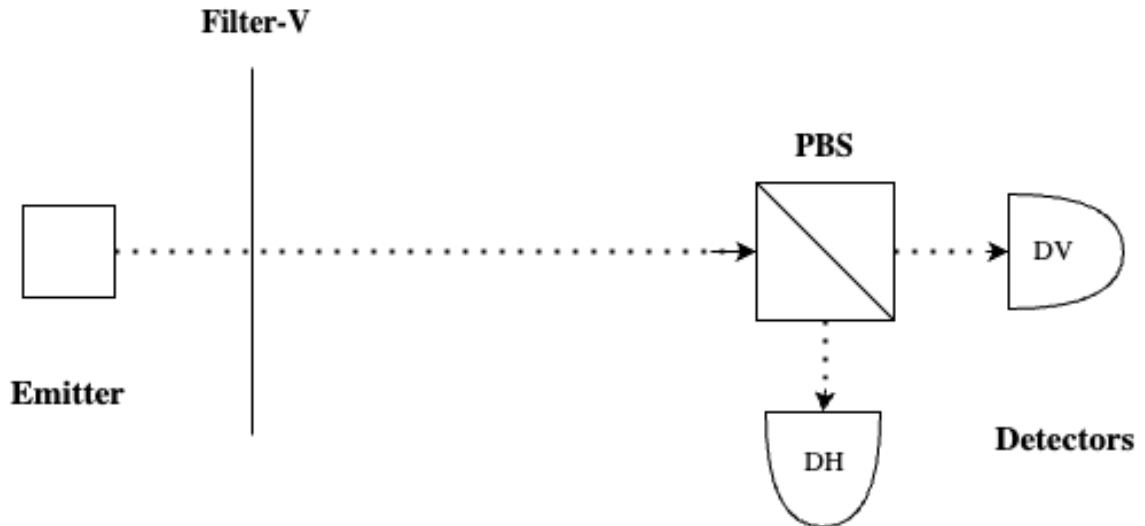


FIGURE 2.1: All Photons are detected in DV

(which are measured at the detector DH). Therefore, all photons will be measured at DV in Figure 2.1. If a filter that only transmits diagonally polarized photons called

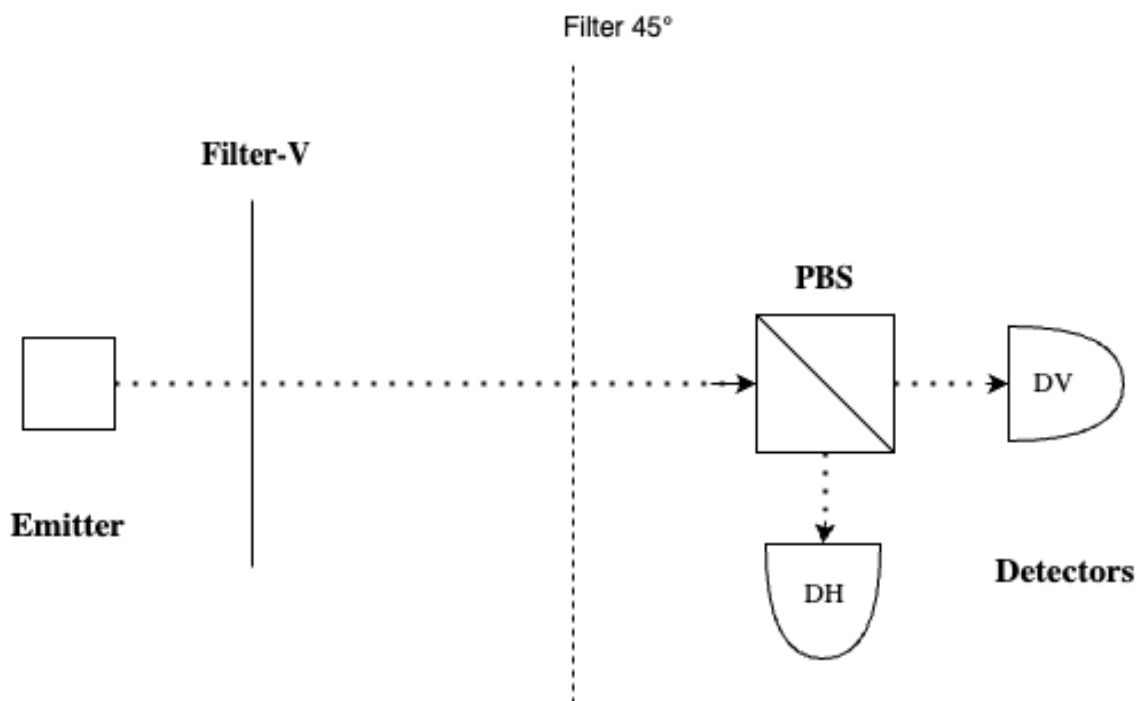


FIGURE 2.2: The half of photons are found at DV, and the rest are at DH

“Filter-45°”, however, is placed between Filter-V and PBS (Figure 2.2), a vertically

or horizontally polarized photon is found at each detector with the probability of $\frac{1}{2}$.

Since the probability of finding the horizontally polarized photon $|\psi\rangle = |H\rangle$ or vertically polarized photon $|\psi\rangle = |V\rangle$ at the PBS is $\frac{1}{2}$, the probability amplitudes α and β should be $\frac{1}{\sqrt{2}}$. Thus, the state of a photon just before PBS can be written as

$$|\psi\rangle = \frac{1}{\sqrt{2}}|H\rangle + \frac{1}{\sqrt{2}}|V\rangle \quad (2.4)$$

Since $|H\rangle$ is used to represent a classical bit “0” and $|V\rangle$ is used for “1”, the expression (2.4) is written as

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad (2.5)$$

When a photon is prepared in this state, the digital information encoded in the photon is “0” or “1”. We can interpret the equation (2.5) as meaning the states “0” and “1” exist at the same time. This unique state is called a superposition state. When two qubits are both in the superposition state (2.5), the state can be written as

$$\begin{aligned} |\psi_1\rangle|\psi_2\rangle &= \left\{ \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right\} \left\{ \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right\} \\ &= \frac{1}{2}[1, 1]^T \otimes [1, 1]^T = \frac{1}{2}[1, 1, 1, 1]^T \\ &= \frac{1}{2} \{ [1, 0, 0, 0]^T + [0, 1, 0, 0]^T + [0, 0, 1, 0]^T + [0, 0, 0, 1]^T \} \\ &= \frac{1}{2}|0\rangle|0\rangle + \frac{1}{2}|0\rangle|1\rangle + \frac{1}{2}|1\rangle|0\rangle + \frac{1}{2}|1\rangle|1\rangle \end{aligned} \quad (2.6)$$

This two-qubit state represents four classical binary states (00, 01, 10, 11) at the same time. When a quantum computer prepares n qubits in a superposition state as its input for a quantum circuit (Figure 3), 2^n possible inputs can be processed simultaneously. This quantum parallelism is one of the significant advantages of quantum computers.

