Introduction to Quantum Computing

Dr. Thyagaraju G S

Professor and HoD, Department of CSE,

SDM Institute Of Technology, Ujire-574240

Topics :

1. **Introduction to Quantum Computing**
   1. **Why Quantum Computing?**
   2. **What is Quantum Computing ?**
   3. **Fundamental Principles of Quantum Computing**
2. **Qubits**
3. **Introduction to Python Qiskit**
4. **Quantum Gates and Quantum Circuits**
5. **Introduction to Quantum Algorithms**
6. **Future Directions and Research Opportunities**

1.0 Why Quantum Computing?

1.1 Primary reasons

1. **Enhanced Problem-Solving/Computing Capabilities**
   - Solving complex problems that involve numerous variables and uncertainties
2. **Exponential Scaling**
   - Power of quantum computers increases exponentially with the addition of qubits
3. **Energy Efficiency**
   - Lower energy consumption and reduced carbon emissions
4. **Communication**
   - Provide better security and improved long-distance communications,
5. **Sensing**
   - Extremely precise measurements

1.2 Primary Applications

1. **Chemistry and Materials Science**
   - Aiding in **drug discovery** and **materials development**
2. **Logistics and Optimization**
   - Optimize logistics and route planning
3. **Cryptography**
   - Quantum-resistant algorithms, secure communication networks using quantum key distribution (QKD)
4. **Artificial Intelligence**
   - Handling complex datasets and Models
5. **Weather Forecasting and Climate Change/Disaster Management**
   - by simulating complex atmospheric and oceanic systems more accurately

3.0 What is Quantum Computing ?

3.1 What is Quantum ?

- The term "**quantum”** (plural: ***quanta***) originates from the Latin word for "**how much**.“
- **Quantum** refers to the smallest possible **discrete unit** of any physical property, usually related to **energy and matter**.
- **Example :**
  - A quantum of **light** is a ***photon***, and
  - A quantum of **electricity** is an ***electron***
- ***Quantum Particles***
  - **Fermions (Particles that make up matter):** Electrons , Protons , Neutrons, Quarks, Neutrinos
  - **Bosons (Force-carrying particles):** Photons, Gluons, W and Z Bosons, Higgs Boson, Gravitons (Theoretical)
  - **Composite Particles:** Mesons, Baryons

3.2 What is Quantum Computing?

- Quantum computing is a type of computing that leverages the **principles of quantum mechanics** to process the information.
- Quantum mechanics is the theory **that describes the behavior** of microscopic systems, such as **photons, electrons, atoms, molecules, etc.**

Principles of Quantum Mechanics
(Unique Characteristics of Quantum Particles)

1. **Wave-Particle Duality**
2. **Superposition**
3. **Entanglement**
4. **Quantization**
5. **Uncertainty Principle**
6. **Probability and Wave functions**

4.0 Fundamental Principles of Quantum Computing

4.1 Qubits

4.2 Superposition

4.3 Entanglement

4.4 Quantum Interference

4.5 Decoherence

4.6 Quantum Tunneling

4.7 Measurement and Collapse

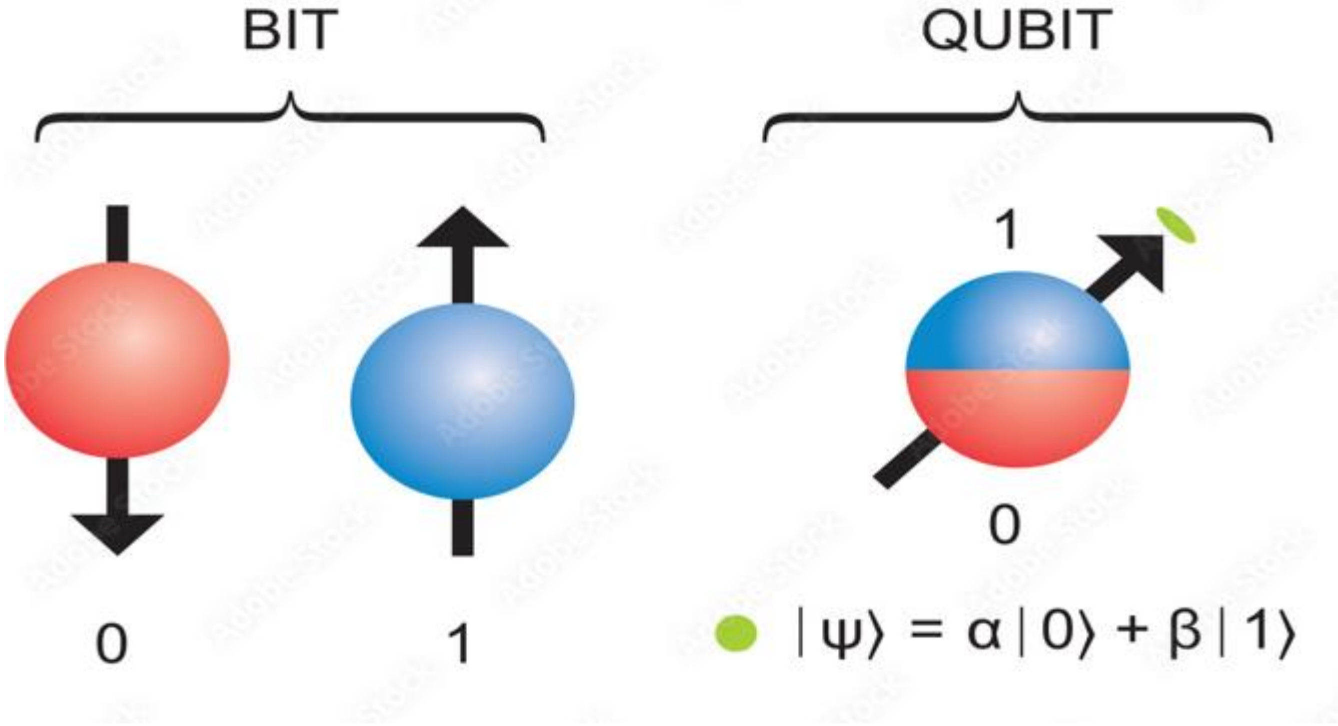
4 Core Components of Quantum Computer

A quantum computer is a device that exploits quantum mechanical phenomena, such as superposition and entanglement, to perform computations.

1. **Qubits**
2. **Quantum Registers**
3. **Quantum Gates**
4. **Quantum Circuits**
5. **Quantum Processing Unit (QPU)**
6. **Measurement Devices**

1. What is a Qubit?

- A qubit, or quantum bit, is the fundamental unit of information in quantum computing, analogous to a classical bit in traditional computing.



Key Characteristics of Qubits

1. **Superposition:** A qubit can exist in a superposition of both states simultaneously. This means that a qubit can represent 0, 1, or any combination of the two at the same time.
   1. Mathematically, a qubit can be expressed as: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
   2. where α and β are complex numbers representing the probability amplitudes of the qubit being in state $|0\rangle$ or $|1\rangle$, respectively.
2. **Entanglement**: Qubits can be entangled, meaning the ***state of one qubit is directly related to the state of another***, regardless of the distance separating them.
3. **Measurement:** When a qubit is measured, it collapses from its superposition state to one of the basis states (either $|0\rangle$ or $|1\rangle$). The outcome of the measurement is probabilistic.

Type of particles used to build Qubits

ELECTRONS

Electrons are subatomic particles with a negative charge. An electron can be seen as a single unit of electricity

ATOMS

Atoms are the smallest neutral building blocks of matter

PHOTONS

Photons are the single units of light, they can be used as single photon particles or as larger beams (continuous variables)

QUBIT HARDWARE: ELECTRONS

Example: Superconducting circuits

Superconducting qubits: IBM creates transmon qubits using niobium and aluminium on a silicon substrate

Superconduction is the phenomena of free flowing electricity, without any resistance. When put in a circuit, superconductors can be used to make qubits.

Google amazon IBM rigetti IQM ALICE & BOB

ADVANTAGES

- Closest to classical computer chips, can therefore leverage many of the existing enabling technologies
- The big number of players speeds up the developments
- Record number of qubits so far, good initial scaling potential

CHALLENGES

- Relatively sensitive to errors, short time to compute
- Individual qubits differ and are therefore more complex to control
- Only nearest-neighbor connections, therefore many components needed
- Ultra low temperatures needed

Source: [Quantum Computing Hardware - An Introduction (youtube.com)]

Source: [Quantum Computing Hardware - An Introduction (youtube.com)]

QUBIT HARDWARE: ATOMS

Example 2: Ion Trap Qubits

Trapped –ion qubits: Quantinuum and IonQ create qubits using ionized ytterbium atoms

Ions are atoms where one electron is removed so that they have a positive charge. Because of the charge, they can be trapped and controlled by a magnetic or electric field and used as qubits

IONQ AQT Honeywell Universal Quantum

ADVANTAGES

- Very uniform and stable qubits, decreased complexity
- Long coherence time → long time to compute
- Each qubits can be connected to any other, therefore the computation is fast
- Ions are easy to entangle with the help of light

CHALLENGES

- Not clear how it scales beyond 50 qubits
- Ions require ultra vacuum and cooling so quite a lot of infrastructure
- Gates/ operations are relatively slow which is problematic for very complex algorithms

QUBIT HARDWARE: PHOTONS

Example: Single photons

Source: [Quantum Computing Hardware - An Introduction (youtube.com)]

Xanadu creates qubits by squeezing laser light using ring resonators

Photonics is widely used for the control and readout of qubits. However, the photons themselves can also be controlled, measured and entangled and thus can be used as qubits.

Ψ PsiQuantum XANADU ORCA Computing

ADVANTAGES

- Processors work at room temperature and don't need complex infrastructure
- Mostly based on existing optical components which also enables integration into the classical computing infrastructure.

CHALLENGES

- Probabilistic character of the photons and the so far limited quality of the single photon sources lead to architectures with a lot of redundancy = less scalable
- Photons cannot be 'stopped' or stored for a long term, limits the number of operation and coherence time

QC Hardware : Basic Requirements

1. **A scalable physical system with well characterized qubits**
   - Scalable in terms of material, infrastructure and architecture
2. **The ability to initialize the state of the qubit in a simple fiducial state**
   - Neutral starting position that doesn't influence the operations
3. **Long relevant coherence time**
   - Enough time to compute before significant errors arise
4. **A "universal" set of quantum gates**
   - A universal set of operations that form the basis of computing
   - Gates can also be circumvented, this for example happens in quantum annealers
5. **A qubit-specific measurement capability**
   - A clearly defined way to measure to obtain the final answer

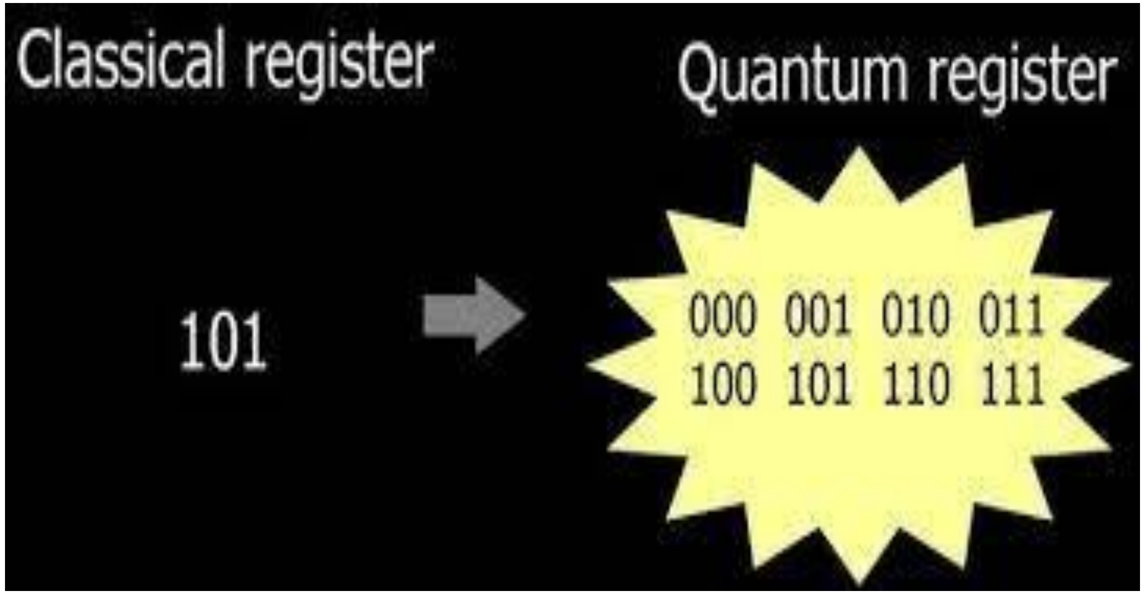
Year	Qubit Size	Milestone
1998	9	First demonstration of quantum error correction using 9 physical qubits to encode 1 logical qubit.
2016	5	IBM introduces the 5-qubit IBM Q 5 Tenerife and IBM Q 5 Yorktown processors.
2017	14	IBM launches the 14-qubit IBM Q 14 Melbourne processor.
	16	IBM introduces the 16-qubit IBM Q 16 Rüschlikon processor.
	17	IBM unveils the 17-qubit IBM Q 17 processor.
	20	IBM releases the 20-qubit IBM Q 20 Tokyo processor.
2018	20	IBM releases the 20-qubit IBM Q 20 Austin processor.
	50	IBM introduces the 50-qubit IBM Q 50 Prototype.
2019	53	IBM launches the 53-qubit IBM Q 53 processor.
	53	Google claims quantum supremacy with its 53-qubit Sycamore processor.
2020	27	IBM achieves a Quantum Volume of 64 with a 27-qubit processor.
2021	127	IBM releases the 127-qubit IBM Quantum Eagle processor.
2022	433	IBM unveils the 433-qubit IBM Quantum Osprey processor.
2023	1,121	IBM presents the 1,121-qubit IBM Quantum Condor processor.
	1,305	Researchers at TU Darmstadt demonstrate a 1,305-qubit array based on optical tweezers.
	1,180	Atom Computing announces a 1,180-qubit array based on Rydberg atoms.
2024	Up to 8	Researchers fuse small quantum states into states with up to eight qubits.

Challenges and Considerations

- **Decoherence**: Quantum systems are sensitive to their environment, and maintaining coherence is critical for reliable computations.
- **Error Correction**: Quantum computations are prone to errors, necessitating the development of quantum error correction codes and fault-tolerant techniques.
- **Scalability**: As quantum computers grow in size and complexity, challenges related to maintaining qubit coherence and reducing noise become increasingly significant.

2. Quantum Register

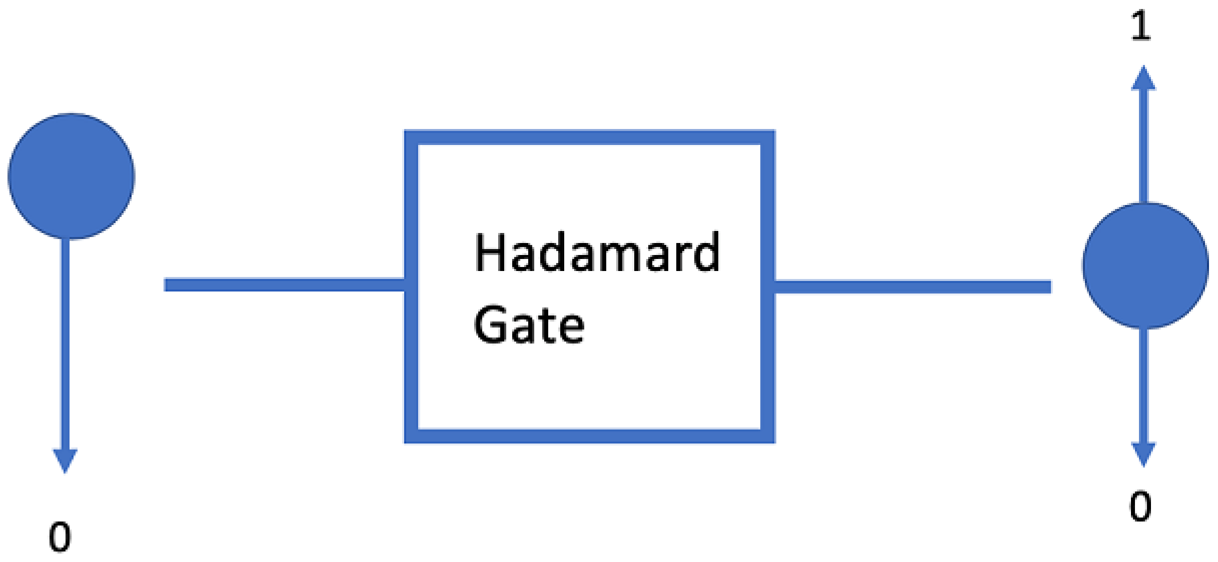
- A quantum register is a system comprising multiple qubits, serving as the quantum analogue of the classical processor register.



Source: Google Images

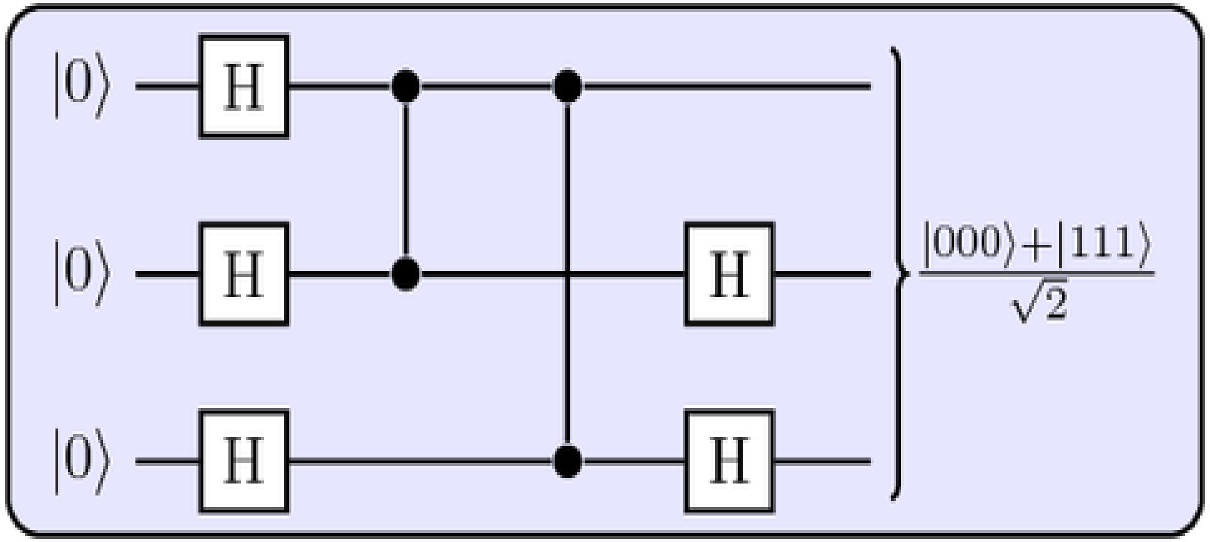
3. Quantum Gates

- Quantum gates is a mathematical operation that acts on the state of one or more qubits, and it can be represented by a matrix.



4. Quantum Circuits

- Quantum circuits are composed of quantum gates and are used to perform quantum algorithms.
- A quantum circuit is a series of **quantum gates** that act on one or more qubits.
- The gates are arranged in a specific order, and the circuit is executed in a specific sequence.



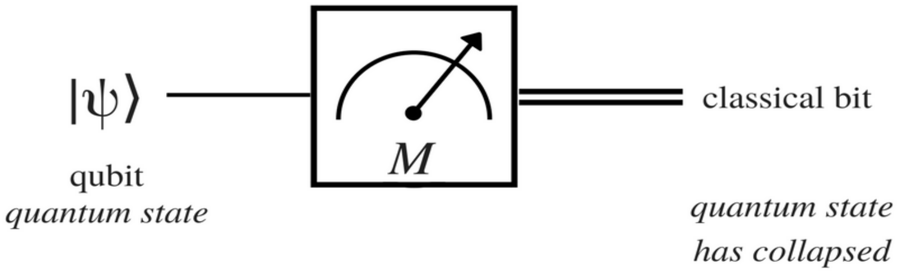
5. Quantum Processing Unit

A quantum processing unit (QPU) is a computational unit that relies on quantum principles to perform a task. The QPU includes the:

- **QRAM (register + gates)**
- **Quantum control unit (QCU) which drives the system to the desired state.**
- **Classical controller interface** which defines the interaction between the host CPU and the QPU

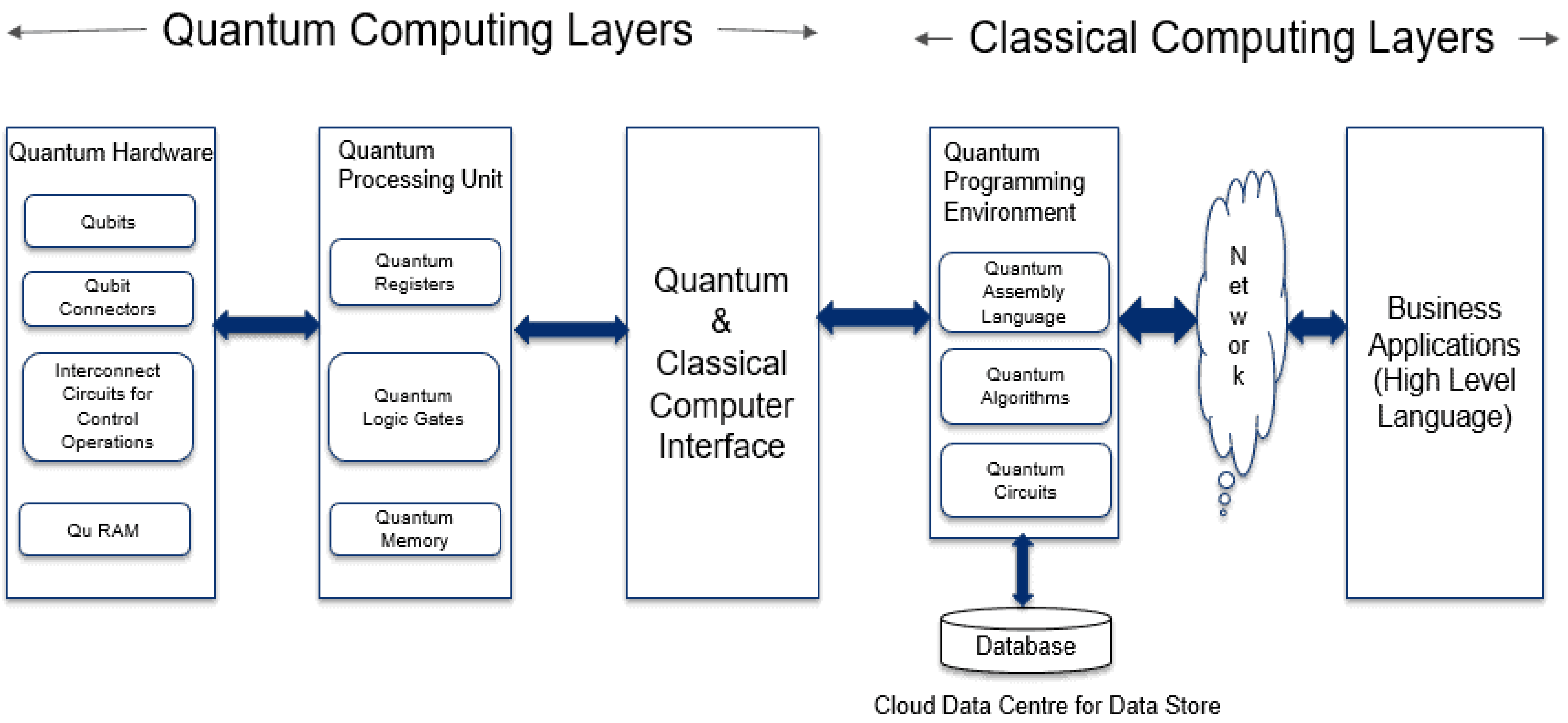
6. Measurement Devices

- Quantum measurement is all about obtaining information about the state of a quantum system
- Quantum Sensors, Quantum Microscopes, Quantum Clocks and Frequency Standards, Quantum Gravimeters and Gyroscopes.