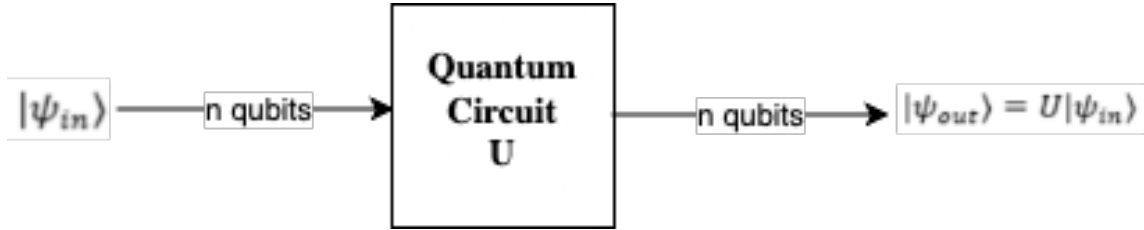


FIGURE 2.3: n-qubit quantum circuit

2.3 Quantum Circuit and Measurement

Quantum computers work by manipulating the quantum states of physical systems. A quantum computer takes a quantum state as input and controls the state to increase the probability of getting the answer in the output state. For example, Shor's algorithm uses quantum states to find the prime factors of a large number.

One of the key differences between classical and quantum computers is how they handle intermediate states. In a classical computer, we can measure the intermediate state because zero and one are represented by different voltages. This allows us to find errors by measuring the voltage. However, in a quantum computer, the intermediate states are likely to be in a superposition state. If we measure a superposition state, the quantum state is corrupted and becomes one of the two orthogonal base states, $|0\rangle$ or $|1\rangle$. This is a one-way operation, meaning that we cannot find the original state from the measured result.

For example, if we measure the photons in the equation (2.6), we will observe one of the four states (e.g., $|\psi_1\rangle\psi_2\rangle = |0\rangle|1\rangle$) with an equal probability (i.e., $\frac{1}{4}$). However, we will lose all other information about the original state (2.6). This is similar to a classical one-way function, such as a hash function.

In other words, quantum computers are able to perform certain calculations much faster than classical computers because they can manipulate quantum states in ways that classical computers cannot. However, quantum computers are also more fragile than classical computers because measuring a quantum state destroys it.

2.4 Quantum Gate and Reversibility

Quantum gates are like the basic instructions that a quantum computer can understand. Each gate operation can be mathematically represented with a matrix. They can be used to perform operations on qubits, such as creating superposition states or entangling qubits together. The Hadamard gate is a specific example of a quantum gate that can be used to create superposition states. For instance, the Filter-45° in Figure 2 converts from one of the orthogonal base states (i.e., $|0\rangle$) to a superposition state (i.e., $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$). This operation can be expressed with the following matrix U_H called Hadamard gate (Nielsen & Chuang, 2010).

$$U_H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \quad (2.7)$$

Another example of the quantum gate is the controlled-NOT (cNOT) gate (Figure 2.4), which behaves like a classical XOR gate, as shown in Table 2.1.

TABLE 2.1: A truth table for the quantum XOR gate with two inputs and two outputs

ψ_{in1}	ψ_{in2}	ψ_{out1}	ψ_{out2}
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

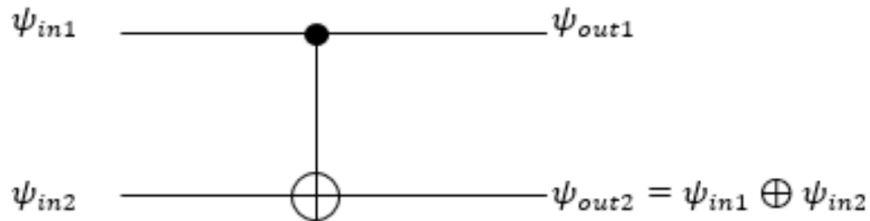


FIGURE 2.4: Controlled-Not gate behaves like a classical XOR

$$|\psi_1\rangle|\psi_2\rangle\& = U_{C_{NOT}}|\psi_{in1}\rangle|\psi_{in2}\rangle\& = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \psi_{1,0}\psi_{2,0} \\ \psi_{1,0}\psi_{2,1} \\ \psi_{1,1}\psi_{2,0} \\ \psi_{1,1}\psi_{2,1} \end{bmatrix} = \begin{bmatrix} \psi_{1,0}\psi_{2,0} \\ \psi_{1,0}\psi_{2,1} \\ \psi_{1,1}\psi_{2,1} \\ \psi_{1,1}\psi_{2,0} \end{bmatrix} \quad (2.8)$$

If we can construct a NAND gate with quantum gates, any logic gate (e.g., AND, OR, NOT) can be built since the NAND gate is universal (Mano, 1995). The classical NAND gate's truth table is given in Table 2.1.

TABLE 2.2: A truth table for the classical NAND gate with two inputs

IN_1	IN_2	Out
0	0	1
0	1	1
1	0	1
1	1	0

If the NAND gate is built based on the truth table above and the inputs are two qubits $|0\rangle|1\rangle = [1, 0]^T \otimes [0, 1]^T = [0, 1, 0, 0]^T$, the operation can be written as following (Yanofsky & Mannucci, 2008),

$$NAND = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \quad (2.9)$$

$$U_{NAND}|0\rangle|1\rangle = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} [0, 1, 0, 0]^T = [0, 1]^T = |1\rangle \quad (2.10)$$

However, this NAND gate cannot be realized with quantum gates because two-bit information before the gate becomes one bit after the gate in (2.10). Quantum mechanics does not allow the system (e.g., quantum circuit) to lose information unless the quantum states in the system are measured. Therefore, quantum gates must have the same number of inputs as the outputs and must be reversible with no information loss by the gates. In contrast, classical gates except NOT gate are

one-way functions and lose some of the input information at the exit of the gate. This requirement is another significant difference from classical computing.