3.3 Architecture of Quantum Computer

1. Application Layer 2. Classical Processing Layer 3.   Quantum Computing Layers



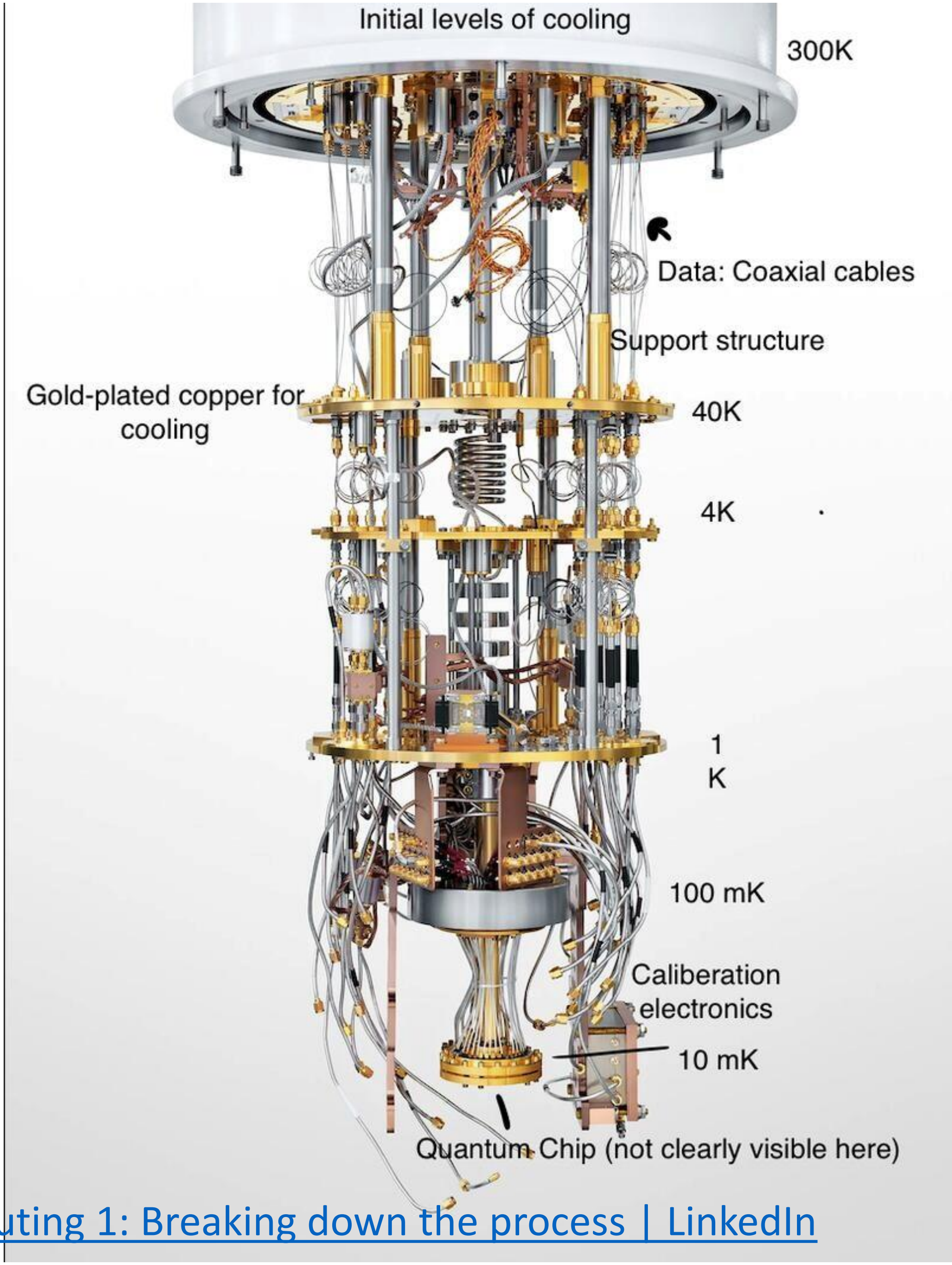
[Source: Overview of Quantum Computer Platform (analyticsinsight.net)]

Quantum Computer Cooling Systems

- Quantum bits, or qubits, typically operate at extremely low temperatures to maintain their quantum states and minimize thermal noise.
- The operational temperature for most superconducting qubits is around **10 to 20 milliKelvin (mK),** which is just above absolute zero (approximately -273.15°C)
- There are advancements in quantum technology that allow for operation at higher temperatures. For instance, silicon qubits have been shown to function at temperatures up to 10 Kelvin (K).

Quantum Computer Cooling Systems

Cooling System	Minimum Temperature Achievable	Type of Qubits
Dilution Refrigerator	~10 mK (10 milliKelvin)	Superconducting qubits
Adiabatic Demagnetization Refrigeration (ADR)	< 2 K (2000 milliKelvin)	Various types of qubits
Pulse Tube Refrigeration (PTR)	< 4 K (4000 milliKelvin)	Superconducting qubits
Laser Cooling	Near absolute zero	Trapped ions, atomic qubits
2D Quantum Cooling System	100 mK (100 milliKelvin)	Various types of qubits
Immersion Cooling with Helium-3	< 1 K (1000 milliKelvin)	Superconducting qubits



Source: [(4) Quantum Computing 1: Breaking down the process | LinkedIn]

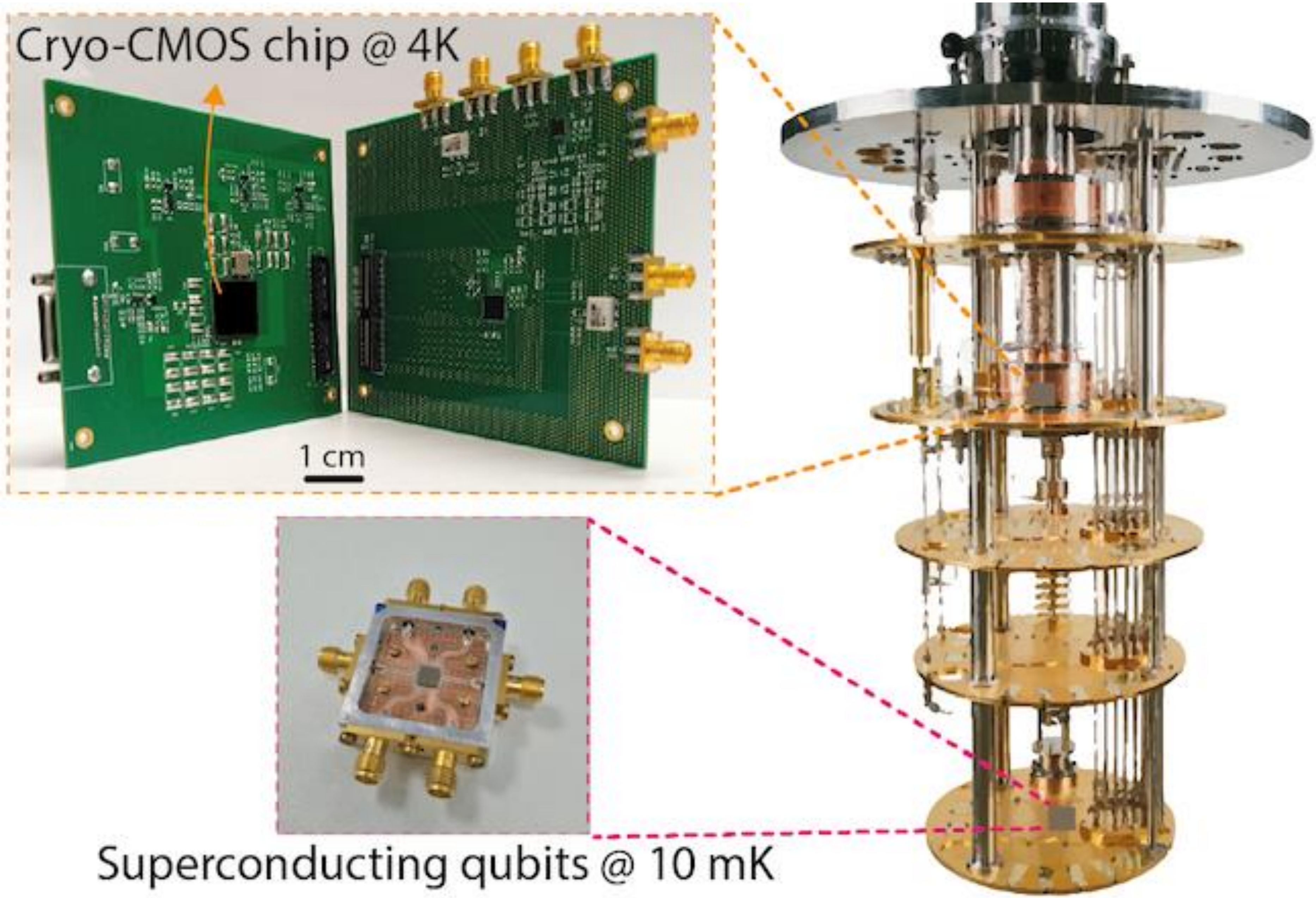
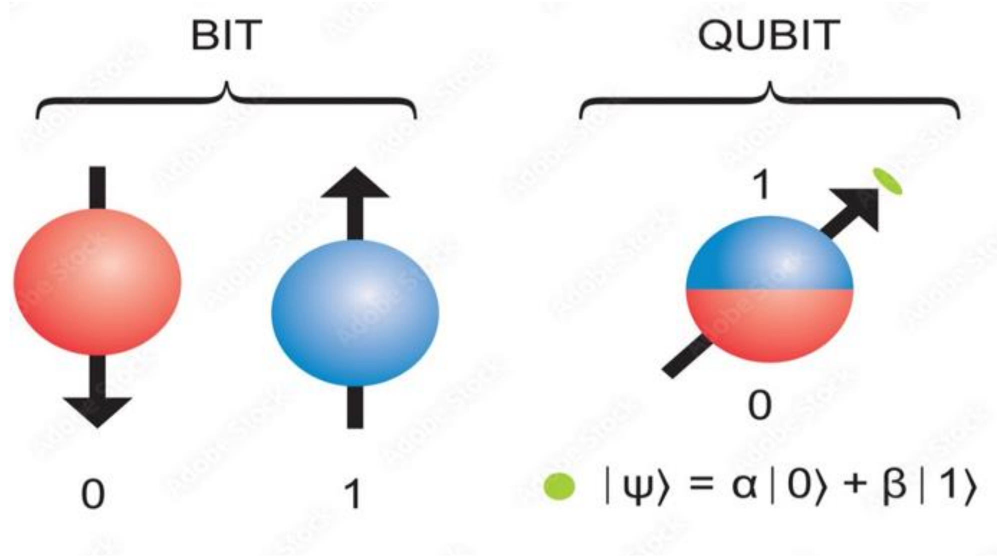


Image Source: [University of Glasgow - University news - Archive of news - 2021 - November - UofG lends support to £6.5m quantum computing consortium]()

Quiz 1

2. Qubit: Topics

A qubit, or quantum bit, is the fundamental unit of information in quantum computing, analogous to a classical bit in traditional computing.



1. Qubits and States
2. Representation of Qubits :
   - State Vectors
   - Dirac Notation : Basic Elements
   - Representing General Qubit States
   - Common Qubit states
   - Measurement in Qubit Notation
3. Inner Product and Outer Product
4. Perpendicular and Parallel Qubit Vectors
5. Magnitude and Normalization of the Qubit Vector
6. Angle between two Qubit vector using Dot Product
7. Linear Combination of two qubit vectors
8. Superposition of Qubits
9. Hilbert Space
10. Basis
11. Tensor Products
12. Entanglement of Qubits in Hilbert Space
13. Bell State
14. Complex numbers in Polar Form
15. Representing Qubits states in Bloch Sphere

N-qubit	Number of states	States	Examples
1	2^1 = 2	|0⟩ and |1⟩	A single qubit can be used as a highly sensitive quantum sensor to measure magnetic fields, electric fields, temperature, pressure and other quantities with extremely high precision
2	2^2 = 4	00⟩, |01⟩, |10⟩, and |11⟩	Used to create entangled states, such as the Bell state 1/sqrt(2)*(|00⟩+|11⟩).
3	2^3 = 8	|000⟩, |001⟩, |010⟩, |011⟩, |100⟩, |101⟩, |110⟩, and |111⟩	A 3-qubit quantum computer can be used to simulate the behavior of a simple molecule like hydrogen (H2)
4	2^4 = 16	|0000⟩, |0001⟩, |0010⟩, |0011⟩, |0100⟩, |0101⟩, |0110⟩, |0111⟩, |1000⟩, |1001⟩, |1010⟩, |1011⟩, |1100⟩, |1101⟩, |1110⟩, and |1111⟩.	A 4-qubit computer can be used to implement Grover's algorithm, which searches an unsorted database
8	2^8 = 256		An 8-qubit quantum computer can be used to factor large numbers using **Shor's** algorithm,
30	2^30 = 1 billion		Could be used to simulate the behavior of complex molecules and materials, which is crucial for developing new drugs, batteries, and other technologies

Classical 2-Bit Example:

- **In classical computing, two bits can represent four distinct states**:
  - 00, 01, 10, and 11 whose binary values are 0,1,2 and 3.
  - Each bit is either 0 or 1.
- Each state represents a unique combination of the two bits, corresponding to a specific decimal value from 0 to 3.
- The system can only be in **one of these states at a time**.

Quantum 2-Qubit System:

- In quantum computing, two qubits can represent a superposition of all four states (00, 01, 10, and 11) simultaneously.
- The state of the qubits is described as a linear combination of these basis states.

A general 2-qubit state can be written as:

$$\text{State} = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

where α_{00}, α_{01}, α_{10}, and α_{11} are complex numbers representing the probability amplitudes for each state.

2.2 Representation of Qubits

State Vector : Each element in the state vector represents probability of being in that particular state.

State vector = [H T]

If the coin is in the head's state , state vector = [1 0]

If the coin is in the tail's state , state vector = [0 1]

State vector is used to represent the state of quantum systems

Dirac Notation

- Dirac notation, also known as **bra-ket notation**, is a standard way to represent quantum states and operations in quantum mechanics. It is particularly useful in describing **qubits** and **quantum** systems.
- **Ket |ψ⟩ : Column Vector**
- **Bra ⟨ψ| : Row Vector**
- **Bra-Ket** : **⟨ψ| ψ⟩ Inner Product**
- **Ket- Bra: |ψ⟩ ⟨ψ| Outer Product**

Basic Elements of Dirac Notation

Ket |ψ⟩:

- Represents a column vector (**a quantum state**).

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

Bra ⟨ψ|: Represents the conjugate transpose (row vector) of the **ket.**

$$\langle\psi| = \begin{pmatrix} \alpha^* & \beta^* \end{pmatrix}$$

where $\alpha*$ and $\beta*$ are the complex conjugates of α and β.

Qubit States

The general state of a qubit can be written as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are complex numbers. These numbers describe the probability amplitudes of the qubit being in the states $|0\rangle$ and $|1\rangle$, respectively.

The probabilities themselves are given by the squares of the magnitudes of these amplitudes: $|\alpha|^2$ for $|0\rangle$ and $|\beta|^2$ for $|1\rangle$, with the condition that $|\alpha|^2 + |\beta|^2 = 1$.

Common Qubit States

1. **Standard Basis States:**
   - $|0\rangle$: The qubit is definitely in the "0" state.
   - $|1\rangle$: The qubit is definitely in the "1" state.
2. **Superposition States:**
   - $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$: This is an equal superposition of $|0\rangle$ and $|1\rangle$.
   - $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$: This is another type of superposition, with a phase difference.

Measurement in Qubit Notation

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

- When you measure a qubit, you collapse its state to either $|0\rangle$ or $|1\rangle$.
  - The probability of measuring $|0\rangle$ is $|\alpha|^2$.
  - The probability of measuring $|1\rangle$ is $|\beta|^2$.

Example

If a qubit is in the state $|\psi\rangle = \frac{1}{\sqrt{3}}|0\rangle + \sqrt{\frac{2}{3}}|1\rangle$, it means:

- There is a $\frac{1}{3}$ chance of measuring it as $|0\rangle$.
- There is a $\frac{2}{3}$ chance of measuring it as $|1\rangle$.

2.3 Inner Product and Outer Product

Bra-Ket: Inner Product (⟨ψ|ϕ⟩)

The inner product is the product of a bra and a ket, which results in a scalar (a complex number). This scalar represents the overlap or similarity between two quantum states.

For two qubit states $|\psi\rangle$ and $|\phi\rangle$, the inner product is given by:

$$\langle\psi|\phi\rangle = \alpha^*\gamma + \beta^*\delta$$

where:

- $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$
- $|\phi\rangle = \gamma|0\rangle + \delta|1\rangle$
- α, β, γ, and δ are complex numbers, and α^* and β^* are the complex conjugates of α and β.

Ket-Bra: Outer Product (|ψ⟩⟨ϕ|)

The outer product is the product of a ket and a bra, resulting in a matrix (also called an operator). This matrix can be used to describe transformations in quantum mechanics.

For the same states $|\psi\rangle$ and $|\phi\rangle$, the outer product is written as:

$$|\psi\rangle\langle\phi| = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \begin{pmatrix} \gamma^* & \delta^* \end{pmatrix} = \begin{pmatrix} \alpha\gamma^* & \alpha\delta^* \\ \beta\gamma^* & \beta\delta^* \end{pmatrix}$$

This matrix describes how the state $|\phi\rangle$ can be transformed by the state $|\psi\rangle$.

Inner Product Example

Consider a qubit in the following state:

$$|\psi\rangle = \frac{3}{5}|0\rangle + \frac{4}{5}|1\rangle$$

In ket notation:

$$|\psi\rangle = \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \end{pmatrix}$$

The bra for this state would be:

$$\langle\psi| = \begin{pmatrix} \frac{3}{5} & \frac{4}{5} \end{pmatrix}$$

Inner Product Example

Let's say we have another qubit state $|\phi\rangle$ given by:

$$|\phi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

In ket notation:

$$|\phi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

The bra for this state would be:

$$\langle\phi| = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

Inner Product Example

Now, let's calculate the **inner product** $\langle \phi | \psi \rangle$, which gives the overlap between the two states:

$$\langle \phi | \psi \rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \end{pmatrix}$$

Perform the multiplication:

$$\langle \phi | \psi \rangle = \frac{1}{\sqrt{2}} \times \frac{3}{5} + \frac{1}{\sqrt{2}} \times \frac{4}{5} = \frac{3}{5\sqrt{2}} + \frac{4}{5\sqrt{2}} = \frac{7}{5\sqrt{2}}$$

Outer Product Example

The **outer product** $|\psi\rangle\langle\phi|$ results in a matrix (operator):

$$|\psi\rangle\langle\phi| = \begin{pmatrix} \frac{3}{5} \\ \frac{4}{5} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

Perform the matrix multiplication:

$$|\psi\rangle\langle\phi| = \begin{pmatrix} \frac{3}{5\sqrt{2}} & \frac{3}{5\sqrt{2}} \\ \frac{4}{5\sqrt{2}} & \frac{4}{5\sqrt{2}} \end{pmatrix}$$

2.4 Perpendicular and Parallel Qubit Vectors

Perpendicular Qubit Vectors

Two qubit vectors are perpendicular if their inner product (dot product) is zero. For example:

- **Qubit Vector 1:** $|\phi_1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
- **Qubit Vector 2:** $|\phi_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

$$\langle\phi_1|\phi_2\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = (1 \cdot 0) + (0 \cdot 1) = 0$$

Since the inner product is zero, $|\phi_1\rangle$ and $|\phi_2\rangle$ are perpendicular.

Two Parallel Qubit Vectors

Two vectors are parallel if one is a scalar multiple of the other. Let's check if $|\psi_2\rangle$ is a scalar multiple of $|\psi_1\rangle$.

Example:

- **Qubit Vector 1:** $|\psi_1\rangle = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$
- **Qubit Vector 2:** $|\psi_2\rangle = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$

We can express $|\psi_2\rangle$ as:

$$|\psi_2\rangle = 2 \cdot |\psi_1\rangle = 2 \cdot \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \end{pmatrix}$$

Since $|\psi_2\rangle$ is exactly 2 times $|\psi_1\rangle$, the vectors $|\psi_1\rangle$ and $|\psi_2\rangle$ are parallel.