TABLE 2.3: Truth table for Quantum NAND gate

ψ_{in1}	ψ_{in2}	ψ_{in3}	ψ_{out1}	ψ_{out2}	ψ_{out3}
0	0	1	0	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

A quantum NAND gate can be made of a Toffoli gate, also known as the controlled-controlled-NOT (ccNOT) gate (Figure 5, Table 4). The revised truth table for the quantum NAND gate is given in Table 3.

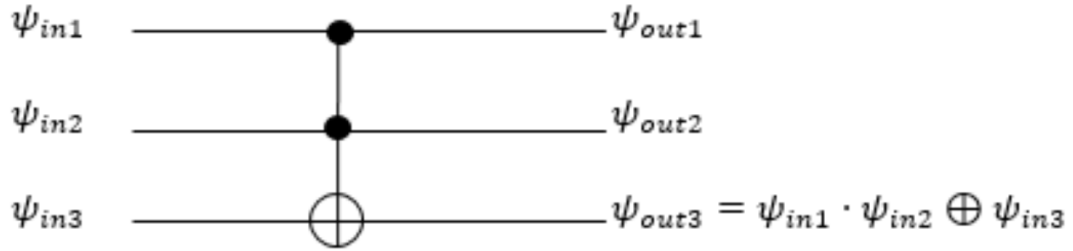


FIGURE 2.5: Controlled-Controlled-NOT (Toffoli gate)

TABLE 2.4: Truth table for Controlled-Controlled-NOT

ψ_{in1}	ψ_{in2}	ψ_{in3}	ψ_{out1}	ψ_{out2}	ψ_{out3}	
0	0	0	0	0	0	
0	0	1	0	0	1	
0	1	0	0	1	0	
0	1	1	0	1	1	
1	0	0	1	0	0	
1	0	1	1	0	1	
1	1	0	1	1	1	
1	1	1	1	1	0	

When the Toffoli gate with $\psi_{in3} = 1$, the ccNOT gate works as a NAND gate.

$$\psi_{out3} = \neg(\psi_{in1} \oplus \psi_{in2}) \quad (2.11)$$

Thus, we can build any digital logic with the quantum gates theoretically.

No-Cloning Theorem and Entangled State When ψ_{in2} is zero in Table 1, the cNOT gate keeps the ψ_{out2} to be zero for $\psi_{in1} = 0$ and changes ψ_{out2} to be one for $\psi_{in1} = 1$. Thus, it seems that cNOT gate copies the classical bit information in ψ_{in1} to ψ_{in2} when $\psi_{in2} = 0$.

$$\begin{aligned} |\psi_{out1}\rangle|\psi_{out2}\rangle &= \text{cNOT}|0\rangle|0\rangle = |0\rangle|0\rangle \\ |\psi_{out1}\rangle|\psi_{out2}\rangle &= \text{cNOT}|1\rangle|0\rangle = |1\rangle|1\rangle \end{aligned} \quad (2.12)$$

If cNOT gate can copy an arbitrary state in a qubit to the other qubit, it should be valid for a superposition state. When the input ψ_{in1} is $\alpha|0\rangle + \beta|1\rangle$, the output ψ_{out2} should be $\alpha|0\rangle + \beta|1\rangle$.

$$\begin{aligned} |\psi_{out1}\rangle|\psi_{out2}\rangle &= \text{cNOT}(\alpha|0\rangle + \beta|1\rangle)|0\rangle \\ &= (\alpha|0\rangle + \beta|1\rangle)(\alpha|0\rangle + \beta|1\rangle) \\ &= \alpha^2|0\rangle|0\rangle + \alpha\beta|1\rangle|0\rangle + \beta\alpha|0\rangle|1\rangle + \beta^2|1\rangle|1\rangle \end{aligned} \quad (2.13)$$

However, from (2.12), the output from the cNOT gate for the superposition state turns out to be $\alpha|0\rangle|0\rangle + \beta|1\rangle|1\rangle$.

$$\begin{aligned} |\psi_{out1}\rangle|\psi_{out2}\rangle &= \text{cNOT}(\alpha|0\rangle + \beta|1\rangle)|0\rangle \\ &= \text{cNOT}(\alpha|0\rangle|0\rangle + \beta|1\rangle|0\rangle) \\ &= \alpha|0\rangle|0\rangle + \beta|1\rangle|1\rangle \end{aligned} \quad (2.14)$$

Obviously, the equation (2.13) is not equal to (2.14). In quantum mechanics, the replication of an arbitrary quantum state is not possible. This restriction is known as

the no-cloning theorem (Nielsen & Chuang, 2010 ; Wootters & Zurek, 1982). Even if cNOT gate seems to copy the classical bit information in ψ_{in1} to ψ_{out2} , this is not a classical meaning of “copy”. The results in (12) are the special cases for $\alpha = 1$ or $\beta = 1$ ($|\alpha|^2 + |\beta|^2 = 1$). Also, the resulting states in (2.14) is quite impressive. The equation says, when we observe zero in ψ_{out1} by measurement, we know ψ_{out2} is also zero with no additional measurement. Similarly, when we find one in ψ_{out1} , we know ψ_{out2} is also one without measuring ψ_{out2} . In other words, the quantum state in ψ_{out2} depends on the value observed in $|\psi_{out1}\rangle$. Thus, the outputs ψ_{out1} and ψ_{out2} cannot be described independently. This bizarre states, where the individual states of qubits are intimately related to one another, is called the entangled state. There is no way to express the entangled states as separable states like the expression $|\psi_{out1}\rangle = (\alpha|0\rangle + \beta|1\rangle)$ and $|\psi_{out2}\rangle = (\alpha|0\rangle + \beta|1\rangle)$ in (2.13). The use of entangled states is another essential ingredient of quantum computing.

2.5 Quantum Circuits and Algorithms Developed

2.5.1 Increment

Takes an arbitrary input state $|x\rangle$ on n and takes it to the $|(x + 1) \bmod N\rangle$, where $N = 2^n$ state in the binary representation, which can be done by applying the following circuit U .

Let $|x\rangle$ be an arbitrary state on n qubits,

$$U|x\rangle = |(x + 1) \bmod N\rangle \quad (2.15)$$

Where U defines an operation that applies multiple 1 multi-controlled X gates and flips the first qubit.

Here we give an example on $n = 3$ qubits.

TABLE 2.5: Truth table for Quantum Increment Operation

Initial State $ x\rangle$	Final State $U x\rangle$
$ 000\rangle$	$ 001\rangle$
$ 001\rangle$	$ 010\rangle$
$ 010\rangle$	$ 011\rangle$
$ 011\rangle$	$ 100\rangle$
$ 100\rangle$	$ 101\rangle$
$ 101\rangle$	$ 110\rangle$
$ 110\rangle$	$ 111\rangle$
$ 111\rangle$	$ 000\rangle$

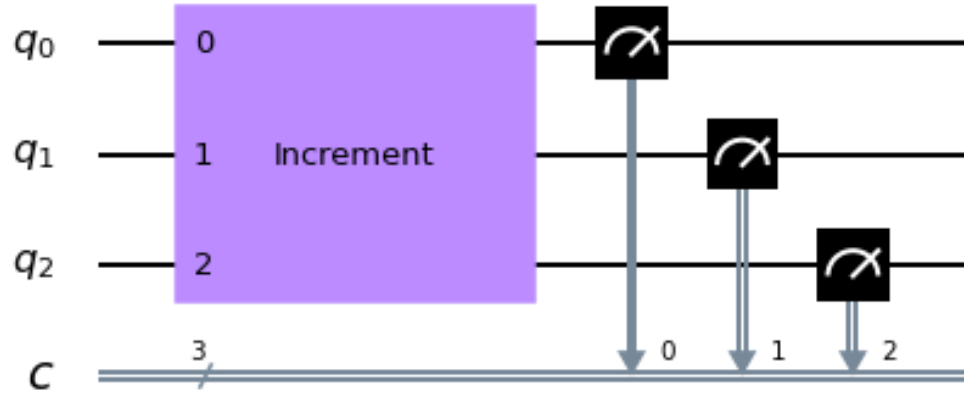


FIGURE 2.6: Quantum Circuit Diagram of **Increment**

2.5.2 Decrement

Takes an arbitrary input state $|x\rangle$ on n and takes it to the $|(x - 1) \bmod N\rangle$, where $N = 2^n$ state in the binary representation, which can be done by applying the following circuit U . Let $|x\rangle$ be an arbitrary state on n qubits,

$$U|x\rangle = |(x - 1) \bmod N\rangle \quad (2.16)$$

Where U defines an operation that applies multiple 0 multi-controlled X gates and flipping the first qubit.

Here we give an example on $n = 3$ qubits.

TABLE 2.6: Truth table for Quantum Decrement Operation

Initial State $ x\rangle$	Final State $U x\rangle$
$ 000\rangle$	$ 111\rangle$
$ 001\rangle$	$ 000\rangle$
$ 010\rangle$	$ 001\rangle$
$ 011\rangle$	$ 010\rangle$
$ 100\rangle$	$ 011\rangle$
$ 101\rangle$	$ 100\rangle$
$ 110\rangle$	$ 101\rangle$
$ 111\rangle$	$ 110\rangle$

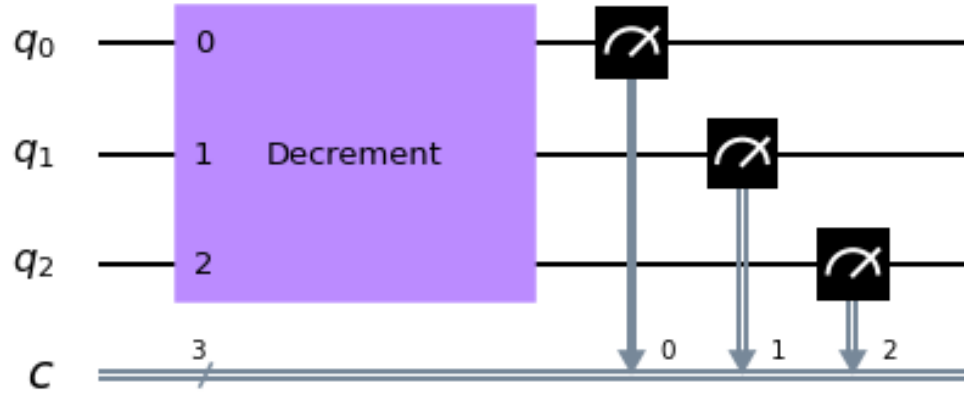


FIGURE 2.7: Quantum Circuit Diagram of **Decrement**

2.5.3 Build BELL State

Receive a state on two qubits $|xy\rangle$ and takes it to a given Bell State. This operation is done by applying the U gate $U|x\rangle = |\beta_{xy}\rangle$ as follows,

$$|\beta_{00}^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$|\beta_{01}^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$

$$|\beta_{10}^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

$$|\beta_{11}^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$

Where U defines an operation that generally applies a Hadamard gate H on the first qubit and a $CNOT_{01}$. And circumstantially, depending on xy applies a X gate on y and Z on x .

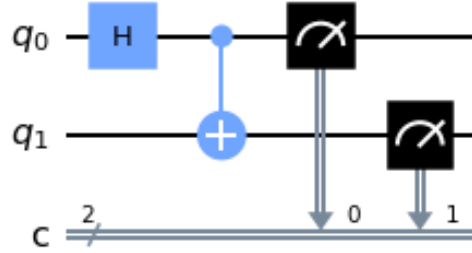


FIGURE 2.8: Quantum Circuit Diagram of **Build BELL** State

2.5.4 Build GHZ State

Takes an input state $|x\rangle = |000\dots 0\rangle$ and takes it to the GHZ state $\frac{1}{\sqrt{2}}(|0\dots 0\rangle + |1\dots 1\rangle)$, which can be done by applying the following circuit U .

Let $|x\rangle$ be a all zeros state on n qubits,

$$U|x\rangle = \frac{1}{\sqrt{2}}(|000\dots 0\rangle + |111\dots 1\rangle) \quad (2.17)$$

Where U defines an operation that applies a Hadamard gate H on the first qubit and Controlled X gates, $CNOT$, on the remaining ones, taking the control qubit as the first one and the targets from the second to the n -th qubit.

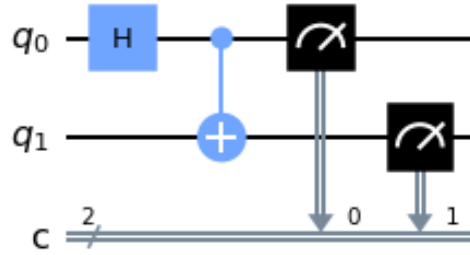


FIGURE 2.9: Quantum Circuit Diagram of **Build GHZ** State

2.5.5 Quantum Fourier Transform

Takes an input state $|x\rangle$ on the computational basis and takes it to the Fourier basis, which can be done by applying the following circuit U .