2.5 Magnitude and Normalization of the Qubit Vector

Magnitude of the Qubit Vector

$$|\psi\rangle = \begin{pmatrix} a \\ b \end{pmatrix}$$

The magnitude (or norm) of $|\psi\rangle$ is calculated as:

$$||\psi|| = \sqrt{|a|^2 + |b|^2}$$

where $|a|$ and $|b|$ represent the absolute values (or moduli) of the complex numbers a and b, respectively.

Example:

Let's say the qubit vector is:

$$|\psi\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

The magnitude would be:

$$||\psi|| = \sqrt{\left|\frac{1}{\sqrt{2}}\right|^2 + \left|\frac{1}{\sqrt{2}}\right|^2} = \sqrt{\frac{1}{2} + \frac{1}{2}} = \sqrt{1} = 1$$

Normalization Process

To normalize $|\psi\rangle$, you divide each component by the norm of the vector:

$$|\psi_{\text{normalized}}\rangle = \frac{1}{||\psi||}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \frac{a}{||\psi||} \\ \frac{b}{||\psi||} \end{pmatrix}$$

Example

Let's consider a qubit vector:

$$|\psi\rangle = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

Step 1: Calculate the norm of $|\psi\rangle$:

$$||\psi|| = \sqrt{|3|^2 + |4|^2} = \sqrt{9 + 16} = \sqrt{25} = 5$$

Step 2: Normalize the vector:

To normalize $|\psi\rangle$, divide each component by the norm:

$$|\psi_{\text{normalized}}\rangle = \frac{1}{5}\begin{pmatrix}3\\4\end{pmatrix} = \begin{pmatrix}\frac{3}{5}\\\frac{4}{5}\end{pmatrix}$$

So the normalized qubit vector is:

$$|\psi_{\text{normalized}}\rangle = \begin{pmatrix}0.6\\0.8\end{pmatrix}$$

Step 3: Verify the normalization:

Let's check the magnitude of the normalized vector:

$$\|\psi_{\text{normalized}}\| = \sqrt{(0.6)^2 + (0.8)^2} = \sqrt{0.36 + 0.64} = \sqrt{1} = 1$$

Since the magnitude is 1, the vector is properly normalized.

Normalization ensures that the qubit vector has a magnitude of 1, making it consistent with the principles of quantum mechanics. In this example, the vector $\begin{pmatrix}3\\4\end{pmatrix}$ was normalized to $\begin{pmatrix}0.6\\0.8\end{pmatrix}$.

2.6 Angle between two Qubit vector using Dot Product

Finding the Angle Between Two Qubit Vectors Using the Dot Product

In quantum mechanics, the angle between two qubit vectors can be found using the inner product (dot product) of the vectors. This angle is related to the overlap of the states, and it can be calculated using the following formula:

$$\cos(\theta) = \frac{|\langle\psi_1|\psi_2\rangle|}{||\psi_1|| \, ||\psi_2||}$$

Where:

- θ is the angle between the two qubit vectors.
- $\langle\psi_1|\psi_2\rangle$ is the inner product of the two vectors.
- $||\psi_1||$ and $||\psi_2||$ are the magnitudes (norms) of the vectors $|\psi_1\rangle$ and $|\psi_2\rangle$.

Example

Let's consider two qubit vectors:

$$|\psi_1\rangle = \begin{pmatrix} 1+i \\ 0 \end{pmatrix}, \quad |\psi_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Step 1: Compute the Inner Product

$$\langle\psi_1|\psi_2\rangle = (1+i)\cdot 0 + 0\cdot 1 = 0$$

The inner product is 0.

Step 2: Find the Magnitudes

$$||\psi_1|| = \sqrt{|1+i|^2 + 0^2} = \sqrt{(1^2+1^2)} = \sqrt{2}$$

$$||\psi_2|| = \sqrt{0^2 + |1|^2} = \sqrt{1} = 1$$

Step 3: Calculate the Cosine of the Angle

Since the inner product is 0, $\cos(\theta)$ will also be 0:

$$\cos(\theta) = \frac{0}{\sqrt{2} \cdot 1} = 0$$

Step 4: Find the Angle

$$\theta = \arccos(0) = \frac{\pi}{2} \text{ radians} = 90^\circ$$

The angle θ between the two qubit vectors $|\psi_1\rangle = \begin{pmatrix} 1+i \\ 0 \end{pmatrix}$ and $|\psi_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ is 90°, meaning they are orthogonal (perpendicular) to each other.

2.7 Linear Combination of Two Qubit Vectors

A **linear combination** of two qubit vectors involves creating a new qubit vector by adding the vectors together, each multiplied by a scalar (which can be a complex number).

Given two qubit vectors $|\psi_1\rangle$ and $|\psi_2\rangle$, a linear combination of these vectors can be expressed as:

$$|\psi\rangle = c_1|\psi_1\rangle + c_2|\psi_2\rangle$$

where c_1 and c_2 are complex numbers (scalars).

Example

Consider two qubit vectors:

$$|\psi_1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |\psi_2\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

These are the basis states $|0\rangle$ and $|1\rangle$, respectively.

Let's form a linear combination:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|\psi_1\rangle + \frac{1}{\sqrt{2}}|\psi_2\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

2.7 Superposition of Qubits

In general, the superposition state can have any complex coefficients c_1 and c_2:

$$|\psi\rangle = c_1|0\rangle + c_2|1\rangle = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

The state is normalized if:

$$|c_1|^2 + |c_2|^2 = 1$$

1. Equal Superposition

This is a state with equal probabilities of measuring $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

Probabilities:

- $P(0) = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2} = 50\%$
- $P(1) = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2} = 50\%$

2. Biased Towards |0⟩

In this state, the probability of measuring $|0\rangle$ is higher than that of measuring $|1\rangle$:

$$|\psi\rangle = \frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$$

Probabilities:

- $P(0) = \left|\frac{\sqrt{3}}{2}\right|^2 = \frac{3}{4} = 75\%$
- $P(1) = \left|\frac{1}{2}\right|^2 = \frac{1}{4} = 25\%$

3. Biased Towards |1⟩

Here, the probability of measuring $|1\rangle$ is higher:

$$|\psi\rangle = \frac{1}{3}|0\rangle + \frac{2\sqrt{2}}{3}|1\rangle$$

Probabilities:

- $P(0) = \left|\frac{1}{3}\right|^2 = \frac{1}{9} \approx 11.11\%$
- $P(1) = \left|\frac{2\sqrt{2}}{3}\right|^2 = \frac{8}{9} \approx 88.89\%$

4. Closer to |0⟩

This state has a higher probability of being in $|0\rangle$ but still a significant chance of being in $|1\rangle$:

$$|\psi\rangle = \frac{2}{\sqrt{5}}|0\rangle + \frac{1}{\sqrt{5}}|1\rangle$$

Probabilities:

- $P(0) = \left|\frac{2}{\sqrt{5}}\right|^2 = \frac{4}{5} = 80\%$
- $P(1) = \left|\frac{1}{\sqrt{5}}\right|^2 = \frac{1}{5} = 20\%$

5. Very Close to |1⟩

This state has a high probability of being in $|1\rangle$:

$$|\psi\rangle = \frac{1}{4}|0\rangle + \frac{\sqrt{15}}{4}|1\rangle$$

Probabilities:

- $P(0) = \left|\frac{1}{4}\right|^2 = \frac{1}{16} = 6.25\%$
- $P(1) = \left|\frac{\sqrt{15}}{4}\right|^2 = \frac{15}{16} = 93.75\%$

Note

- **Ket notation $|\psi\rangle$**: Represents the state as a **column vector**.
- **Bra notation $\langle\psi|$**: Represents the conjugate transpose of the ket as a **row vector.**
- **Inner product $\langle\phi|\psi\rangle$**: Gives a **scalar**, indicating the **overlap** between two quantum states.
- **Outer product $|\psi\rangle\langle\phi|$**: Results in a **matrix**, useful for constructing quantum operators.

2.9 Hilbert Space in Quantum Computing

- A **Hilbert space** in quantum computing is a mathematical framework used to describe the state space of quantum systems. It is a complete inner product space where:
  - **Vectors** represent quantum states.
  - **Inner product** defines the overlap or similarity between states.
  - **Norm** of a vector represents the probability amplitude of finding the system in that state.
  - **Unitary Operators** represent quantum gates.
  - **Projection Operators** represent measurements.
  - **Probabilities** are calculated based on the norms and inner products.

Hilbert Space in Quantum Computing

In quantum computing, the Hilbert space $\mathbb{C}^2$ for a single qubit includes:

- **Basis Vectors:** $|0\rangle$ and $|1\rangle$, represented as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ respectively.
- **State Vectors:** Any qubit state can be expressed as a superposition of the basis vectors.
- **Unitary Operators:** Transform the state vectors; for example, the Hadamard gate.
- **Measurement:** Projects the state vector onto the basis vectors and gives probabilities for measurement outcomes.

2.10 Basis in Quantum Mechanics

- In **quantum mechanics**, a basis typically refers to a set of orthonormal vectors in a **Hilbert space**.

- For qubits, the basis vectors are often represented as **$|0\rangle$ and $|1\rangle$,** which are the standard basis vectors for a single qubit

|0> and |1> are ortho normal basis

1. Orthogonality

To check orthogonality, we calculate the inner product (dot product) of $|0\rangle$ and $|1\rangle$:

$$\langle 0|1\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = (1 \times 0) + (0 \times 1) = 0$$

Since $\langle 0|1\rangle = 0$, the vectors $|0\rangle$ and $|1\rangle$ are orthogonal.

|0> and |1> are ortho normal basis

2. Normalization

To check normalization, we calculate the norm of each vector:

- For $|0\rangle$:

$$\| \, |0\rangle \, \| = \sqrt{\langle 0|0\rangle} = \sqrt{\begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}} = \sqrt{(1 \times 1) + (0 \times 0)} = \sqrt{1} = 1$$

- For $|1\rangle$:

$$\| \, |1\rangle \, \| = \sqrt{\langle 1|1\rangle} = \sqrt{\begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix}} = \sqrt{(0 \times 0) + (1 \times 1)} = \sqrt{1} = 1$$

Both $|0\rangle$ and $|1\rangle$ are normalized since their norms are equal to 1.

2.11 Tensor Products

- If we have two vector spaces **V** and **W**, their tensor product of **V and W** is a new vector space formed from all possible combinations of vectors from **V and W**.
- The **dimension** of the tensor product space is the product of the dimensions of the individual spaces.
- For example, if **V** has dimension **m** and **W** has dimension **n**, then tensor product of **V and W** has dimension **m×n**.

Tensor Product Notation

The tensor product of two vectors $|\psi_1\rangle$ and $|\psi_2\rangle$ is denoted as:

$$|\psi_1\rangle \otimes |\psi_2\rangle$$

It is often written simply as $|\psi_1\rangle|\psi_2\rangle$ or $|\psi_1\psi_2\rangle$.

Example 1 :

$$X \otimes I = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 0(I) & 1(I) \\ 1(I) & 0(I) \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

Example 2:

$$\begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\begin{bmatrix}1\\0\end{bmatrix}\\0\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

$$= \begin{bmatrix}0\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\\1\begin{bmatrix}1\\0\\0\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\\1\\0\\0\\0\end{bmatrix}$$

Example 3:

Consider two qubits in the states $|\psi_1\rangle$ and $|\psi_2\rangle$:

- Let $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$
- Let $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$

The tensor product $|\psi_1\rangle \otimes |\psi_2\rangle$ is:

$$|\psi_1\rangle \otimes |\psi_2\rangle = (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle)$$

Expanding this:

$$|\psi_1\rangle \otimes |\psi_2\rangle = \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle$$

Example 4:

Let's take specific qubit states:

- $|\psi_1\rangle = |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
- $|\psi_2\rangle = |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

The tensor product is:

$$|0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

This results in:

$$|0\rangle \otimes |1\rangle = \begin{pmatrix} 1 \cdot 0 \\ 1 \cdot 1 \\ 0 \cdot 0 \\ 0 \cdot 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle$$

Example 5:

Now, let's consider two qubits each in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$:

- $|\psi_1\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
- $|\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$

The tensor product gives:

$$|\psi_1\rangle \otimes |\psi_2\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$|\psi_1\rangle \otimes |\psi_2\rangle = \frac{1}{\sqrt{2} \cdot \sqrt{2}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

$$|\psi_1\rangle \otimes |\psi_2\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

2.12 Entanglement of Qubits in Hilbert Space

- **Entanglement** is a quantum phenomenon where two or more qubits become linked in such a way **that the state of one qubit is dependent on the state of the other, no matter the distance between them**.
- This relationship persists even if the qubits are ***separated by large distances***, leading to correlations in their measurements.
- In quantum computing, **entangled states are described** using the **Hilbert space of multiple qubits.**
- **Entanglement** involves quantum states that **cannot be factored** into separate states of individual qubits:

Entanglement of Qubits in Hilbert Space

- **Entanglement** is a quantum phenomenon where two or more qubits become linked in such a way **that the state of one qubit is dependent on the state of the other, no matter the distance between them**.
- **Entanglement** involves quantum states that cannot be factored into separate states of individual qubits.
- **Example :**

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

2.13 What is Bell State?

- The **Bell states** are specific quantum states of two qubits that are **maximally entangled**.
- They are named after physicist **John Bell**, who studied the implications of entanglement for quantum mechanics and classical physics.
- There are **four Bell states**, each representing a different kind of entanglement between the **two qubits**.

1. $|\Phi^+\rangle$:

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

This state represents two qubits that are perfectly correlated: if one qubit is measured in the $|0\rangle$ state, the other will also be in the $|0\rangle$ state, and similarly for the $|1\rangle$ state.

2. $|\Phi^-\rangle$:

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

In this state, the qubits are still correlated, but with a relative phase difference of -1 between the $|00\rangle$ and $|11\rangle$ components.

3. $|\Psi^+\rangle$:

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

This state represents two qubits that are anti-correlated: if one qubit is measured in the $|0\rangle$ state, the other will be in the $|1\rangle$ state, and vice versa.

4. $|\Psi^-\rangle$:

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

Similar to $|\Psi^+\rangle$, this state has anti-correlated qubits, but with a relative phase difference of -1 between the $|01\rangle$ and $|10\rangle$ components.

Prove that The Bell State is entangled state

To prove that the Bell state is an entangled state, we can consider the specific Bell state $|\Phi^+\rangle$, defined as:

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

A quantum state is considered **entangled** if it cannot be expressed as a product of individual states of its components. Specifically, a two-qubit state $|\Psi\rangle$ is separable (not entangled) if it can be written as:

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$$

for some states $|\psi_1\rangle$ and $|\psi_2\rangle$. Conversely, if no such factorization is possible, the state is entangled.

Proof of Entanglement for $|\Phi^+\rangle$

1. **Assume Separability**: Suppose $|\Phi^+\rangle$ is separable. Then there exist states $|\psi_1\rangle$ and $|\psi_2\rangle$ such that:

$$|\Phi^+\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$$

2. **Form of States**: Let $|\psi_1\rangle = a|0\rangle + b|1\rangle$ and $|\psi_2\rangle = c|0\rangle + d|1\rangle$, where a, b, c, d are complex coefficients satisfying normalization conditions.

3. **Tensor Product**: The tensor product of these states gives:

$$|\psi_1\rangle \otimes |\psi_2\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) = ac|00\rangle + ad|01\rangle + bc|10\rangle$$

4. **Equating States**: For $|\Phi^+\rangle$ to equal this tensor product, we must have:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$$

5. **Coefficient Comparison**: This leads to the following equations based on the coefficients of $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$:

    - $ac = \frac{1}{\sqrt{2}}$ (coefficient of $|00\rangle$)
    - $ad = 0$ (coefficient of $|01\rangle$)
    - $bc = 0$ (coefficient of $|10\rangle$)
    - $bd = \frac{1}{\sqrt{2}}$ (coefficient of $|11\rangle$)

6. **Analyzing the Equations:**

- From $ad = 0$ and $bc = 0$, we conclude that either $a = 0$ or $d = 0$ and either $b = 0$ or $c = 0$.
- If $a = 0$, then $|\psi_1\rangle = b|1\rangle$ and $|\psi_2\rangle$ must yield $|11\rangle$, which contradicts $ac = \frac{1}{\sqrt{2}}$.
- If $b = 0$, then $|\psi_1\rangle = a|0\rangle$ and $|\psi_2\rangle$ must yield $|00\rangle$, again leading to a contradiction.

7. **Conclusion:** Since all scenarios lead to contradictions, $|\Phi^+\rangle$ cannot be factored into a product of two states. Therefore, it is an entangled state.

2.14 Complex Numbers in Polar Form

A complex number z can be written as:

$$z = x + iy$$

Where x is the real part and y is the imaginary part of the complex number, and i is the imaginary unit, defined as $i^2 = -1$.

In polar form, this complex number can also be written as:

$$z = |z|e^{i\phi}$$

Where:

- $|z|$ is the **magnitude** (or modulus) of the complex number.
- ϕ is the **phase angle** (or argument) of the complex number.
- $e^{i\phi}$ represents the **direction** or **rotation** in the complex plane.

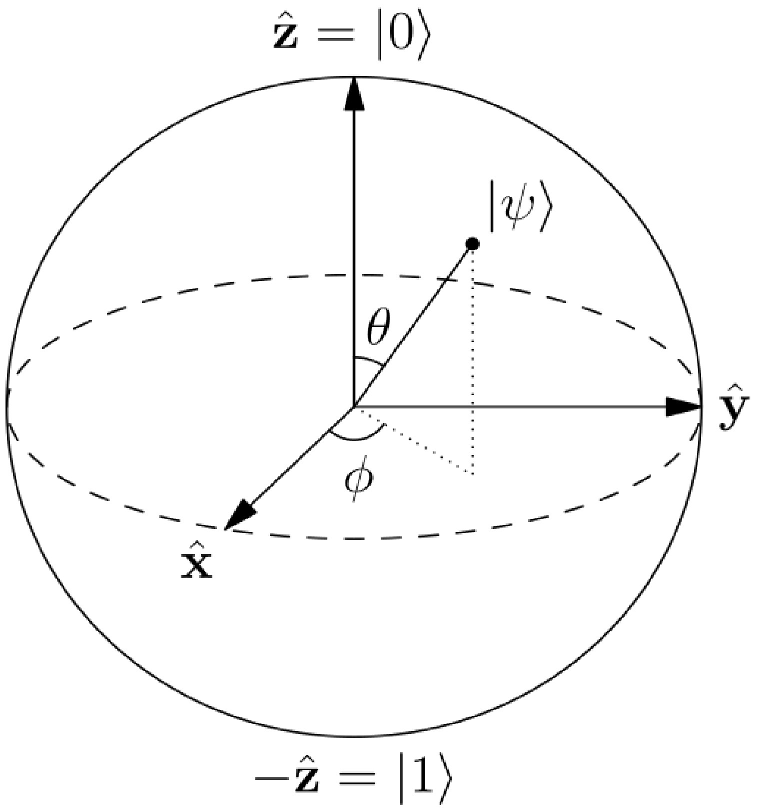
2.15 Representing QuBits states using Bloch Sphere

- The state of a qubit can be represented as a point on the Bloch Sphere
- It is a unit sphere, where r=1.

Qubit State Representation

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Where α and β are complex numbers.