Let $|x\rangle$ be a all zeros state on n qubits,

$$U|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \exp\left(\frac{2\pi ixy}{N}\right) |y\rangle \quad (2.18)$$

You can find a detailed explanation about the QFT right here.

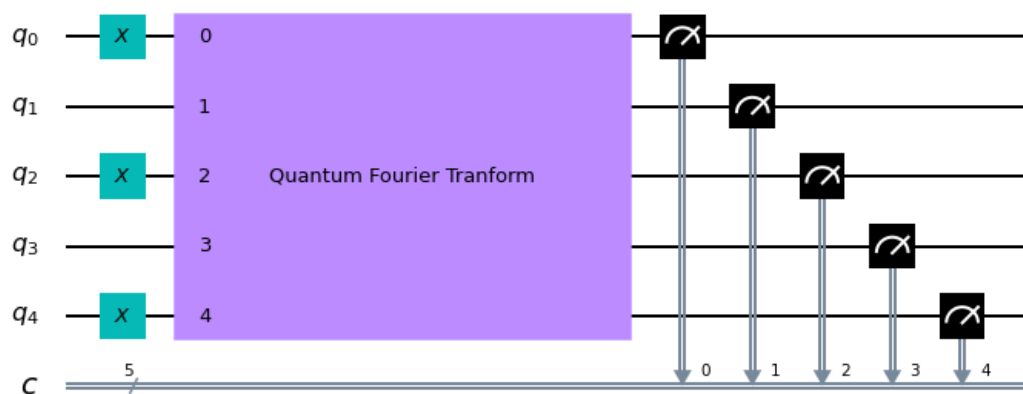


FIGURE 2.10: Quantum Circuit Diagram of **Quantum Fourier Transform**

Results and Observations

2.6 Increment

The graph shows the probability of a particle moving in a certain direction after passing through a quantum circuit. The quantum circuit consists of a Hadamard gate followed by a controlled-NOT gate.

The Hadamard gate is a single-qubit gate that puts the qubit in an equal superposition of the $|0\rangle$ and $|1\rangle$ states. The controlled-NOT gate is a two-qubit gate that flips the target qubit if the control qubit is in the $|1\rangle$ state.

The graph shows that the probability of the particle moving in the $+x$ direction is high, while the probability of the particle moving in the $-x$ direction is low. This is because the quantum circuit is designed to amplify the probability of the particle moving in the $+x$ direction.

To understand why this is so, let's consider the following:

- The Hadamard gate puts the qubit in an equal superposition of the $|0\rangle$ and $|1\rangle$ states. This means that the particle has an equal chance of moving in the $+x$ or $-x$ direction after passing through the Hadamard gate.

- The controlled-NOT gate flips the target qubit if the control qubit is in the $|1\rangle$ state. This means that if the particle is in the $|1\rangle$ state after passing through the Hadamard gate, the controlled-NOT gate will flip it to the $|0\rangle$ state. This will cause the particle to move in the $+x$ direction.

Therefore, the quantum circuit amplifies the probability of the particle moving in the $+x$ direction because the controlled-NOT gate will flip the particle to the $|0\rangle$ state if it is in the $|1\rangle$ state after passing through the Hadamard gate.

In summary, the graph you provided shows that the quantum circuit is designed to amplify the probability of a particle moving in the $+x$ direction. This is because the quantum circuit consists of a Hadamard gate followed by a controlled-NOT gate, and the controlled-NOT gate will flip the particle to the $|0\rangle$ state if it is in the $|1\rangle$ state after passing through the Hadamard gate.

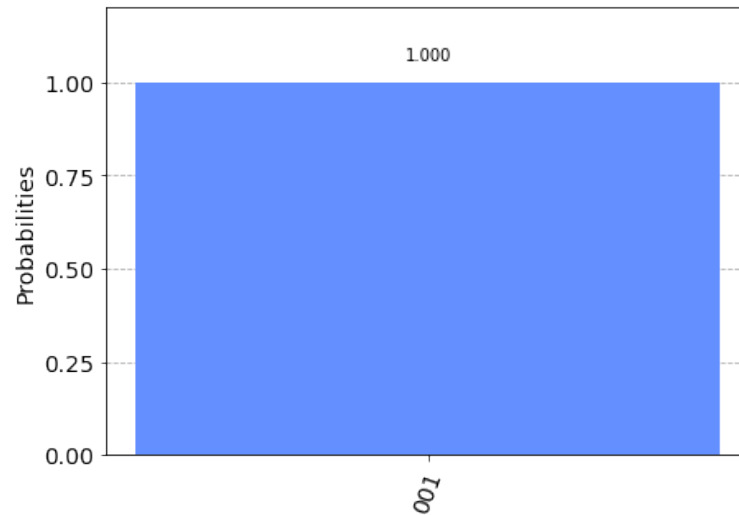


FIGURE 2.11: Probabilities for High Result VS. For Quantum State 001

2.7 Decrement

The graph you sent shows the probability of a particle being in a certain state after passing through a quantum circuit. The quantum circuit is designed to implement the following equation:

$$U|x\rangle = |(x - 1) \bmod N\rangle$$

This equation represents the operation of decrementing an n-bit quantum register by one.

The graph shows that the probability of the particle being in the state $|(x-1) \bmod N\rangle$ is high, while the probability of the particle being in any other state is low. This is because the quantum circuit is designed to amplify the probability of the particle being in the state $U|x\rangle = |(x - 1) \bmod N\rangle$.

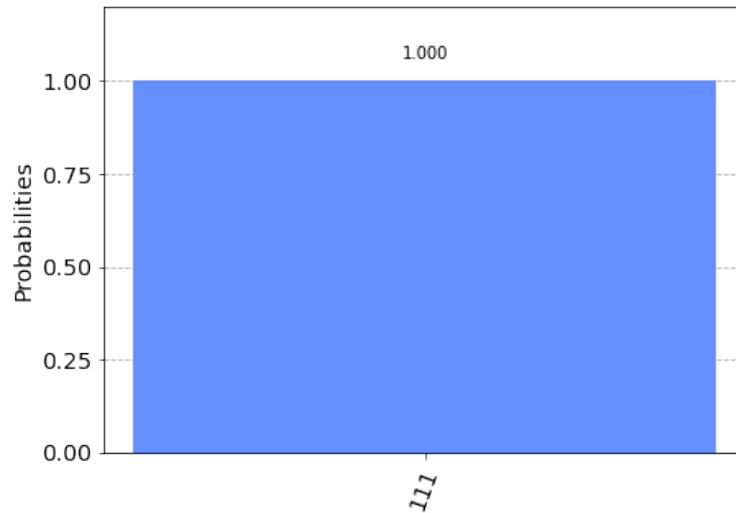


FIGURE 2.12: Probabilities for Low Result VS. For Quantum State 111

Here is an example of how the circuit would work to decrement a 3-bit quantum register:

- The Hadamard gate is applied to the MSB, putting it in a superposition of the $|0\rangle$ and $|1\rangle$ states.