Expressing α and β:

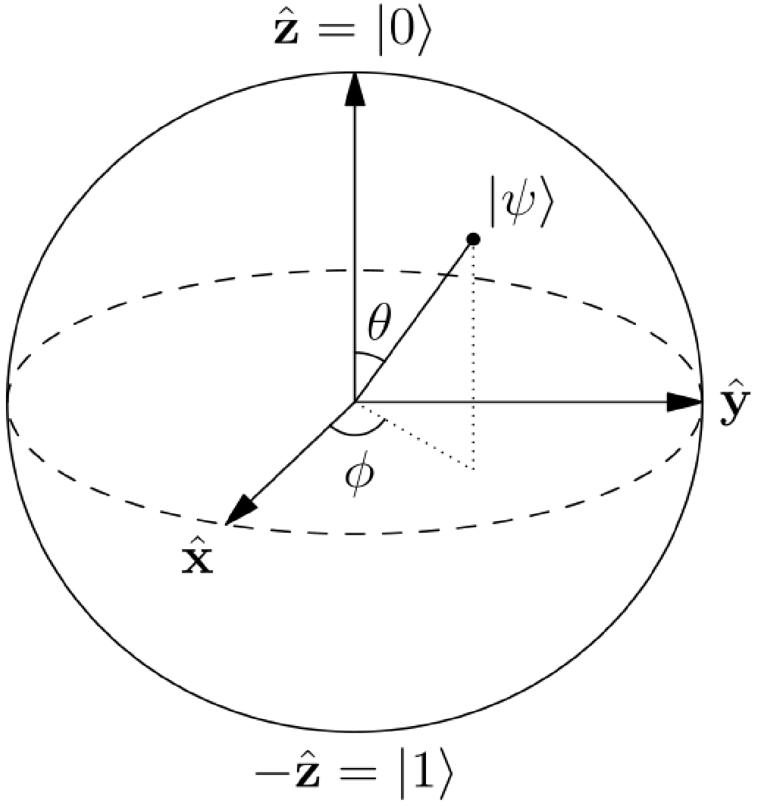
$$\alpha = |\alpha| e^{i\phi_\alpha}, \quad \beta = |\beta| e^{i\phi_\beta}$$

The state can also be written as:

$$|\psi\rangle = |\alpha| e^{i\phi_\alpha} |0\rangle + |\beta| e^{i\phi_\beta} |1\rangle$$

This can be factored as:

$$|\psi\rangle = e^{i\phi_\alpha} \left[|\alpha||0\rangle + |\beta| e^{i(\phi_\beta - \phi_\alpha)} |1\rangle \right]$$



Here, $\phi = \phi_\beta - \phi_\alpha$ is the **relative phase.**

Since the global phase $e^{i\phi_\alpha}$ is insignificant, it can be ignored:

$$|\psi\rangle = |\alpha||0\rangle + |\beta| e^{i\phi} |1\rangle$$

Magnitudes and Trigonometric Representation:

We know:

$$|\alpha|^2 + |\beta|^2 = 1$$

Set:

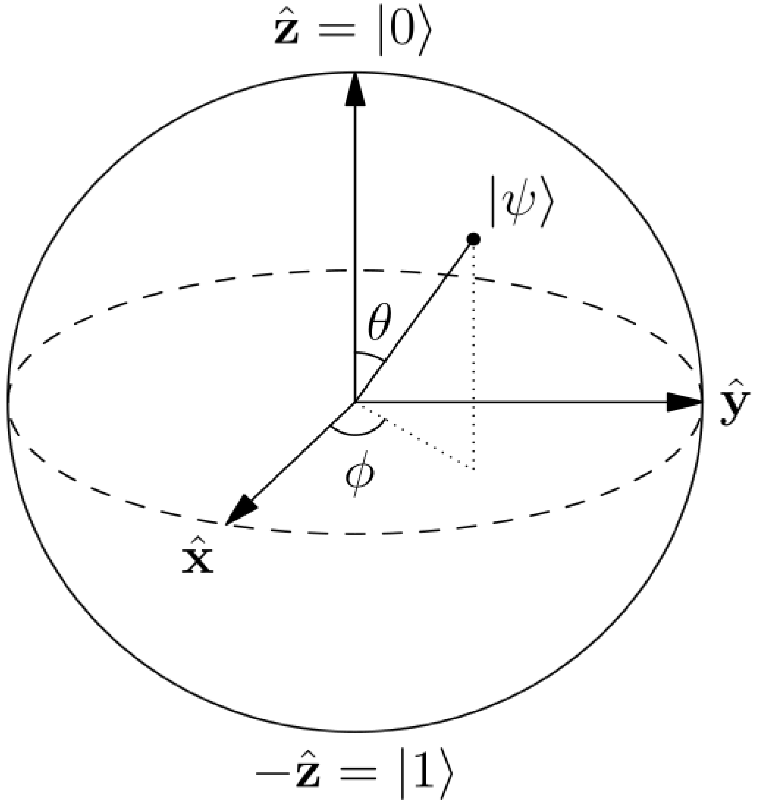
$$|\alpha| = \cos(\theta/2), \quad |\beta| = \sin(\theta/2)$$

Then:

$$\cos^2(\theta/2) + \sin^2(\theta/2) = 1$$

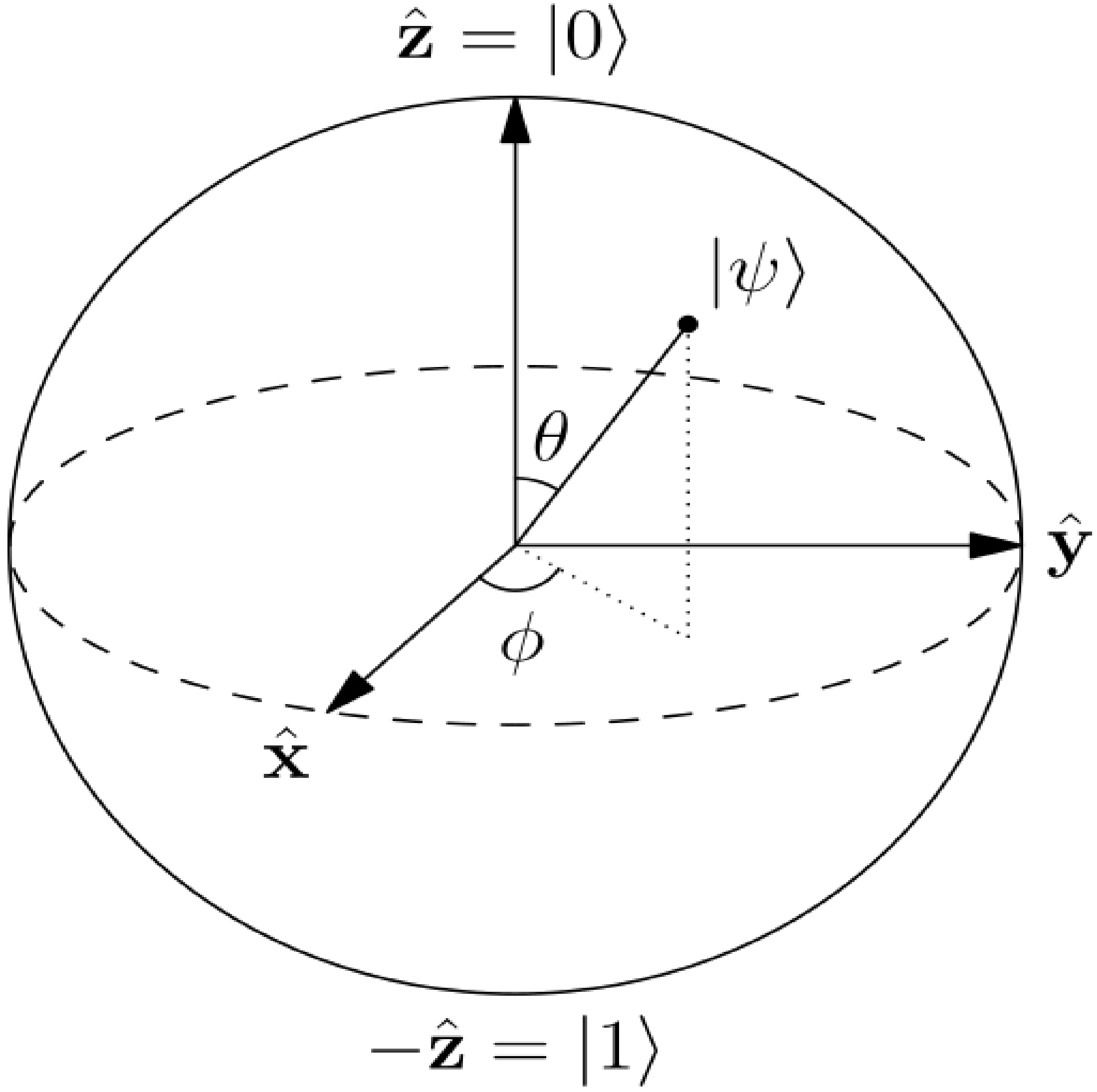
So, the qubit state can be written as:

$$|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)e^{i\phi}|1\rangle$$



Representing QuBits states using Bloch Sphere

- A Bloch sphere uses its three axes to represent a **qubit's** state. The state vector originates in the center of the sphere and terminates at a point with **z, x, and y** coordinates.
- The **z-axis** represents the probability of the qubit being measured as a **0 or a 1**.
- The **x-axis** represents the **real part** of the state vector.
- The **y-axis** represents the **imaginary part** of the state vector.



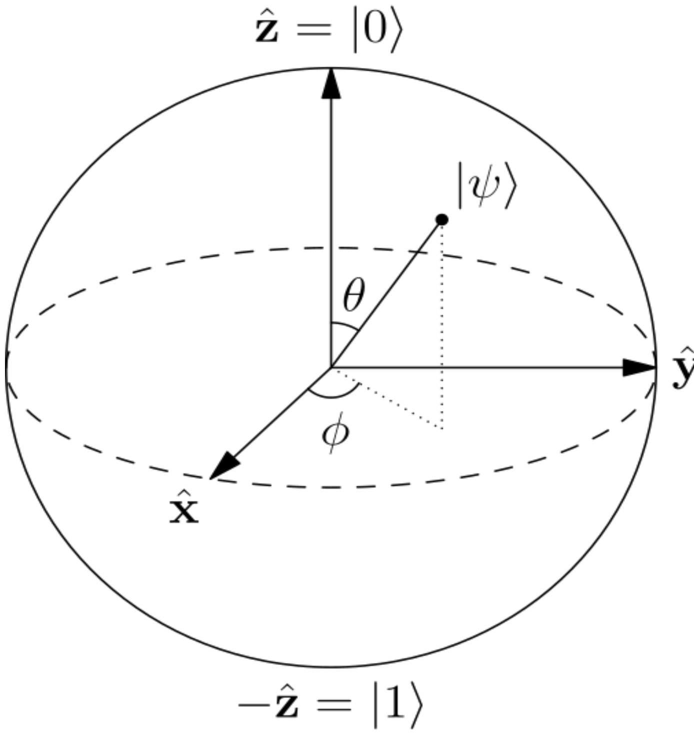
Qubit State in Polar Coordinates

To represent this state in polar coordinates on the Bloch sphere, we express α and β using two angles θ and ϕ:

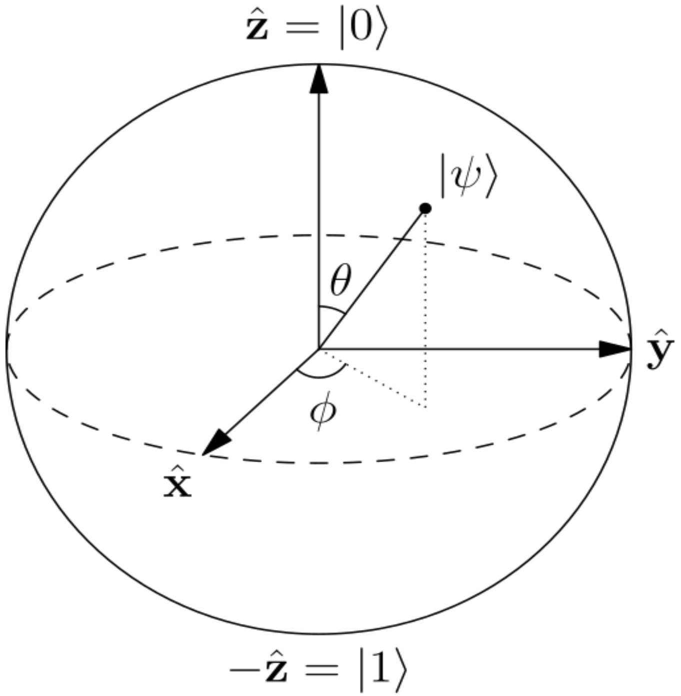
1. θ: This is the polar angle (latitude) on the Bloch sphere, ranging from 0 to π.
2. ϕ: This is the azimuthal angle (longitude) on the Bloch sphere, ranging from 0 to 2π.

Given these angles, the qubit state $|\psi\rangle$ can be written as:

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



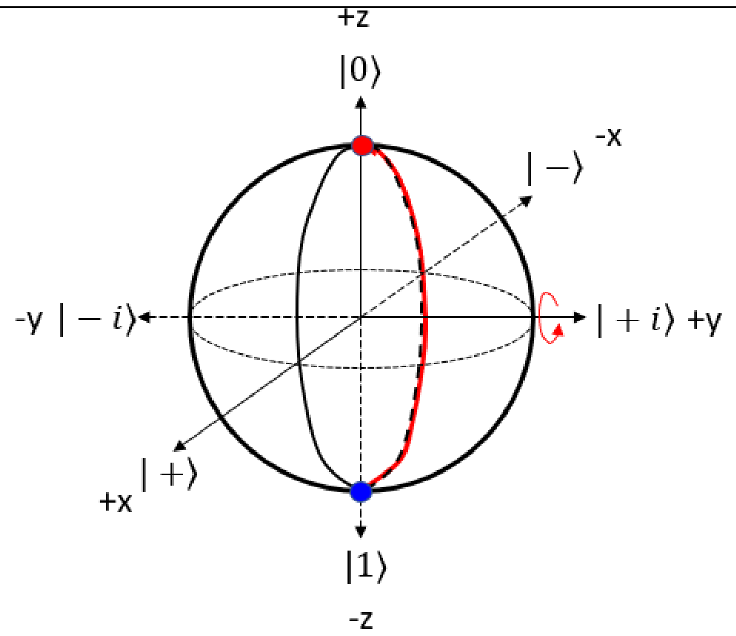
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

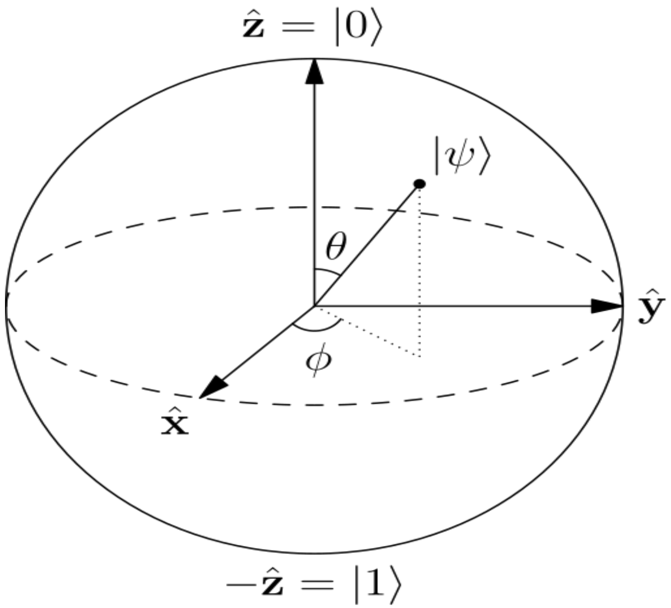


$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle$$

- $\cos\left(\frac{\theta}{2}\right)$ corresponds to the probability amplitude for the qubit being in the $|0\rangle$ state.
- $\sin\left(\frac{\theta}{2}\right)$ corresponds to the probability amplitude for the qubit being in the $|1\rangle$ state.
- $e^{i\phi}$ introduces a phase factor for the $|1\rangle$ component.

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





$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$$

- $\theta = 0$
- ϕ can be any value, but typically we take $\phi = 0$.

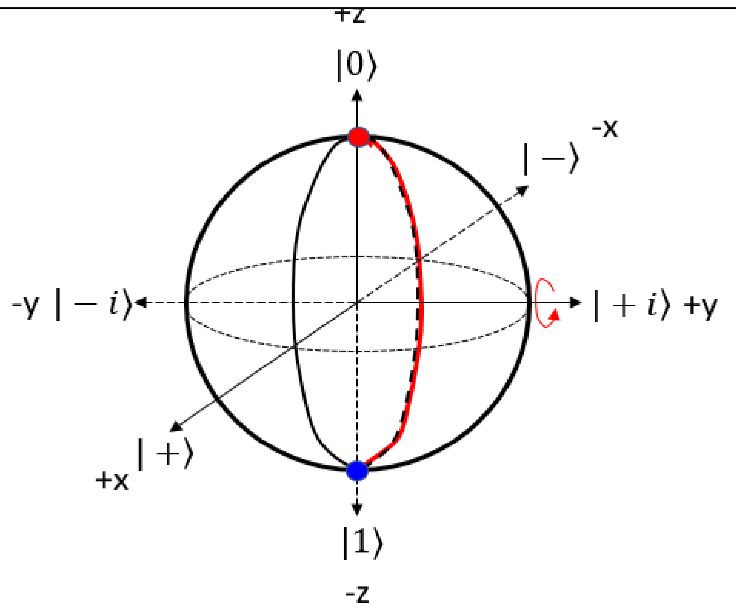
$$|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

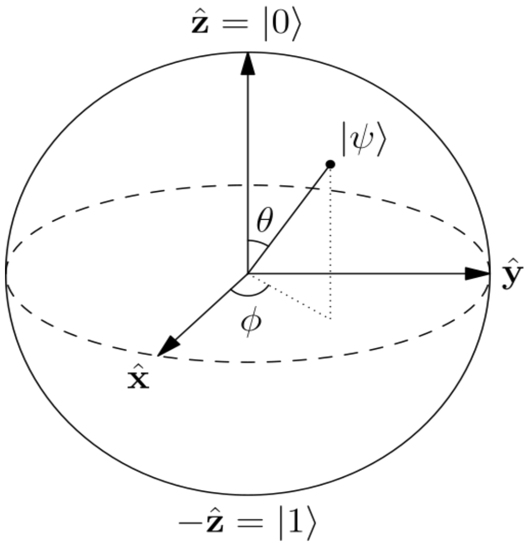
- $\theta = \pi$
- ϕ can be any value, but typically we take $\phi = 0$.

$$|i\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\i\end{pmatrix}$$

- $\theta = \frac{\pi}{2}$ (since both $|0\rangle$ and $|1\rangle$ components have equal magnitude)
- $\phi = \frac{\pi}{2}$ (due to the i phase factor, which corresponds to a phase of $\frac{\pi}{2}$).

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





$$|-i\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-i\end{pmatrix}$$

- $\theta = \frac{\pi}{2}$ (since both $|0\rangle$ and $|1\rangle$ components have equal magnitude)
- $\phi = -\frac{\pi}{2}$ (due to the $-i$ phase factor, which corresponds to a phase of $-\frac{\pi}{2}$).

$$|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

- $\theta = \frac{\pi}{2}$ (since both $|0\rangle$ and $|1\rangle$ components have equal magnitude)
- $\phi = 0$.

$$|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

- $\theta = \frac{\pi}{2}$ (since both $|0\rangle$ and $|1\rangle$ components have equal magnitude)
- $\phi = \pi$ (due to the -1 phase factor).

Quiz 2

Open Source SDKs for Quantum Computing

- There are several open-source SDKs (**Software Development Kits**) available for **quantum computing**, each designed to help developers and researchers build, test, and run quantum algorithms on various **quantum hardware and simulators**.

Open Source SDKs for Quantum Computing

Sl.NO	SDK	Developer	Language
1	**Qiskit**	IBM Research and the Qiskit community.	Python
2	**Cirq**	Google	Python
3	**Braket**	Amazon	Python
4	**Forest**	Rigetti Computing	Python
5	**Ocean**	D-Wave Systems.	Python
6	**ProjectQ**	ETH Zurich	Python
7	**QDK**	Microsoft	Q#
Others : Strawberry Fields (Xanadu)(**Python**), Quipper (**Haskel**), PennyLane(**Python**),etc.			

Qiskit Key Features

- **Components**: Includes
  1. **Terra[Earth]** : circuit construction
  2. **Aer [Air]** : simulation,
  3. **Ignis [Fire]** : error mitigation, and
  4. **Aqua [Water**] : application-specific algorithms.
- **Ecosystem:** Supports a wide range of quantum algorithms and applications, including finance, chemistry, and machine learning.
- **Community**: Strong community support with extensive tutorials and resources.
- **Use Cases**: Suitable for both beginners and advanced users, enabling access to IBM's quantum hardware and simulators.

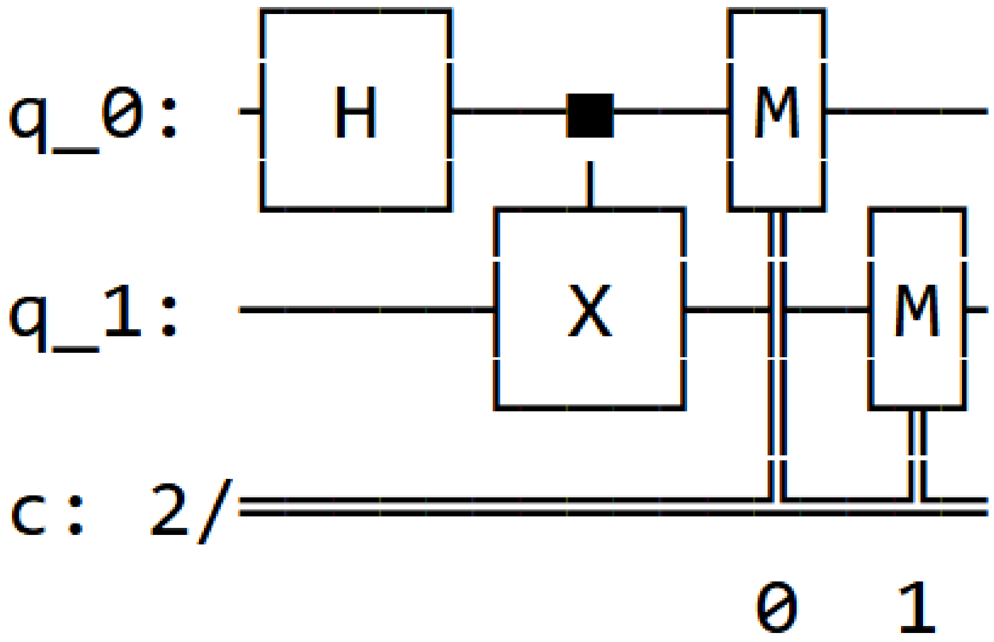
Installing Qiskit

- **pip install qiskit**
  - [This This will install the latest stable version of Qiskit, including: Q**iskit Terra, Qiskit Aer, Qiskit Ignis and Qiskit Aqua**]
  - Upgrading **qistkit**: **pip install --upgrade qiskit**
- **Installing Individual Components**
  - **pip install qiskit-terra**
  - **pip install qiskit-aer**
  - **pip install qiskit-ignis**
  - **pip install qiskit-aqua**

Verifying the Installation

```python
import qiskit
from qiskit import QuantumCircuit

# Create a simple quantum circuit
qc = QuantumCircuit(2, 2)
qc.h(0)
qc.cx(0, 1)
qc.measure([0, 1], [0, 1])

# Print the circuit
print(qc.draw())
```

Verifying the Installation



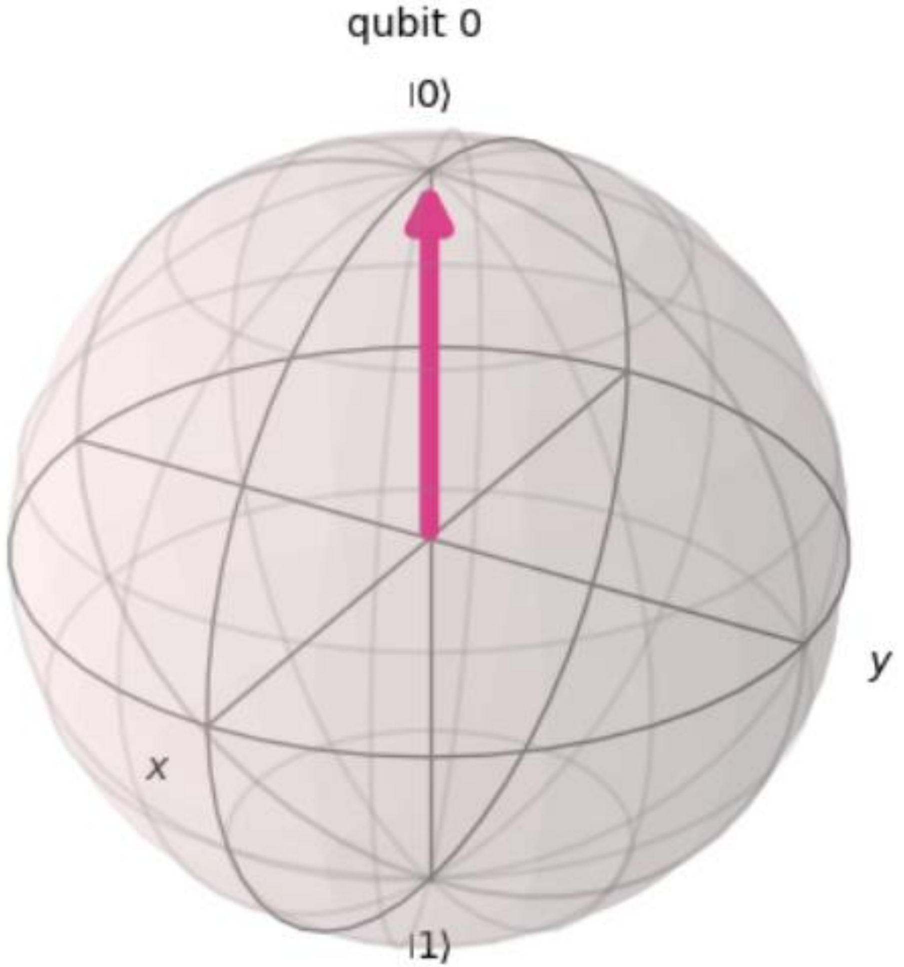
Qiskit Python Programs to representing Single Qubit States

- **$|0\rangle$ state**
- **$|1\rangle$ state**
- **$|i\rangle$ state**
- **$|-i\rangle$ state**
- **$|+\rangle$ state**
- **$|-\rangle$ state**

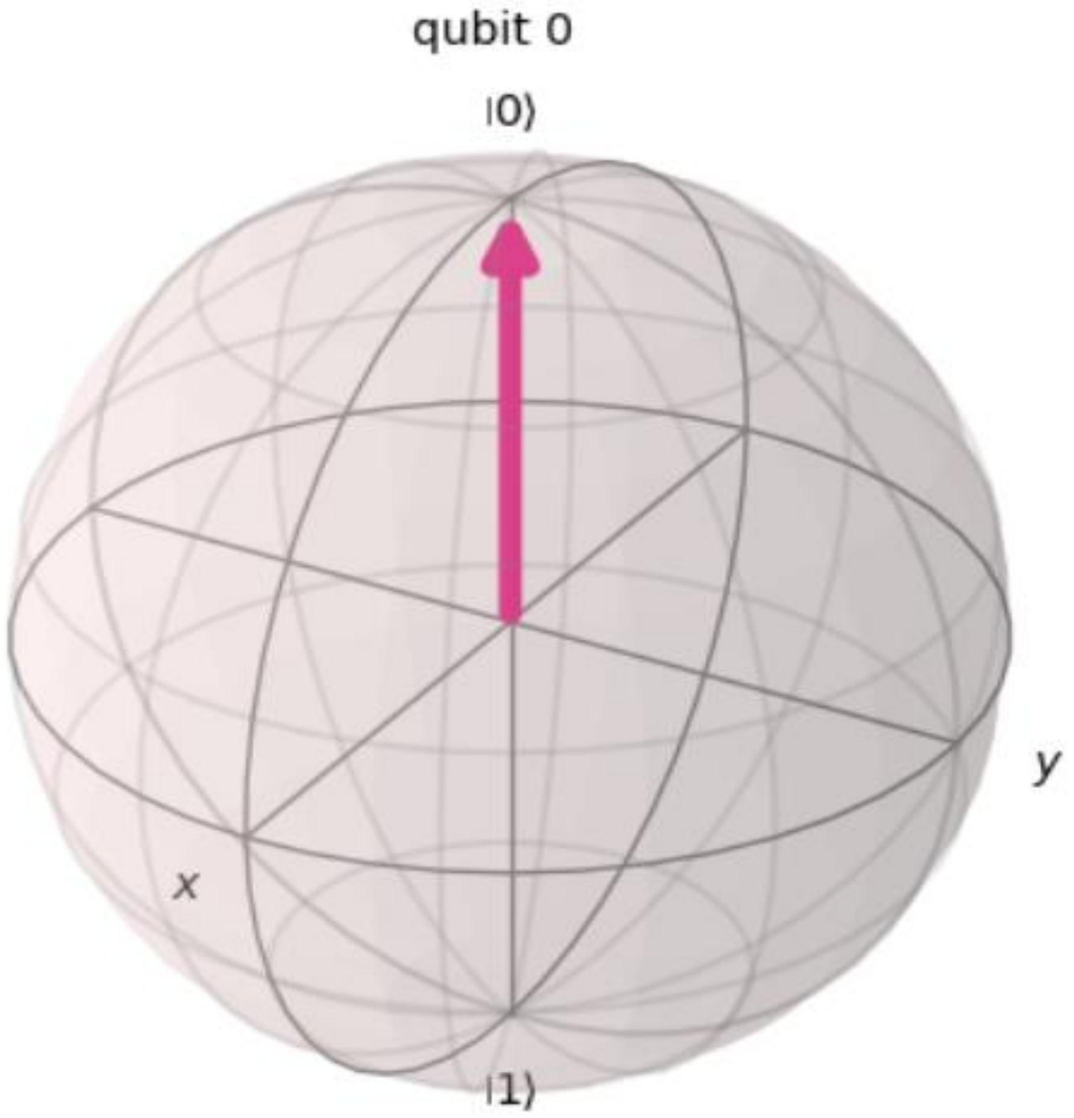
1. Qiskit Python program to represent the $|0\rangle$ state vector

```
from qiskit import QuantumCircuit
from qiskit.quantum_info import Statevector

# Create a quantum circuit with 1 qubit
qc = QuantumCircuit(1)
# Qiskit by default initialises |0> state
# Get the statevector
state = Statevector.from_instruction(qc)

# Print the statevector
print(state)

# Optionally, visualize the statevector
state.draw('bloch')
```

```
Statevector([1.+0.j, 0.+0.j],
            dims=(2,))
```



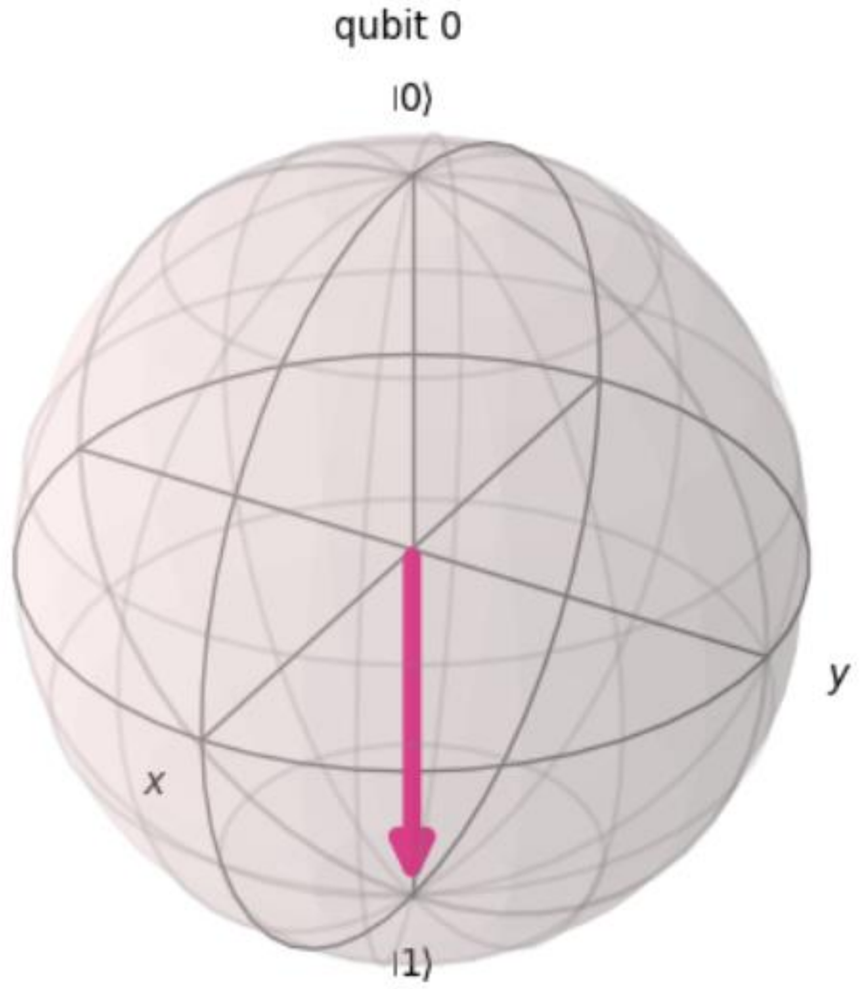
```
Statevector([1.+0.j, 0.+0.j],
            dims=(2,))
```



2. Qiskit Python program to represent the |1⟩ state vector

```python
# |1⟩ state
qc_1 = QuantumCircuit(1)
qc_1.x(0)
state = Statevector.from_instruction(qc_1)
print(f"\n|1⟩ state:")
print(state)
state.draw('bloch')
```

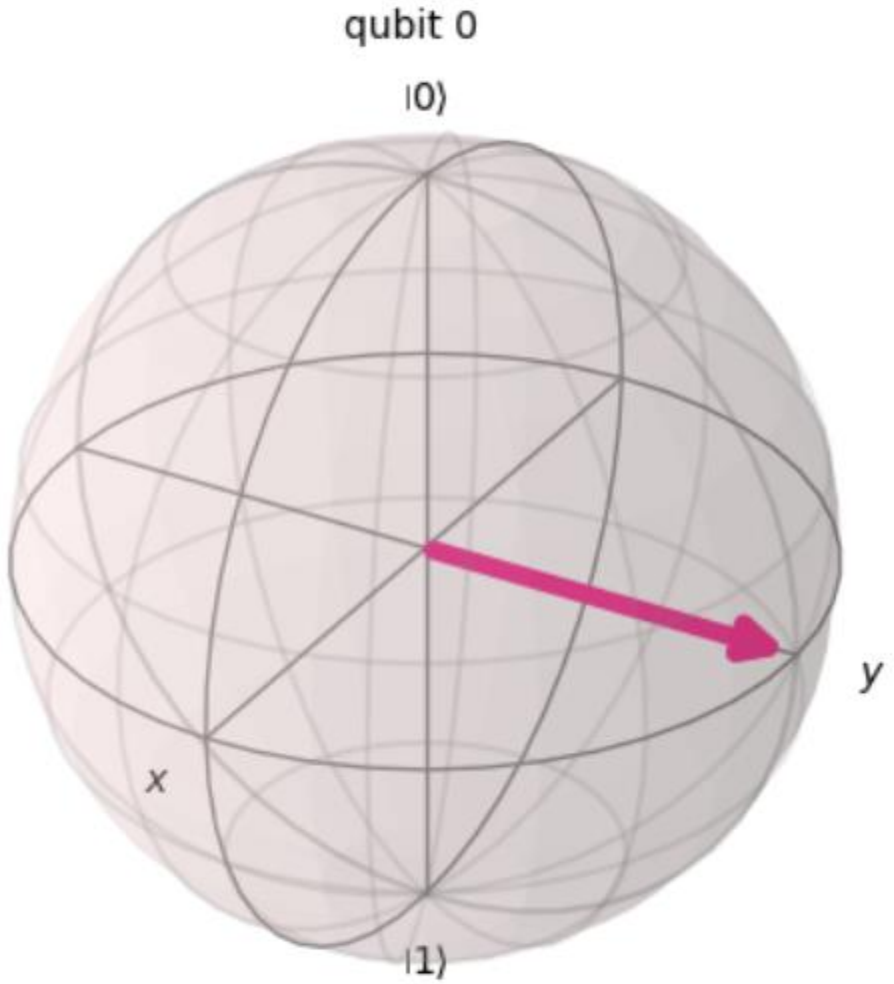
```
|1⟩ state:
Statevector([0.+0.j, 1.+0.j],
            dims=(2,))
```



3. Qiskit Python program to represent the |i⟩ state vector

```
# |i⟩ state
qc_i = QuantumCircuit(1)
qc_i.h(0)
qc_i.s(0)
state = Statevector.from_instruction(qc_i)
print(f"\n|i⟩ state:")
print(state)
state.draw('bloch')
```

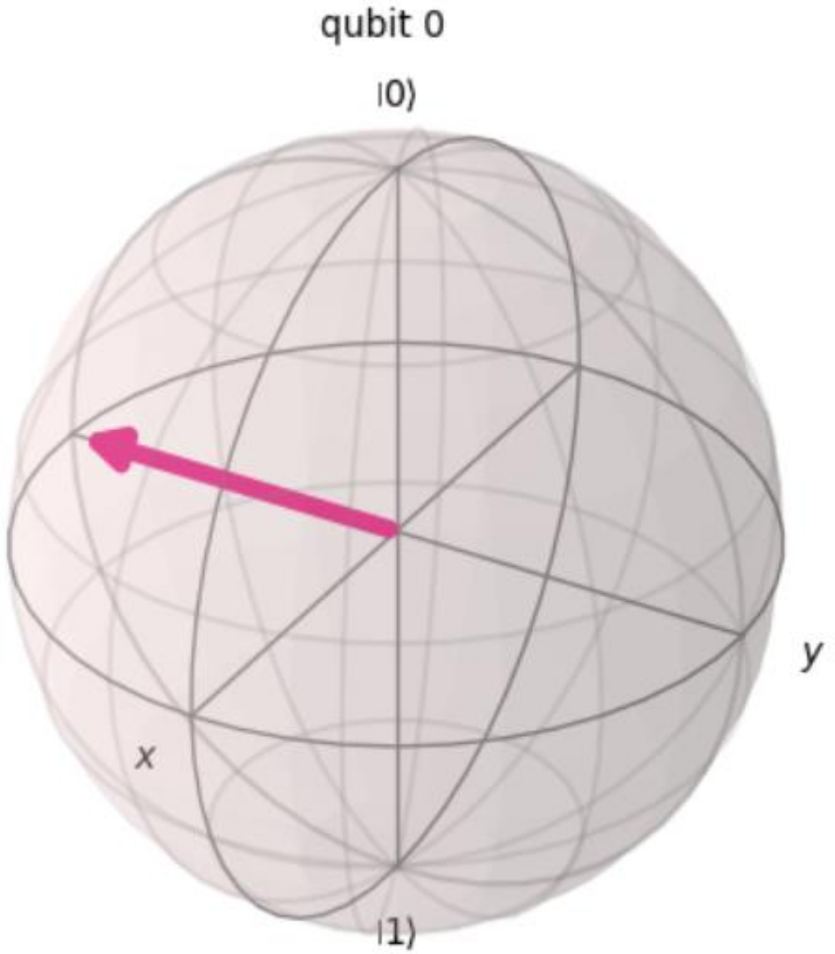
```
|i⟩ state:
Statevector([0.70710678+0.j        , 0.        +0.70710678j],
            dims=(2,))
```



4. Qiskit Python program to represent the |-i⟩ state vector

```
# |-i⟩ state
qc_minus_i = QuantumCircuit(1)
qc_minus_i.h(0)
qc_minus_i.sdg(0)
state = Statevector.from_instruction(qc_minus_i)
print(f"\n|-i⟩ state:")
print(state)
state.draw('bloch')
```

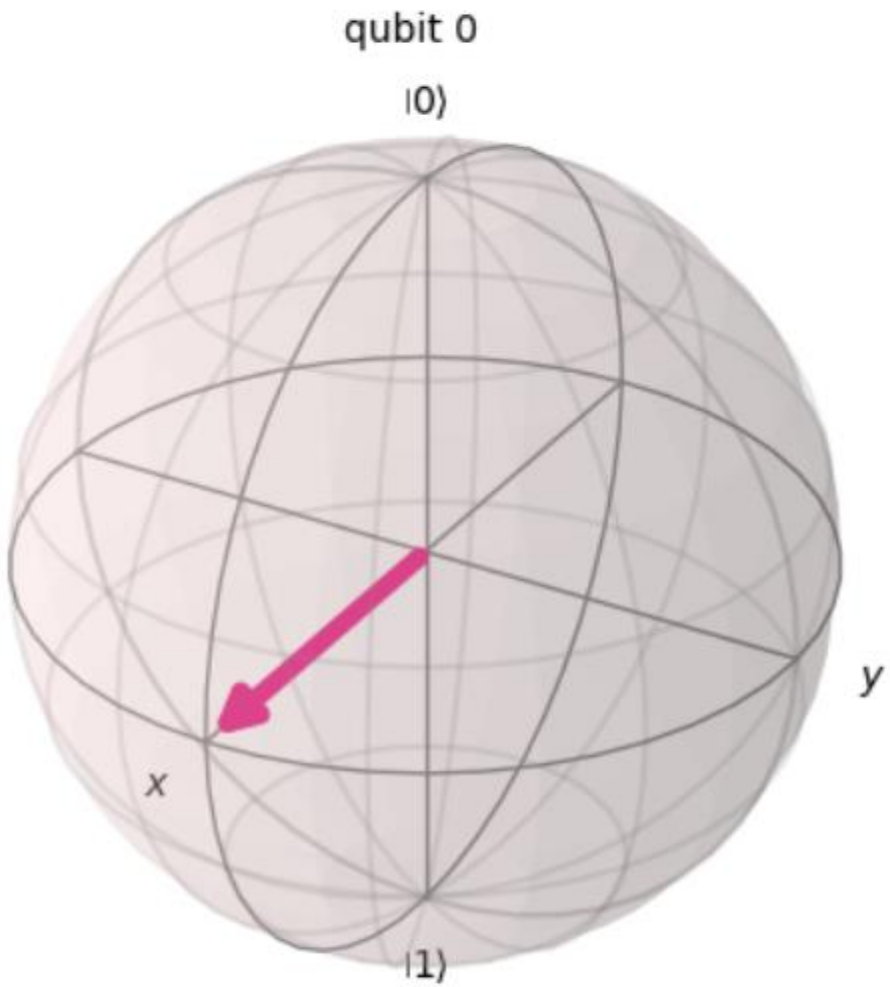
```
|-i⟩ state:
Statevector([0.70710678+0.j        , 0.        -0.70710678j],
            dims=(2,))
```



5. Qiskit Python program to represent the |+⟩ state vector

```
# |+⟩ state
qc_plus = QuantumCircuit(1)
qc_plus.h(0)
state = Statevector.from_instruction(qc_plus)
print(f"\n|+⟩ state:")
print(state)
state.draw('bloch')
```

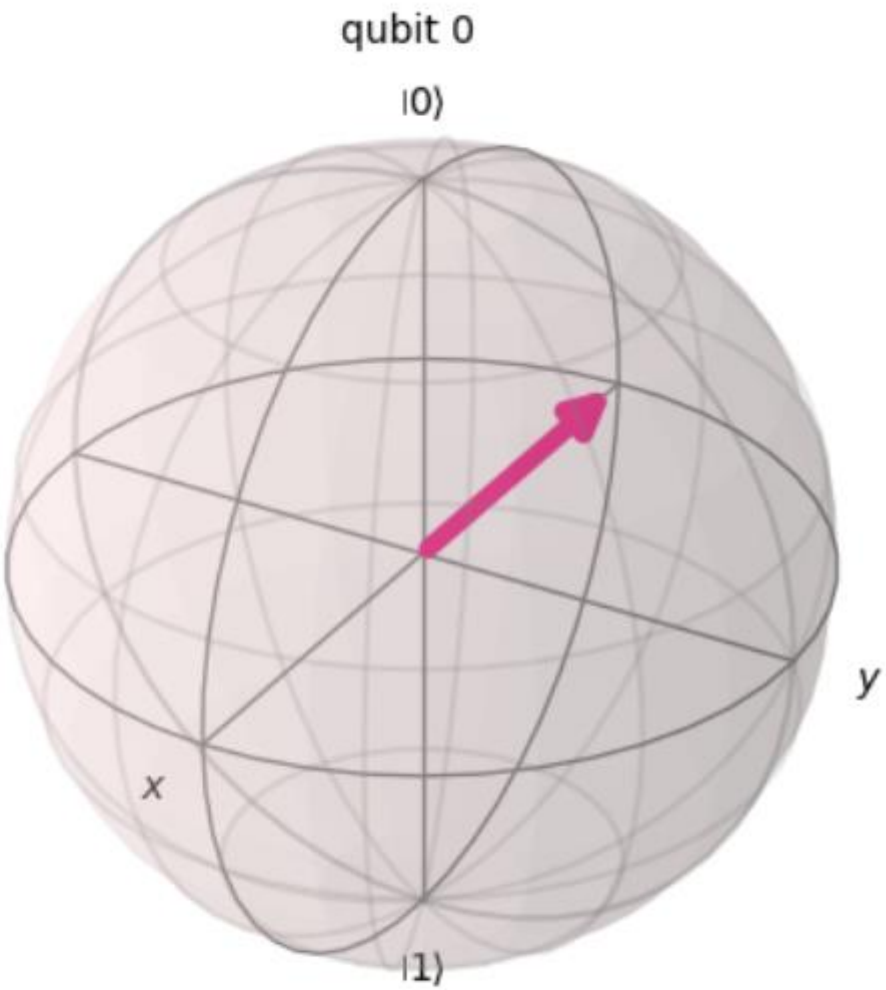
```
|+⟩ state:
Statevector([0.70710678+0.j, 0.70710678+0.j],
            dims=(2,))
```



6. Qiskit Python program to represent the |-⟩ state vector

```
# |-⟩ state
qc_minus = QuantumCircuit(1)
qc_minus.x(0)
qc_minus.h(0)
state = Statevector.from_instruction(qc_minus)
print(f"\n|-⟩ state:")
print(state)
state.draw('bloch')
```

```
|-⟩ state:
Statevector([ 0.70710678+0.j, -0.70710678+0.j],
            dims=(2,))
```



Quiz 3

Quantum Gates and Quantum Circuits

Dr. Thyagaraju G S

Professor and HoD, Department of CSE,

SDM Institute Of Technology, Ujire-574240

Unitary Operations

A **unitary operation** in quantum computing is a mathematical transformation that evolves the state of a quantum system in a way that preserves the total probability. These operations are represented by unitary matrices, which are square matrices U that satisfy the condition:

$$U^\dagger U = UU^\dagger = I$$

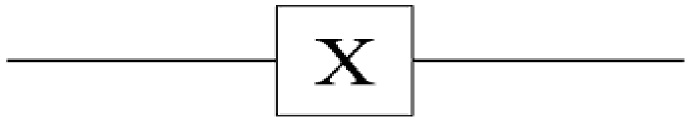
where:

- $U^\dagger$ is the Hermitian adjoint (or conjugate transpose) of U.
- I is the identity matrix.