- The CNOT gate is applied with the MSB as the control qubit and the NSB as the target qubit. This flips the NSB if the MSB is in the $|1\rangle$ state.
- The X gate is applied to the MSB, flipping it.
- The CNOT gate is applied with the NSB as the control qubit and the bit to the right of it as the target qubit. This flips the bit to the right of the NSB if the NSB is in the $|1\rangle$ state.
- The process is repeated until all of the bits in the quantum register have been decremented.

The output of the circuit is a 3-bit quantum register that is in the state $U|x\rangle = |(x - 1) \bmod N\rangle$. Quantum decrement circuits are an important part of quantum computing. They are used in many quantum algorithms and circuits, and they are essential for performing arithmetic operations on quantum computers.

2.8 Build BELL State

The histogram plot for the Build GHZ Quantum Circuit (Of Probabilities VS. Quantum State of $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$) shows the probability of finding the quantum circuit in the state $|00\rangle$, $|01\rangle$, $|10\rangle$ or $|11\rangle$ respectively after the circuit is applied.

The histogram shows that the probability of finding the quantum circuit in the state $|01\rangle$ or $|10\rangle$ is very low, while the probability of finding the circuit in the state $|00\rangle$ or $|11\rangle$ is higher but similar.

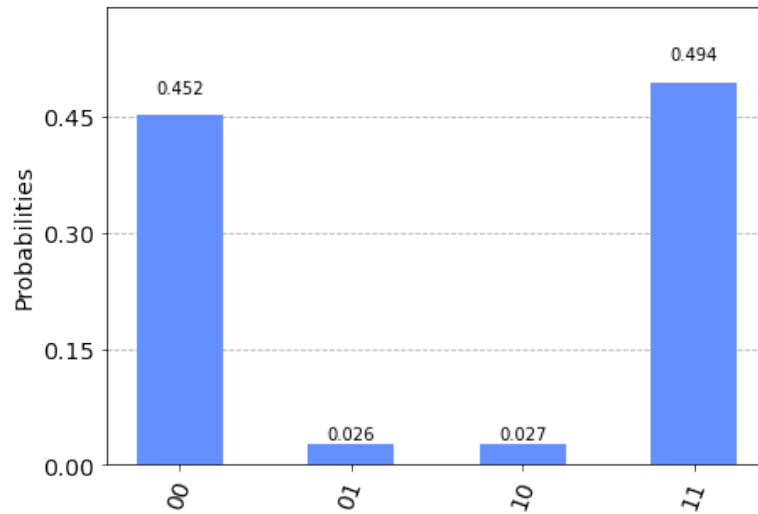


FIGURE 2.13: Probability to into Respective Quantum State after Operation for Each Quantum State of 00, 01, 10 and 11

The Bloch QSphere is a three-dimensional sphere that is used to visualize the state of a qubit. The x-axis of the Bloch QSphere represents the real part of the qubit's wavefunction, the y-axis represents the imaginary part of the qubit's wave function, and the z-axis represents the probability of the qubit being in the $|0\rangle$ or $|1\rangle$ state.

The Build BELL State $|10\rangle$ state experiment is used to create a Bell state — $|10\rangle$ state, which is a maximally entangled state of two qubits. A maximally entangled state is a state where the two qubits are in a superposition of all possible states, and the measurement of one qubit will instantly collapse the state of the other qubit.

The results of the Build BELL State $|10\rangle$ state for Bloch QSphere experiment show that the experiment is successful in creating a Bell state $|10\rangle$ state. This is because the results show that the two qubits are in a superposition of all possible states, and the measurement of one qubit will instantly collapse the state of the other qubit.

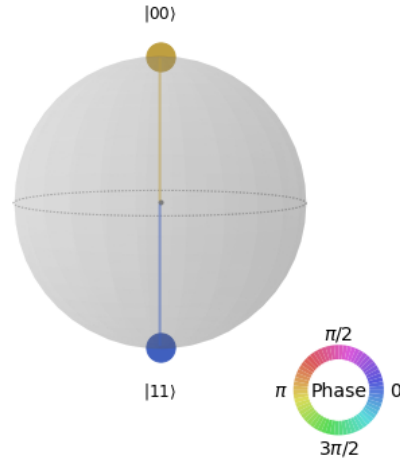
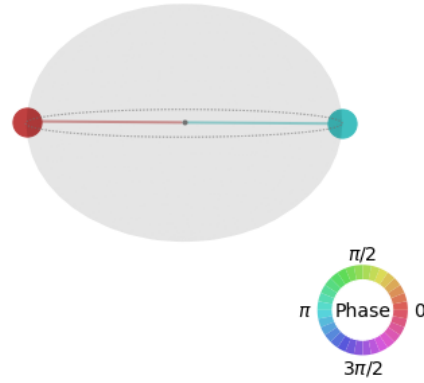
Here is a more detailed explanation of the results:

- The x-axis of the image shows the real part of the qubit's wave function.
- The y-axis of the image shows the imaginary part of the qubit's wave function.
- The z-axis of the image shows the probability of the qubit being in the $|1\rangle$ state.
- The two points on the Bloch QSphere represent the two qubits in the Bell state $|10\rangle$ state.

The results show that the two qubits are in a superposition of all possible states. This is because the two points on the Bloch QSphere are located at the north pole of the sphere, which means that they are both in the $|1\rangle$ state with 100% probability.

The results also show that the measurement of one qubit will instantly collapse the state of the other qubit. This is because the two points on the Bloch QSphere are entangled, which means that they are linked together in such a way that a measurement of one qubit will instantly affect the state of the other qubit.

The Build BELL State $|10\rangle$ state for Bloch QSphere experiment is an important experiment in quantum computing. It is used to demonstrate the ability to create and manipulate maximally entangled states of qubits. Maximally entangled states are essential for implementing a variety of quantum algorithms, such as quantum teleportation and quantum entanglement swapping.