1. Quantum Gates

1. Pauli Gates (X, Y, Z)
2. Hadamard Gate (H)
3. Phase Gates (S, T)
4. Controlled Gates(CX,XZ)
5. Swap Gate
6. Toffoli Gate (CCNOT)
7. Fredkin Gate (CSWAP)
8. Identity Gate (I)
9. Rotation Gates (Rx, Ry, Rz)

1.1.1 Pauli X Gate



a) X Gate (Pauli-X)

- **Operation:** The X gate is analogous to the classical NOT gate. It flips the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice versa.
- **Matrix Representation:**

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

- **Effect on Qubits:**
  - $X|0\rangle = |1\rangle$
  - $X|1\rangle = |0\rangle$

Pauli X Gate

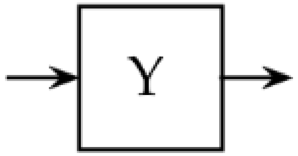
The X-gate is represented by the Pauli-X matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

To see the effect a gate has on a qubit, we simply multiply the qubit's statevector by the gate. We can see that the X-gate switches the amplitudes of the states $|0\rangle$ and $|1\rangle$:

$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

1.1.2 Y Gate (Pauli-Y)

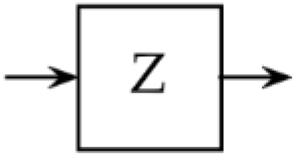


- **Operation:** The Y gate flips the qubit state and adds a phase of π (or 180 degrees).
- **Matrix Representation:**

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

- **Effect on Qubits:**
  - $Y|0\rangle = i|1\rangle$
  - $Y|1\rangle = -i|0\rangle$

1.1.3 Z Gate (Pauli-Z)



- **Operation:** The Z gate flips the phase of the qubit state $|1\rangle$, leaving $|0\rangle$ unchanged. It is often called the phase-flip gate.
- **Matrix Representation:**

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

- **Effect on Qubits:**
  - $Z|0\rangle = |0\rangle$
  - $Z|1\rangle = -|1\rangle$

1.2 Hadamard Gate (H)

- **Operation:** The Hadamard gate creates a superposition of states. It maps the basis states $|0\rangle$ and $|1\rangle$ to an equal superposition of $|0\rangle$ and $|1\rangle$.
- **Matrix Representation:**

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

- **Effect on Qubits:**
  - $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$
  - $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$

  Example: Applying the Hadamard gate to $|0\rangle$ puts the qubit in an equal superposition of $|0\rangle$ and $|1\rangle$:

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

1.3.1 S Gate (Phase Gate)

- **Operation:** The S gate is a phase shift gate that applies a phase of $\pi/2$ to the qubit.
- **Matrix Representation:**

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

- **Effect on Qubits:**
  - $S|0\rangle = |0\rangle$
  - $S|1\rangle = i|1\rangle$

1.3.2 T Gate (Phase Gate)

- **Operation:** The T gate is another phase shift gate, applying a phase of $\pi/4$.
- **Matrix Representation:**

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$$

- **Effect on Qubits:**
  - $T|0\rangle = |0\rangle$
  - $T|1\rangle = e^{i\pi/4}|1\rangle$

Note

Basis States for a Single Qubit

A single qubit has two possible basis states:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Tensor Product for Two Qubits

When we combine two qubits, the possible states are the tensor products of the individual qubit states. For example:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Similarly, for the state $|11\rangle$:

$$|11\rangle = |1\rangle \otimes |1\rangle$$

Computing the Tensor Product

Let's compute the tensor product of $|1\rangle \otimes |1\rangle$:

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Now, compute the tensor product:

$$|11\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 1 \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

The state $|11\rangle$ in matrix (vector) form is:

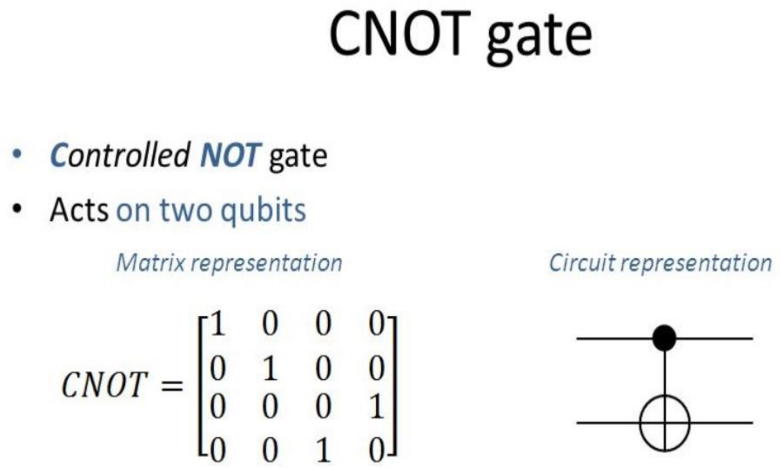
$$|11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

1.4.1 CNOT Gate (Controlled-X)

The Controlled-NOT (CNOT) gate, also known as the Controlled-X gate, is a two-qubit gate that flips the state of the target qubit if the control qubit is in the state |1>

- **Operation:** The CNOT gate flips the target qubit if the control qubit is $|1\rangle$.
- **Matrix Representation:**

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



- **Effect on Qubits:**
  - If the control qubit is $|0\rangle$, the target qubit remains unchanged.
  - If the control qubit is $|1\rangle$, the target qubit is flipped.

Example: For control qubit $|1\rangle$ and target qubit $|0\rangle$:

$$\mathrm{CNOT}|10\rangle = |11\rangle$$

CNOT Gate Matrix Representation

The CNOT gate is represented by the following 4×4 matrix:

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

Basis States and Corresponding Vectors

The CNOT gate acts on two qubits, which have four possible basis states. These basis states are represented as vectors:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

Applying CNOT to $|10\rangle$

The input state $|10\rangle$ corresponds to the vector:

$$|10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Now, apply the CNOT gate matrix to this vector:

$$\text{CNOT} \cdot |10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 1 + 0 \cdot 0 \\ 0 \cdot 1 + 0 \cdot 0 + 0 \cdot 1 + 0 \cdot 0 \\ 0 \cdot 0 + 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 0 \\ 0 \cdot 0 + 0 \cdot 0 + 1 \cdot 1 + 0 \cdot 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

This corresponds to the quantum state $|11\rangle$.

1.4.2 Controlled-Z Gate (CZ)

The Controlled-Z (CZ) gate is a two-qubit quantum gate that applies a Z gate (also known as a phase-flip gate) to the second qubit, but only if the first qubit (the control qubit) is in the state |1⟩|1\rangle|1⟩. Otherwise, it leaves both qubits unchanged.

- **Operation:** The CZ gate applies a Z gate to the target qubit if the control qubit is $|1\rangle$.
- **Matrix Representation:**

$$\mathrm{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

- **Effect on Qubits:**
  - If the control qubit is $|1\rangle$, the target qubit's phase is flipped.

Applying the CZ Gate to Basis States

1. $|00\rangle$ **state:**

$$CZ|00\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle$$

The $|00\rangle$ state remains unchanged.

2. $|01\rangle$ **state:**

$$CZ|01\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle$$

The $|01\rangle$ state remains unchanged.

Applying the CZ Gate to Basis States

3. $|10\rangle$ **state:**

$$CZ|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |10\rangle$$

The $|10\rangle$ state remains unchanged.

4. $|11\rangle$ **state:**

$$CZ|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -1 \end{pmatrix} = -|11\rangle$$

The $|11\rangle$ state acquires a phase flip (multiplied by -1).

1.5 Swap Gate

- **Operation:** The Swap gate swaps the states of two qubits.
- **Matrix Representation:**

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

- **Effect on Qubits:**
  - The states of the two qubits are exchanged.

Applying the Swap Gate to Basis States

1. $|00\rangle$ **state:**

$$\mathrm{SWAP}|00\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle$$

The $|00\rangle$ state remains unchanged.

2. $|01\rangle$ **state:**

$$\mathrm{SWAP}|01\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |10\rangle$$

The $|01\rangle$ state is swapped to $|10\rangle$.

3. $|10\rangle$ **state:**

$$\text{SWAP}|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle$$

The $|10\rangle$ state is swapped to $|01\rangle$.

4. $|11\rangle$ **state:**

$$\text{SWAP}|11\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |11\rangle$$

The $|11\rangle$ state remains unchanged.

1.6 Toffoli Gate (CCNOT)

- **Operation:** The Toffoli gate is a three-qubit gate where two qubits act as control qubits, and the third is the target qubit. The target qubit is flipped if both control qubits are $|1\rangle$.
- **Matrix Representation:** The matrix is 8x8, but conceptually:
  - If both control qubits are $|1\rangle$, flip the target qubit.

  Example: If control qubits are $|11\rangle$ and target qubit is $|0\rangle$, the Toffoli gate gives $|111\rangle$.

Matrix Representation

The Toffoli gate is represented by an 8x8 unitary matrix:

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

The Toffoli gate performs the following operation on the computational basis states:

- $|000\rangle \rightarrow |000\rangle$
- $|001\rangle \rightarrow |001\rangle$
- $|010\rangle \rightarrow |010\rangle$
- $|011\rangle \rightarrow |011\rangle$
- $|100\rangle \rightarrow |100\rangle$
- $|101\rangle \rightarrow |101\rangle$
- $|110\rangle \rightarrow |111\rangle$
- $|111\rangle \rightarrow |110\rangle$

The target qubit (third qubit) is flipped only when both control qubits (first and second qubits) are **|1⟩**.

State Vector |110⟩:

The state |110⟩ corresponds to the following column vector (indexing from 0):

$$|110\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Applying the Toffoli gate to $|110\rangle$:

$$\text{Toffoli} \cdot |110\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |111\rangle$$

1.7 Identity Gate (I)

- **Operation:** The Identity gate does nothing to the qubit. It is equivalent to a "do nothing" operation.
- **Matrix Representation:**

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

- **Effect on Qubits:**
  - $I|0\rangle = |0\rangle$
  - $I|1\rangle = |1\rangle$

1.8.1 Rx Gate

- **Operation:** Rotates the qubit around the X-axis.
- **Matrix Representation:**

$$R_x(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) X$$

- **Effect on Qubits:**
  - Rotates the qubit state by θ radians around the X-axis.

The Rx gate is a **single-qubit rotation gate** that rotates the state of a qubit around the X-axis of the Bloch sphere

The Rx gate is defined as:

$$\mathrm{Rx}(\theta) = \exp\left(-i\frac{\theta}{2}X\right)$$

Where X is the Pauli-X matrix:

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

The exponential term can be expanded into:

$$\mathrm{Rx}(\theta) = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\right) X$$

Substituting the Pauli-X matrix X:

$$\mathrm{Rx}(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

Example 1: Rx Gate on |0⟩

Consider applying the Rx gate with $\theta = \frac{\pi}{2}$ to the qubit state $|0\rangle$.

1. **Initial State:**

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

2. **Rx Gate with $\theta = \frac{\pi}{2}$:**

$$\mathrm{Rx}\left(\frac{\pi}{2}\right) = \begin{pmatrix} \cos\left(\frac{\pi}{4}\right) & -i\sin\left(\frac{\pi}{4}\right) \\ -i\sin\left(\frac{\pi}{4}\right) & \cos\left(\frac{\pi}{4}\right) \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix}$$

3. **Apply Rx to $|0\rangle$:**

$$\mathrm{Rx}\left(\frac{\pi}{2}\right)|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -i \\ -i & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -i \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$$

So, the resulting state is:

$$\frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$$

Example 2: Rx Gate on |1⟩

Consider applying the Rx gate with $\theta = \pi$ to the qubit state $|1\rangle$.

1. **Initial State:**

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

2. **Rx Gate with $\theta = \pi$:**

$$\mathrm{Rx}(\pi) = \begin{pmatrix} \cos\left(\frac{\pi}{2}\right) & -i\sin\left(\frac{\pi}{2}\right) \\ -i\sin\left(\frac{\pi}{2}\right) & \cos\left(\frac{\pi}{2}\right) \end{pmatrix} = \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix}$$

3. **Apply Rx to $|1\rangle$:**

$$\mathrm{Rx}(\pi)|1\rangle = \begin{pmatrix} 0 & -i \\ -i & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -i \\ 0 \end{pmatrix} = -i|0\rangle$$

So, the resulting state is:

$$-i|0\rangle$$

1.8.2 Ry Gate

- **Operation:** Rotates the qubit around the Y-axis.
- **Matrix Representation:**

$$R_y(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Y$$

- **Effect on Qubits:**
  - Rotates the qubit state by θ radians around the Y-axis.

1.8.3 Rz Gate

Operation: Rotates the qubit around the Z-axis.

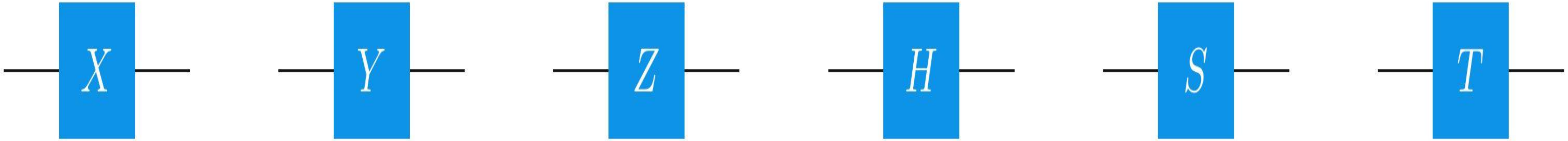
Matrix Representation:

$$R_z(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Z$$

Effect on Qubits:

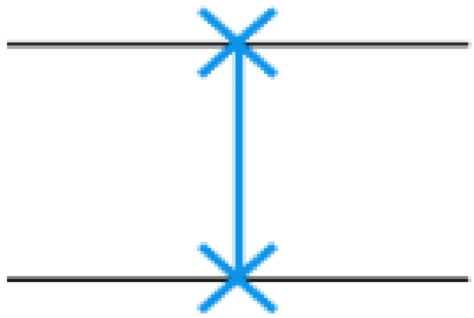
- Rotates the qubit state by θ radians around the Z-axis.

Single-qubit gates are generally shown as squares with a letter indicating which operation it is, like this:

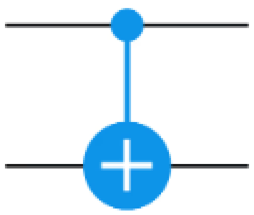


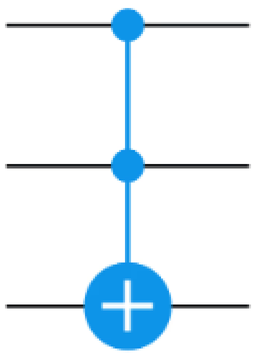
Not gates (also known as X gates) are also sometimes denoted by a circle around a plus sign:

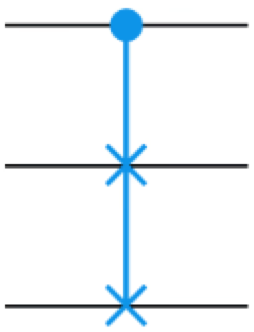
Swap gates are denoted as follows:



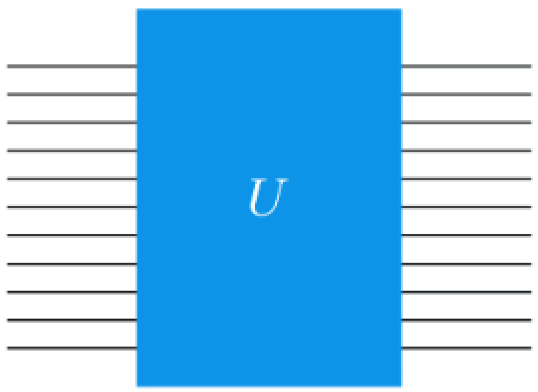
Controlled-gates, meaning gates that describe controlled-unitary operations, are denoted by a **filled-in circle (indicating the control)** connected by a vertical line to whatever operation is being controlled. For instance, **controlled-NOT gates**, **controlled-controlled-NOT (or Toffoli) gates**, and **controlled-swap (Fredkin) gates** are denoted like this:

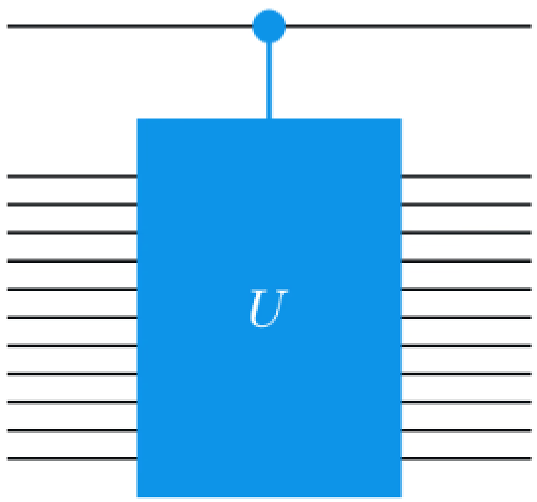






Arbitrary unitary operations on multiple qubits may be viewed as gates. They are depicted by rectangles labeled by the name of the unitary operation. For instance, here is a depiction of an (unspecified) unitary operation U as a gate, along with a controlled version of this gate:





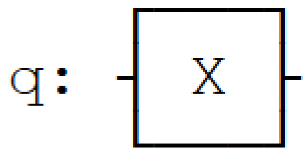
Basic Structure of a Quantum Circuit

A quantum circuit typically consists of:

- **Qubits**: The basic unit of quantum information, analogous to bits in classical computing. Qubits can exist in superposition states.
- **Quantum Gates**: Operations that change the state of qubits.
  - Single-qubit gates (e.g., Hadamard, Pauli-X, Y, Z)
  - Multi-qubit gates (e.g., CNOT, SWAP)
- **Measurement**: The final step in a quantum circuit, where the qubits are measured to produce a classical output. The process of observing qubit states, collapsing superpositions.
- **Circuit diagram**: A visual representation of qubit operations over time.
- **Initialization**: Setting qubits to known starting states.
- **Quantum register**: A collection of qubits used in the circuit.
- **Classical register**: Stores measurement results for further processing.