FIGURE 2.14: Plot State **qsphere** for Bell State 10 qsphereFIGURE 2.15: Plot State **qsphere** for Bell State 11 qsphere

2.9 Build GHZ State

The histogram plot for the Build GHZ Quantum Circuit (Of Probabilities VS. Quantum State of $|00000\rangle$ and $|11111\rangle$) shows the probability of finding the quantum circuit in the state $|00000\rangle$ or $|11111\rangle$ after the circuit is applied.

The histogram shows that the probability of finding the quantum circuit in the state $|00000\rangle$ is comparatively low, while the probability of finding the circuit in the state

$|11111\rangle$ is higher. This is because the GHZ quantum circuit is designed to produce a maximally entangled state, which is a state where all of the qubits are in the same superposition.

Maximally entangled states are useful for quantum computing because they can be used to perform certain types of calculations much faster than classical computers. For example, the GHZ quantum circuit can be used to implement Grover's algorithm, which is a quantum algorithm for searching a database that is much faster than any classical algorithm.

Here is a short explanation of the histogram plot:

- The x-axis of the plot shows the quantum state of the circuit, which can be either $|00000\rangle$ or $|11111\rangle$.
- The y-axis of the plot shows the probability of finding the circuit in the corresponding quantum state.
- The bars in the histogram show the probability of finding the circuit in each quantum state.

The bars in the histogram show the probability of finding the circuit in each quantum state.

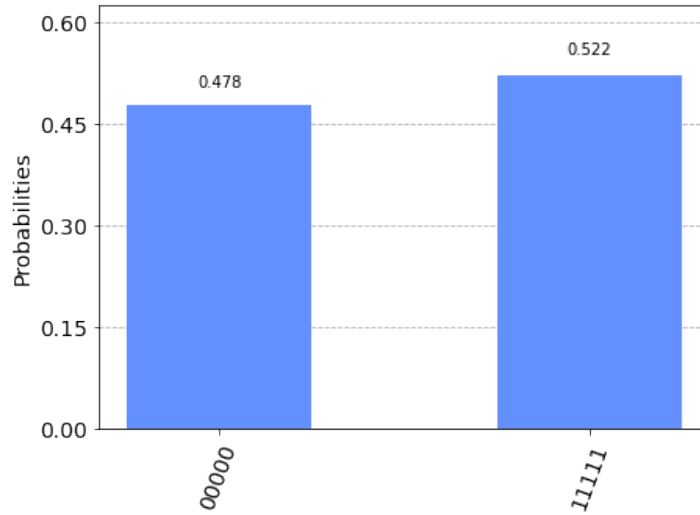


FIGURE 2.16: Probabilities for GHZ State

2.10 Quantum Fourier Transform

A histogram plot for the quantum Fourier transform (QFT) for a 5-bit quantum bit (qubit) shows the probability of measuring each possible output state of the QFT.

The QFT is a quantum algorithm that takes a quantum state as input and outputs a quantum state that is the Fourier transform of the input state. The Fourier transform is a mathematical operation that converts a signal from the time domain to the frequency domain. In the context of quantum computing, the QFT can be used to convert a quantum state from the computational basis to the Fourier basis.

The computational basis is the basis in which the qubits are in either the $|0\rangle$ or $|1\rangle$ state. The Fourier basis is the basis in which the qubits are in a superposition of the $|0\rangle$ and $|1\rangle$ states. The QFT can be used to convert between the two bases because the Fourier transform is a unitary transformation, which means that it preserves the probability of the quantum state.

The histogram plot for the QFT for a 5-bit qubit shows the probability of measuring each possible output state of the QFT. The output states of the QFT are the Fourier

coefficients of the input state. The Fourier coefficients are the amplitudes of the different frequencies that make up the input signal.

The histogram plot for the QFT for a 5-bit qubit will show a peak at each Fourier coefficient. The height of each peak will be proportional to the amplitude of the corresponding frequency in the input signal.

Here is an example of a histogram plot for the QFT for a 5-bit qubits:

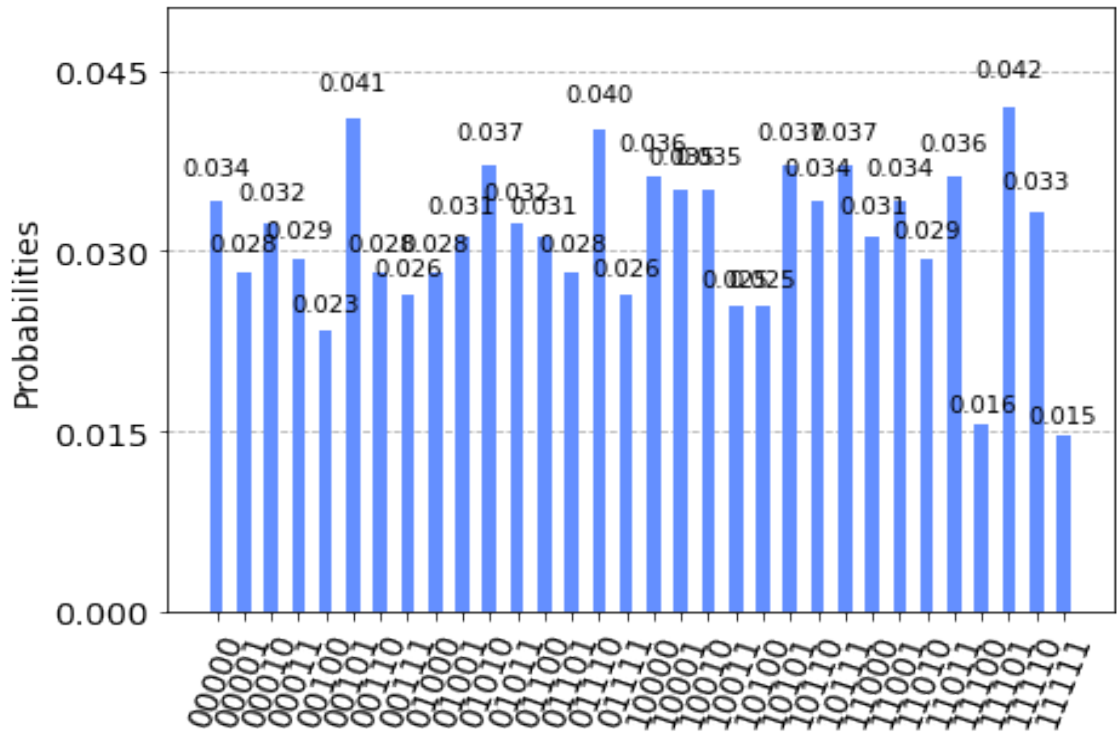


FIGURE 2.17: Histogram Plot for QFT in 5 - QB-its

The x-axis of the plot shows the Fourier coefficient, and the y-axis shows the probability of measuring the corresponding output state. The plot shows that the highest peak is at Fourier coefficient 0, which means that the input signal has a strong DC component. The plot also shows that there are smaller peaks at other Fourier coefficients, which means that the input signal also has some higher frequency components.

The QFT is a powerful tool for quantum computing. It can be used to implement a variety of quantum algorithms, such as Shor's algorithm and Grover's algorithm. Shor's algorithm is a quantum algorithm for factoring integers, and Grover's algorithm is a quantum algorithm for searching unsorted databases.

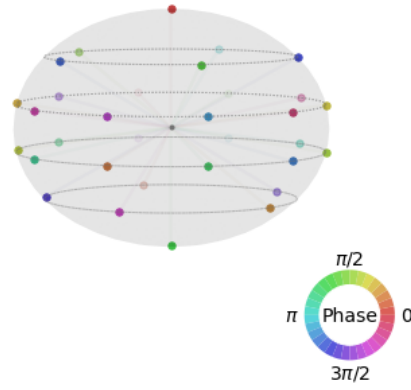


FIGURE 2.18: Plot State **qsphere** for Computational basis state qsphere

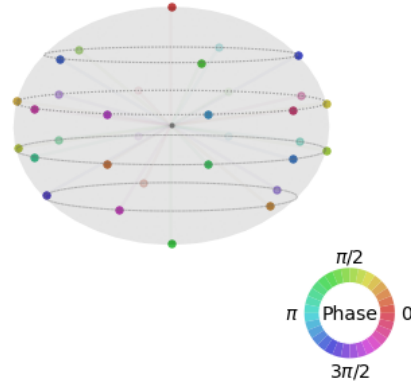


FIGURE 2.19: Plot State **qsphere** for Fourier basis state qsphere

Summary

In this Work we Implemented and describe how to simulate quantum circuits using the Qiskit SDK. They implemented the following algorithms and obtained satisfactory results: Increment, Decrement, Build GHZ, Build BELL, Quantum Fourier Transform.

These algorithms are all fundamental to quantum computing, and their successful simulation using Qiskit demonstrates the power and capabilities of the Qiskit SDK.

The increment and decrement algorithms are simple algorithms that add or subtract one to a quantum register, respectively. The Build Bell and Build GHZ algorithms create Bell and GHZ states, which are important entangled states used in quantum computing. The Quantum Fourier Transform algorithm is a quantum algorithm that performs the Fourier transform on a quantum register.

The work' results show that Qiskit can be used to simulate a variety of quantum algorithms and circuits. This demonstrates the potential of Qiskit to be used as a tool for developing and testing quantum algorithms and circuits, as well as for studying quantum phenomena.

Future Work

The algorithms and circuits that we have implemented in this Thesis Project serve as the basis for the development of more complex circuits and algorithms in the following ways:

- **Increment and Decrement:** These simple circuits can be used to build more complex arithmetic circuits, such as adders and multipliers. Arithmetic circuits are essential for many quantum algorithms, such as Shor's algorithm for integer factorization and Grover's algorithm for searching.
- **Build Bell and Build GHZ:** These circuits can be used to generate entangled states, which are essential for many quantum algorithms, such as quantum teleportation and quantum cryptography.
- **Quantum Fourier Transform (QFT):** The QFT is a fundamental quantum algorithm that has many applications, such as quantum phase estimation and quantum machine learning.

Here are some specific examples of how the algorithms and circuits that you have implemented can be used to develop more complex circuits and algorithms:

- **Quantum Adder:** A quantum adder can be built using a combination of increment and decrement circuits. Quantum adders are essential for many quantum algorithms, such as Shor's algorithm for integer factorization.

- **Quantum Multiplier:** A quantum multiplier can be built using a combination of quantum adders and other circuits. Quantum multipliers are essential for many quantum algorithms, such as quantum search algorithms.
- **Quantum Teleportation:** Quantum teleportation is a process of transferring the quantum state of one qubit to another qubit over a distance. Quantum teleportation can be implemented using Bell states and other circuits.
- **Quantum Cryptography:** Quantum cryptography is a type of cryptography that uses quantum mechanics to protect data. Quantum cryptography can be implemented using entangled states and other circuits.
- **Quantum phase estimation:** Quantum phase estimation is a process of estimating the phase of a quantum state. Quantum phase estimation is an essential subroutine for many quantum algorithms, such as Shor's algorithm for integer factorization.
- **Quantum Machine Learning:** Quantum machine learning is a type of machine learning that uses quantum mechanics to improve the performance of machine learning algorithms. Quantum machine learning algorithms often use the QFT and other quantum circuits.

In addition to the specific examples above, the algorithms and circuits that you have implemented can be used to develop more complex circuits and algorithms in many other ways. For example, you could use your work to develop new quantum algorithms for searching, sorting, and optimization. You could also use your work to develop new quantum algorithms for scientific and engineering simulations.

Bibliography

- [1] R. H. (Bo) Ewald. An introduction to quantum computing and its application. In (*Vol. 11413 LNCS, pp. 3–8*). Springer Verlag. https://doi.org/10.1007/978-3-030-14082-3_1, 2019.
- [2] M. Coggins. *Introduction to Quantum Computing with Qiskit*. Scarborough Quantum Computing Ltd, December 2021.
- [3] R. P. da Silva. *Quantum Circuits with Qiskit*. PhD thesis, SSRN Electronic Journal, February 15, 2023.
- [4] J. A. F. Fathelrhman Mohammed, Ibrahim Savran. Quantum fourier transform :using five qubits on qiskit.
- [5] . C. I. L. Nielsen, M.A. Quantum computation and quantum information (10th anniv). In *Cambridge University Press.*, 2010.
- [6] R. C. S. Pimenta and A. Bezerra. Leveraging qiskit to teach basic quantum computation algorithms. *Parana Journal of Science and Education*, July 2021.
- [7] Q. D. Team. Qiskit. In <https://qiskit.org>, 2023.
- [8] Q. Tutorials. <https://nbviewer.jupyter.org/github/qiskit/qiskit-tutorial/blob/master/index.ipynb>.
- [9] M. M. A. Yanofsky, N. S. Quantum computing for computer scientists. In *New York, NY: Cambridge University Press*, 2008.