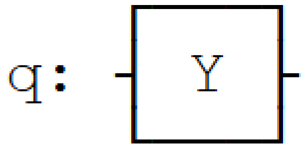
Ex1

```python
from qiskit import QuantumCircuit
circuit = QuantumCircuit(1)
circuit.x(0)
circuit.draw()
```



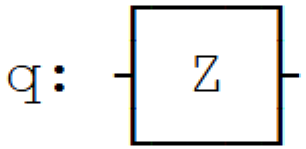
Ex2

```python
from qiskit import QuantumCircuit
circuit = QuantumCircuit(1)
circuit.y(0)
circuit.draw()
```



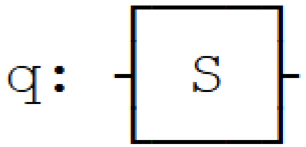
Ex3:

```python
from qiskit import QuantumCircuit
circuit = QuantumCircuit(1)
circuit.z(0)
circuit.draw()
```



Ex4:

```python
from qiskit import QuantumCircuit
circuit = QuantumCircuit(1)
circuit.s(0)
circuit.draw()
```



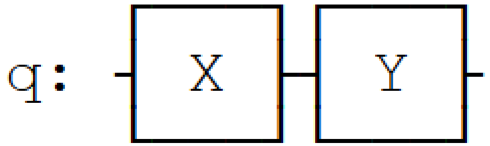
Ex5:

```
from qiskit import QuantumCircuit
circuit = QuantumCircuit(1)
circuit.ss(0)
circuit.draw()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[5], line 5
      1 from qiskit import QuantumCircuit
      3 circuit = QuantumCircuit(1)
----> 5 circuit.ss(0)
      6 circuit.draw()

AttributeError: 'QuantumCircuit' object has no attribute 'ss'
```

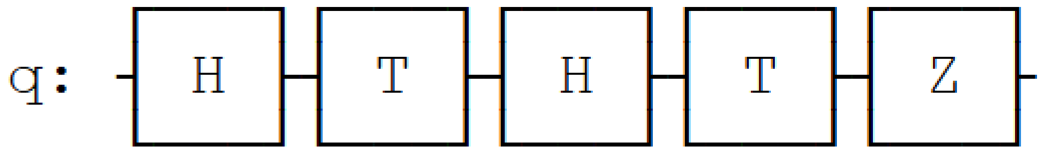
Ex6:

```python
from qiskit import QuantumCircuit
circuit = QuantumCircuit(1)
circuit.x(0)
circuit.y(0)
circuit.draw()
```



Ex7:

```python
from qiskit import QuantumCircuit
circuit = QuantumCircuit(1)
circuit.h(0)
circuit.t(0)
circuit.h(0)
circuit.t(0)
circuit.z(0)
circuit.draw()
```



If we wish to choose our own name we can do this using the Quantum Register class like this:

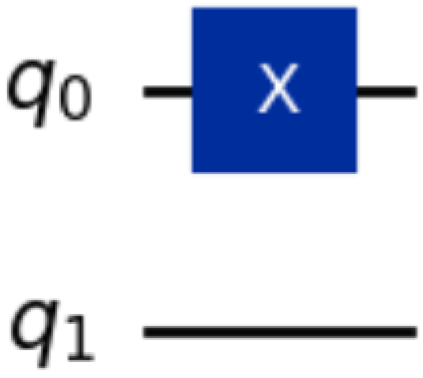
```
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.primitives import Sampler
from qiskit.visualization import plot_histogram
```

```
X = QuantumRegister(1, "X")
circuit = QuantumCircuit(X)

circuit.h(X)
circuit.s(X)
circuit.h(X)
circuit.t(X)

display(circuit.draw())
```

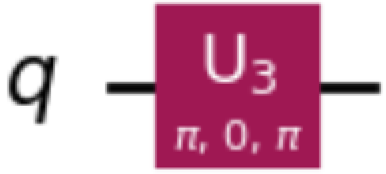
```
   ┌───┐┌───┐┌───┐┌───┐
X: ┤ H ├┤ S ├┤ H ├┤ T ├
   └───┘└───┘└───┘└───┘
```

Program to create a new circuit with two new qubits, then displays the circuit's qubits attribute

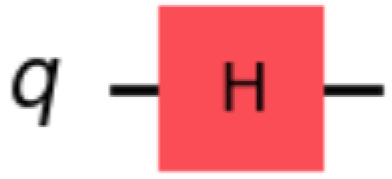
```
from qiskit import QuantumCircuit
qc = QuantumCircuit(2)
qc.x(0)  # Add X-gate to qubit 0
qc.draw("mpl")
```



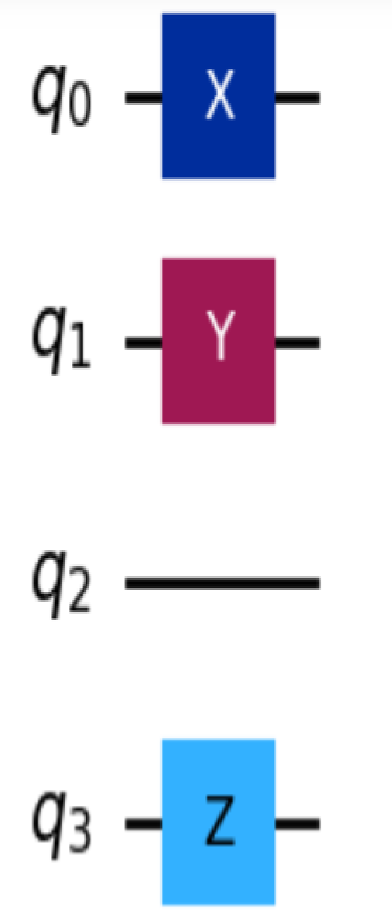

```
# Draw definition circuit of 0th instruction in `qc`
qc.data[0].operation.definition.draw("mpl")
```



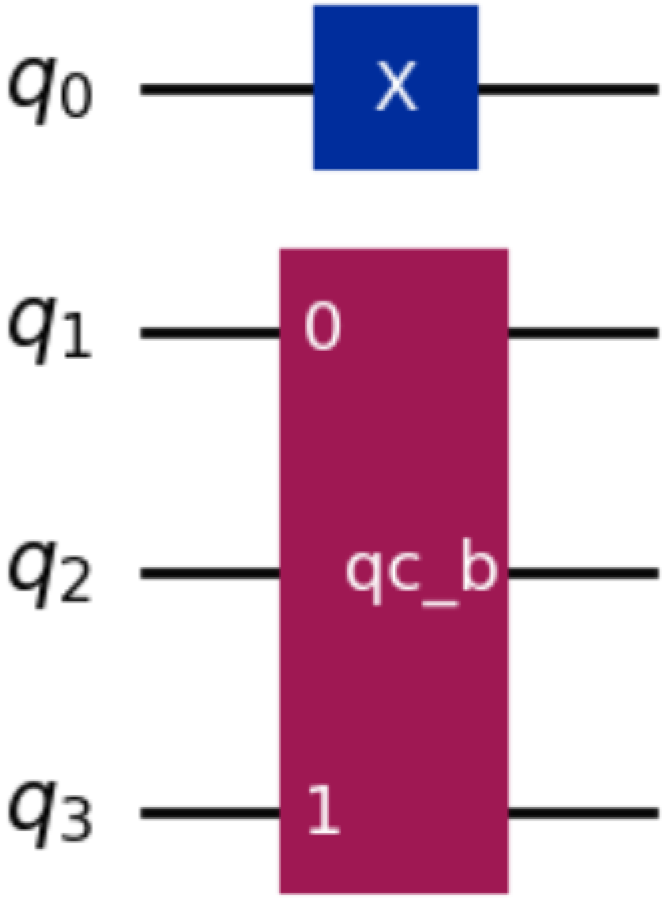
```python
from qiskit.circuit.library import HGate

qc = QuantumCircuit(1)
qc.append(
    HGate(),  # New HGate instruction
    [0]       # Apply to qubit 0
)
qc.draw("mpl")
```



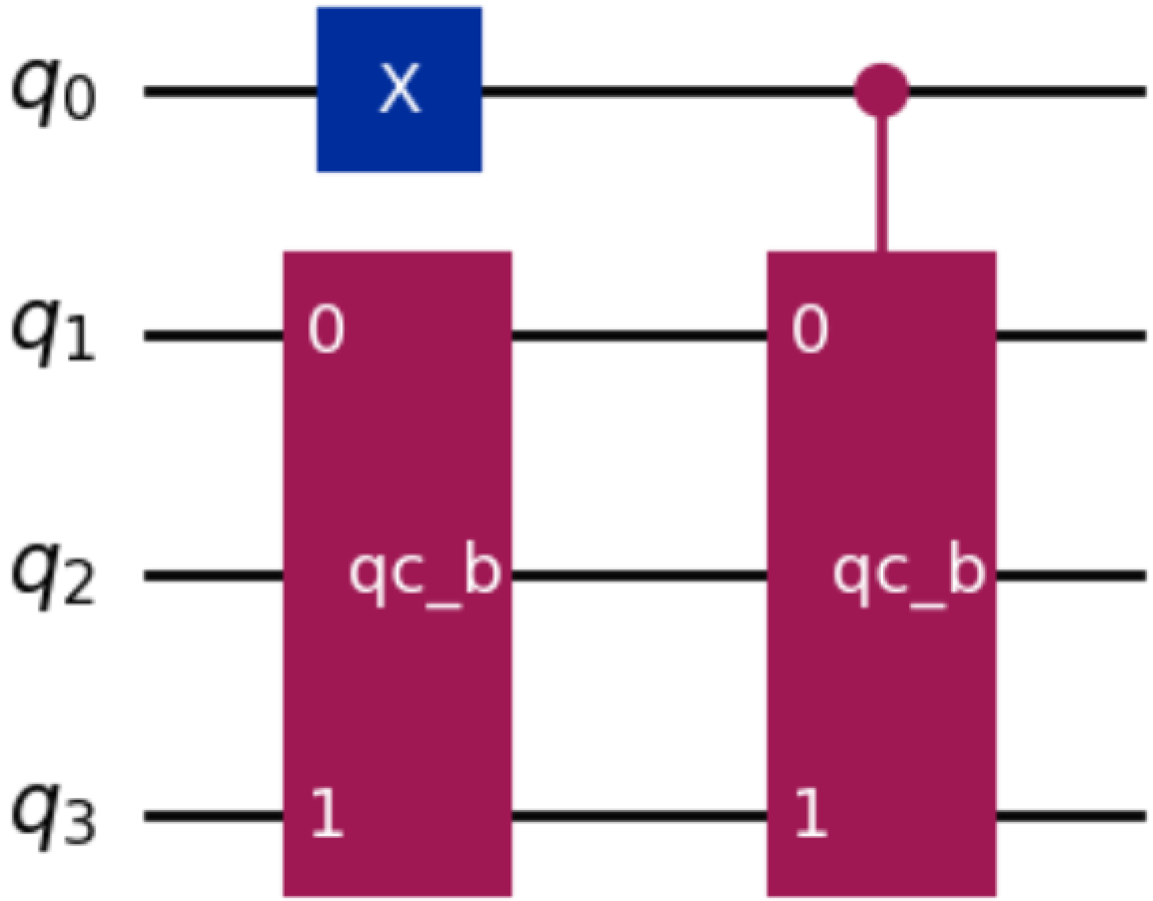
```python
qc_a = QuantumCircuit(4)
qc_a.x(0)

qc_b = QuantumCircuit(2, name="qc_b")
qc_b.y(0)
qc_b.z(1)

# compose qubits (0, 1) of qc_a to qubits (1, 3) of qc_b respectively
combined = qc_a.compose(qc_b, qubits=[1, 3])
combined.draw("mpl")
```



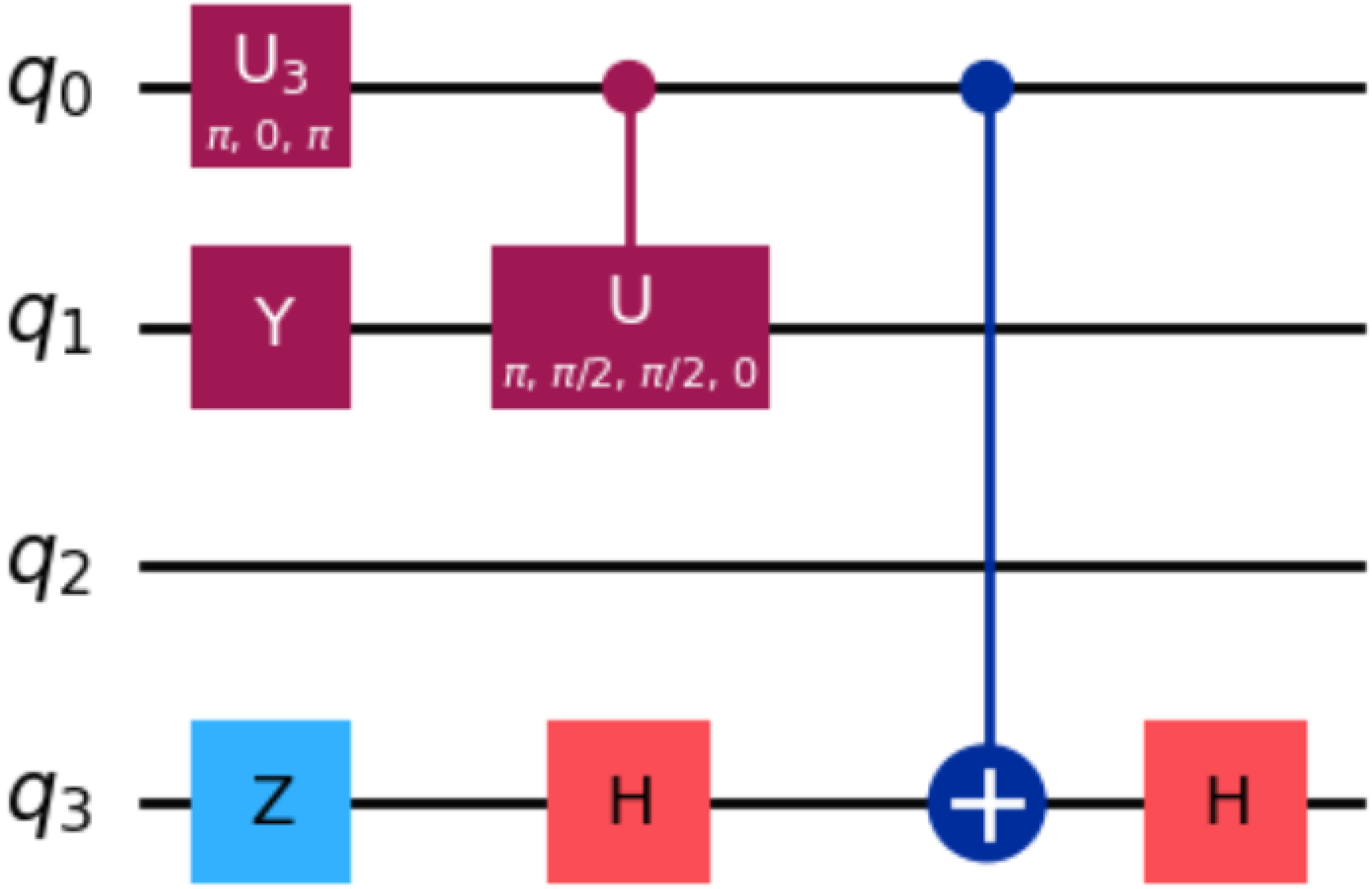
```
inst = qc_b.to_instruction()
qc_a.append(inst, [1, 3])
qc_a.draw("mpl")
```



```
gate = qc_b.to_gate().control()
qc_a.append(gate, [0, 1, 3])
qc_a.draw("mpl")
```

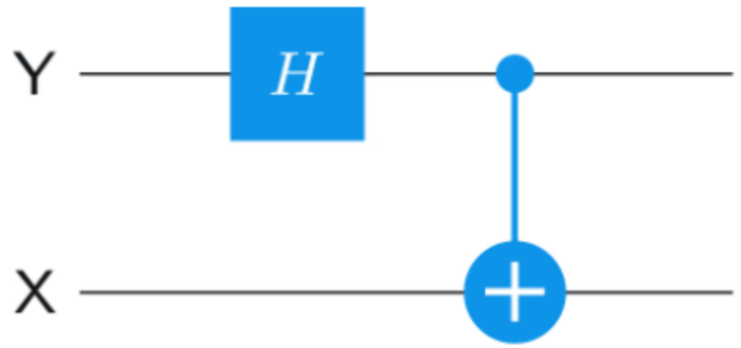


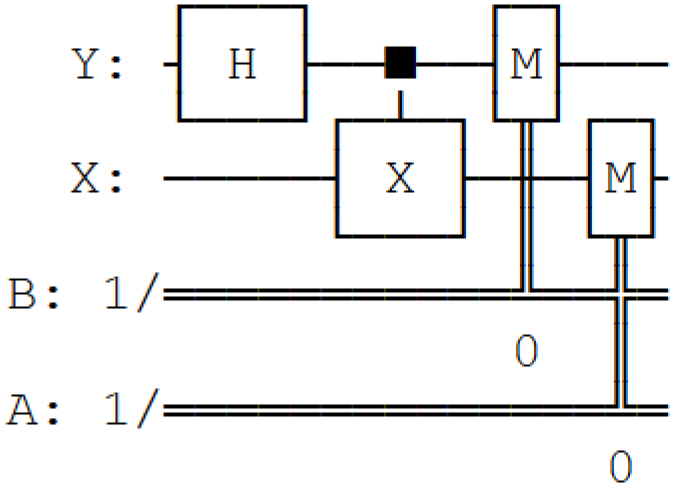
```
qc_a.decompose().draw("mpl")
```



Circuit with Hadmard and Control Gate

```
X = QuantumRegister(1, "X")
Y = QuantumRegister(1, "Y")
A = ClassicalRegister(1, "A")
B = ClassicalRegister(1, "B")

circuit = QuantumCircuit(Y, X, B, A)
circuit.h(Y)
circuit.cx(Y, X)
circuit.measure(Y, B)
circuit.measure(X, A)

display(circuit.draw())
```





Types

- Quantum circuits can be categorized based on their **functionality and the types of operations** they perform.
- Here are some simple types of quantum circuits:

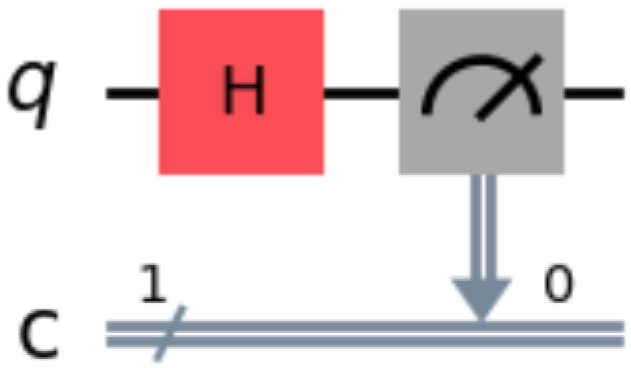
1. Basic Quantum Circuits

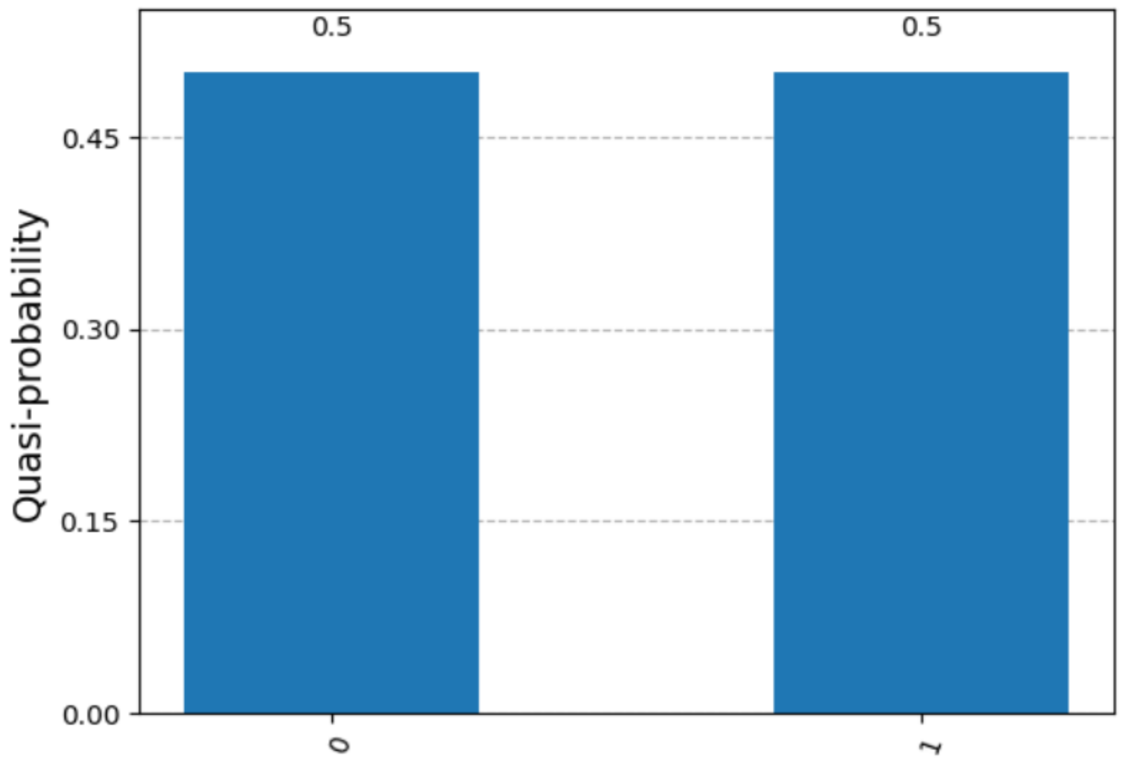
- These circuits perform simple quantum operations such as initializing qubits, applying single-qubit gates, and measuring the output.
- **Example: Basic Quantum Circuit**
- **Circuit**: Apply a Hadamard gate to a single qubit, then measure the result.
- **Operation**: This circuit creates a superposition state

Basic Quantum Circuits

```python
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.primitives import Sampler
from qiskit.visualization import plot_histogram, circuit_drawer
from IPython.display import Image

# Create Quantum and Classical Registers
qreg = QuantumRegister(1, 'q')  # 1 qubit
creg = ClassicalRegister(1, 'c')  # 1 classical bit for measurement
qc = QuantumCircuit(qreg, creg)

# Apply Hadamard gate to the qubit to create a superposition
qc.h(qreg[0])

# Measure the qubit
qc.measure(qreg[0], creg[0])
```

```python
# Save and display the circuit visualization
circuit_image = circuit_drawer(qc, output='mpl')
circuit_image.savefig('basic_quantum_circuit.png')
display(Image(filename='basic_quantum_circuit.png'))

# Simulate the circuit and plot the measurement result
sampler = Sampler()
job = sampler.run(circuits=qc)
result = job.result()
counts = result.quasi_dists[0]
plot_histogram(counts)
```

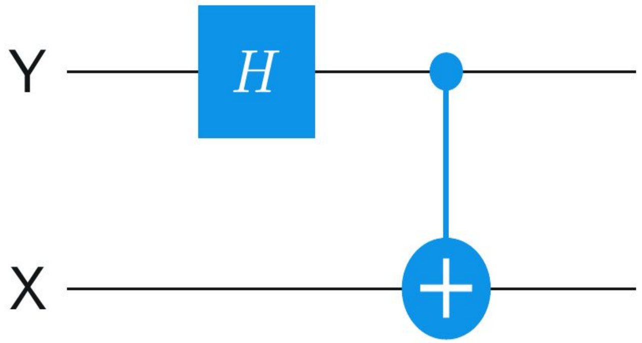




2. Entanglement Circuits

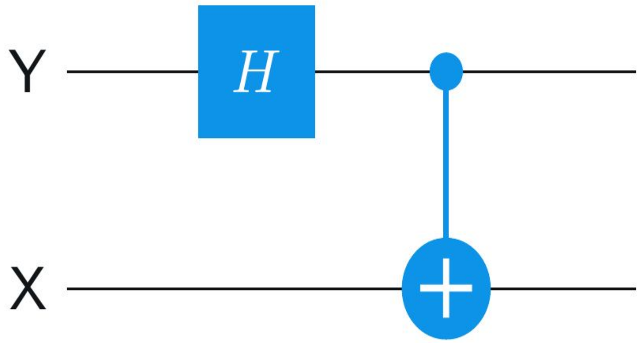
- These circuits involve creating entangled states between two or more qubits, where the state of one qubit is dependent on the state of another. The most common example is the Bell state.
- **Example: Entanglement Circuit**
- **Circuit**: Apply a Hadamard gate to the first qubit and a CNOT gate to entangle it with the second qubit.
- **Operation**: This circuit creates an entangled Bell state.

Another example of a quantum circuit, this time with two qubits



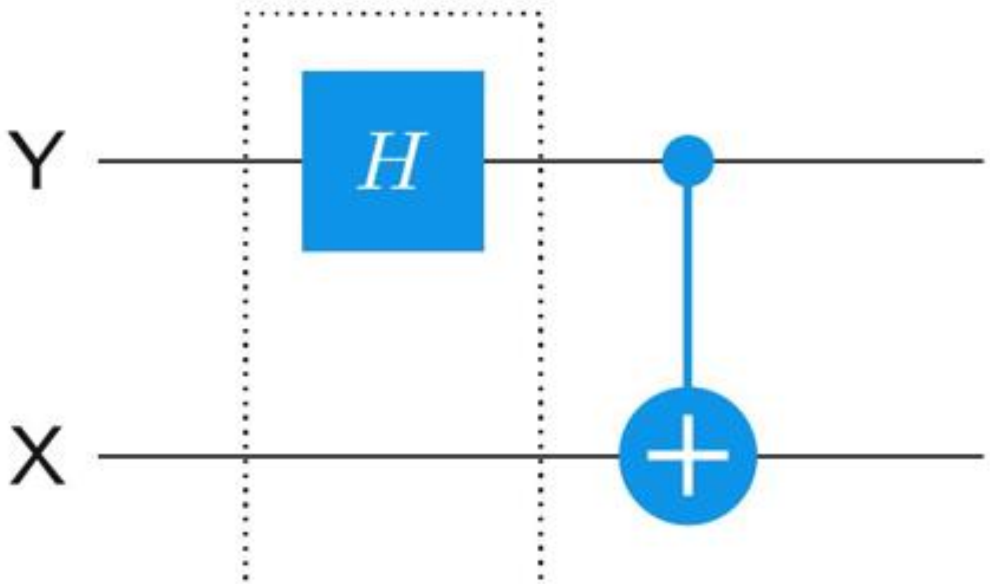
- As always, the gate labeled H refers to a Hadamard operation, while the second gate is a two-qubit gate: it's the *controlled-NOT* operation, where the **solid circle represents the control qubit** and the circle resembling the symbol $\oplus$ denotes the **target qubit**.

Another example of a quantum circuit, this time with two qubits



- Above circuit describes an operation on a pair of qubits (X,Y) — and if the input to the circuit is a quantum state |ψ⟩|ϕ⟩, then this means that the lower qubit X starts in the state |ψ⟩ and the upper qubit Y starts in the state |ϕ⟩.

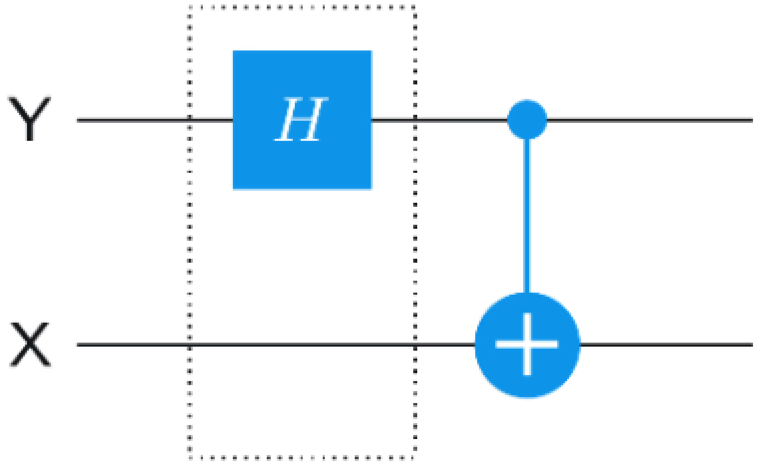
The first operation is a Hadamard operation on Y:



- When applying a gate to a single qubit like this, nothing happens to the other qubits — and nothing happening is equivalent to the identity operation being performed.

The first operation is a Hadamard operation on Y:

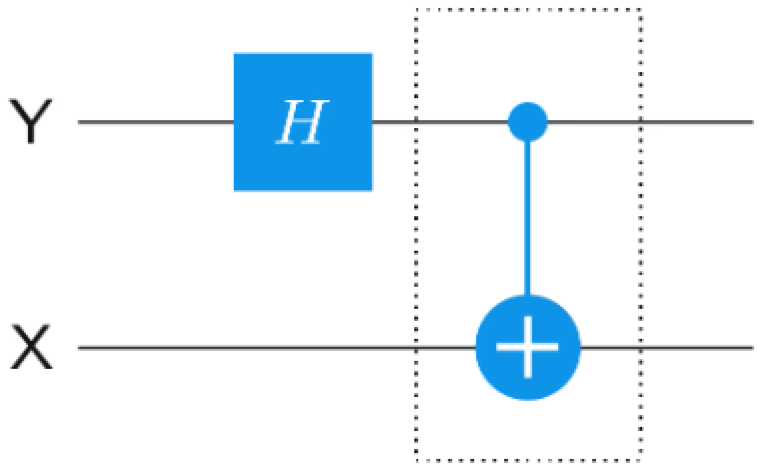
- In our circuit there is just one other qubit, X, so the dotted rectangle in the figure above represents this operation:



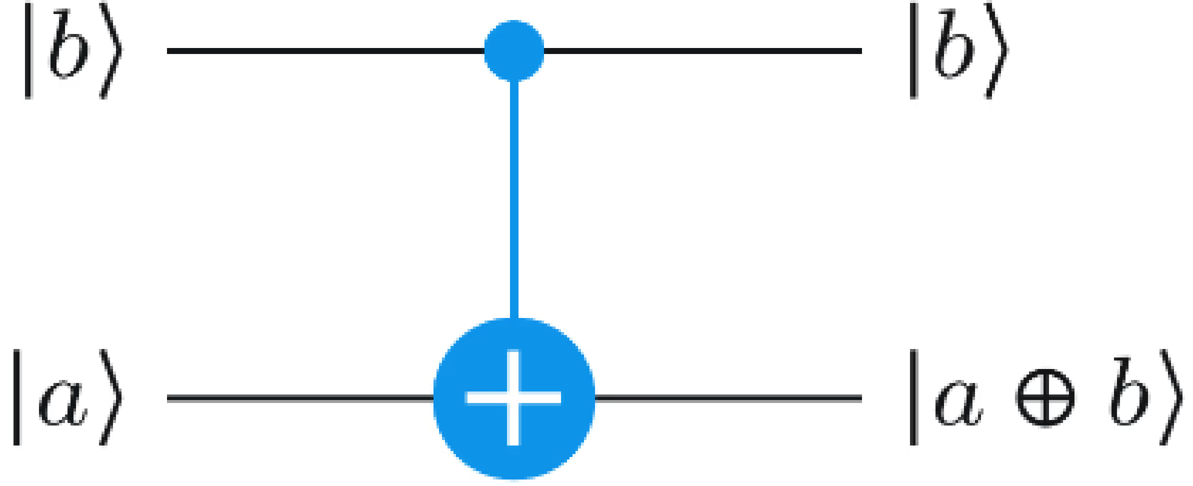
$$\mathbb{I} \otimes H = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

Note that the identity matrix is on the left of the tensor product and H is on the right, which is consistent with **Qiskit's ordering convention**.

The second operation is the controlled-NOT operation, where Y is the control and X is the target:



The controlled-NOT gate's action on standard basis states is as follows:



Given that we order the qubits as (X,Y), the matrix representation of the controlled-NOT gate is this:

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

The unitary operation of the entire circuit, which we'll call U,U, is the composition of the operations:

$$U = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 \end{pmatrix}.$$

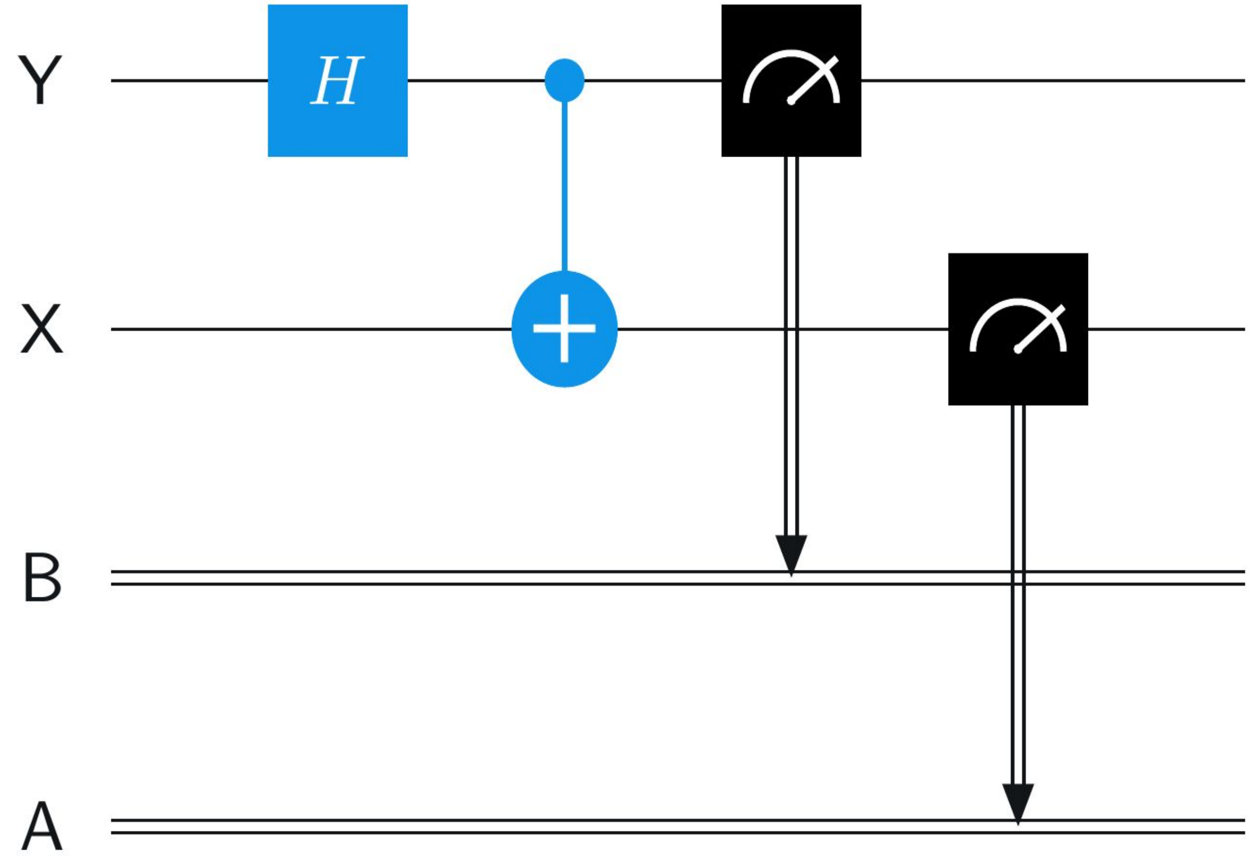
In particular, recalling our notation for the Bell states,

$$
\begin{aligned}
|\phi^+\rangle &= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \\
|\phi^-\rangle &= \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle \\
|\psi^+\rangle &= \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle \\
|\psi^-\rangle &= \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle,
\end{aligned}
$$

We get

$$
\begin{aligned}
U|00\rangle &= |\phi^+\rangle \\
U|01\rangle &= |\phi^-\rangle \\
U|10\rangle &= |\psi^+\rangle \\
U|11\rangle &= -|\psi^-\rangle.
\end{aligned}
$$

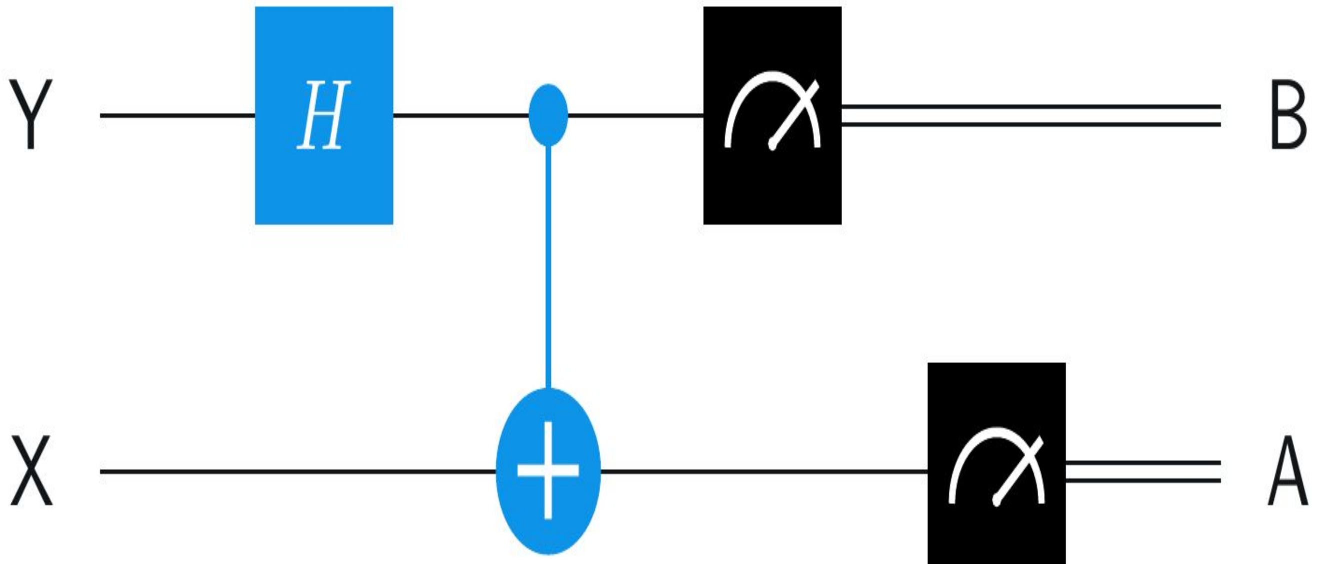
In general, quantum circuits can contain any number of qubit wires. We may also include classical bit wires, which are indicated by double lines like in this example:



- Sometimes it is convenient to depict a measurement as a gate that takes a qubit as input and outputs a classical bit (as opposed to outputting the qubit in its post-measurement state and writing the result to a separate classical bit).

  This means the measured qubit has been discarded and can safely be ignored thereafter.

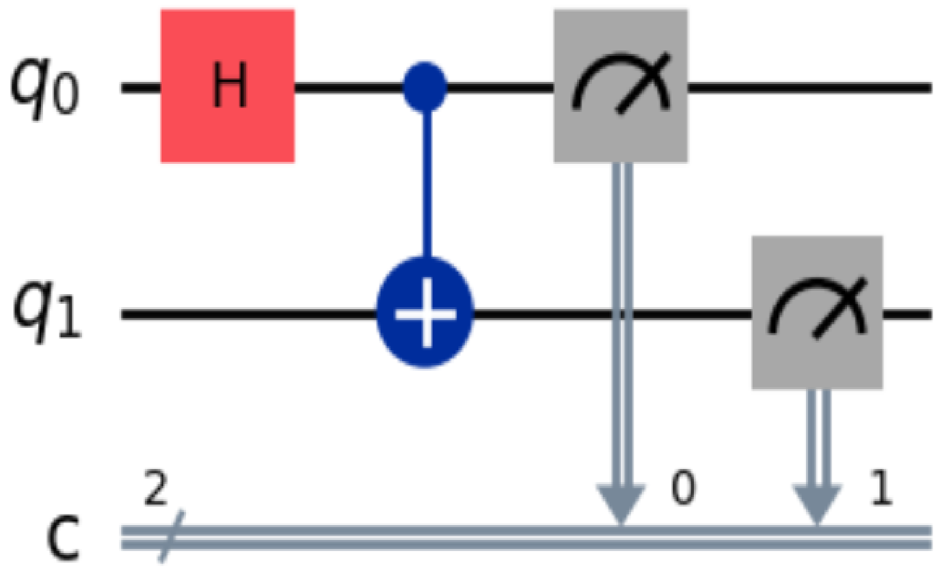
For example, the following circuit diagram represents the same process as the one in the previous diagram, but where we ignore X and Y after measuring them:

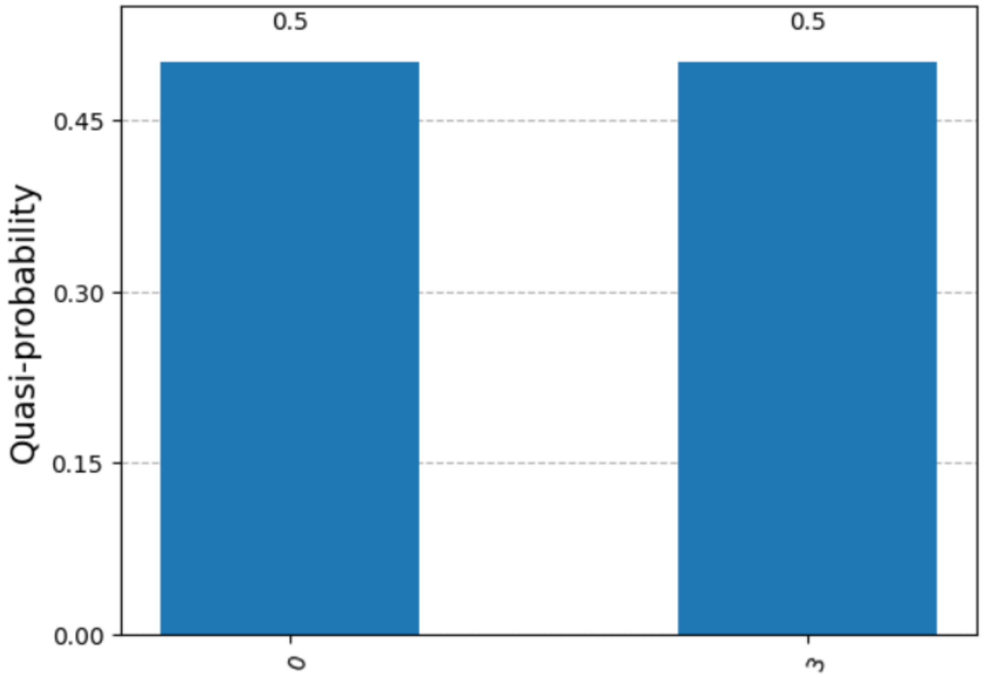


Basic Bell State Circuit

```python
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.primitives import Sampler
from qiskit.visualization import plot_histogram, circuit_drawer
from IPython.display import Image

# Quantum and Classical Registers
qreg = QuantumRegister(2, 'q')
creg = ClassicalRegister(2, 'c')
qc = QuantumCircuit(qreg, creg)

# Circuit: Create Bell State
qc.h(qreg[0])  # Hadamard on q0
qc.cx(qreg[0], qreg[1])  # CNOT on q1 with q0 as control
```

```python
# Measurement
qc.measure(qreg, creg)

# Save and display the circuit visualization
circuit_image = circuit_drawer(qc, output='mpl')
circuit_image.savefig('bell_state_circuit.png')
display(Image(filename='bell_state_circuit.png'))

# Simulate and Plot
sampler = Sampler()
job = sampler.run(circuits=qc)
result = job.result()
counts = result.quasi_dists[0]
plot_histogram(counts)
```





Quiz 4

Introduction to Quantum Algorithms

Dr. Thyagaraju G S

Professor and HoD, Department of CSE,

SDM Institute Of Technology, Ujire-574240

Quantum Algorithms

- Quantum algorithms are procedures that run on quantum computers and take advantage of quantum mechanics' principles to solve problems more efficiently than classical algorithms.
- These algorithms utilize the unique properties of quantum bits (qubits), such as **superposition, entanglement, and interference**, to perform computations that are infeasible or extremely slow on classical computers.

Types of Quantum Algorithms

1. **Quantum Search Algorithms**
2. **Quantum Factoring Algorithms**
3. **Quantum Simulation Algorithms**:
4. Quantum Optimization Algorithms
5. **Quantum Machine Learning Algorithms**:
6. **Quantum Fourier Transform (QFT)**
7. Quantum Cryptography Algorithms

2. Quantum Factoring Algorithms:

- **Shor's Algorithm**: This algorithm can factor large integers exponentially faster than the best-known classical algorithms. It's particularly significant because it could potentially break widely used cryptographic systems like RSA, which relies on the difficulty of factoring large numbers.

3. Quantum Simulation Algorithms:

- **Quantum Simulations**: Quantum computers can simulate quantum systems much more efficiently than classical computers. Algorithms in this category are used to model molecular structures, chemical reactions, and other quantum systems, which has applications in materials science, chemistry, and drug discovery.

4. Quantum Optimization Algorithms:

- **Quantum Approximate Optimization Algorithm (QAOA)**: This algorithm is used to solve combinatorial optimization problems by finding approximate solutions more efficiently than classical methods.
- **Variational Quantum Eigensolver (VQE)**: VQE is used to find the ground state energy of a quantum system, which is crucial in quantum chemistry.

5. Quantum Machine Learning Algorithms

- **Quantum Support Vector Machine (QSVM)**: An adaptation of classical support vector machines, QSVMs leverage quantum computing to classify data points more efficiently.
- **Quantum Neural Networks (QNNs)**: These networks combine quantum computing with the principles of neural networks to potentially outperform classical neural networks in certain tasks.

6. Quantum Fourier Transform (QFT):

- **QFT** is a quantum version of the discrete Fourier transform and is a crucial component of several quantum algorithms, including Shor's algorithm. It is used for transforming quantum states into their frequency components, which is essential in solving problems related to periodicity and number theory.

7. Quantum Cryptography Algorithms:

- **Quantum Key Distribution (QKD)**: QKD uses quantum mechanics to create secure communication channels, ensuring that any eavesdropping attempts can be detected. The most famous QKD protocol is BB84.

1. Quantum Search Algorithms:

- **Grover's Algorithm**: One of the most famous quantum algorithms, Grover's algorithm provides a **quadratic speedup for unstructured** search problems.
- For example, if a classical algorithm needs N steps to search a list of N items, Grover's algorithm can do it in **sqrt(N)** steps.

Grover's Algorithm

- Grover's algorithm is a quantum algorithm that provides a significant speedup for searching an unsorted database or solving unstructured search problems. It is particularly known for its quadratic speedup compared to classical algorithms.

Overview of Grover's Algorithm

Given a function $f(x)$ that outputs 0 for all inputs except one, where it outputs 1, Grover's algorithm helps find the input x for which $f(x) = 1$ in $\sqrt{N}$ steps, where N is the total number of possible inputs.

Steps of Grover's Algorithm

1. **Initialization**:
   - Start with n qubits in the state $|0\rangle$, and apply the Hadamard gate to each qubit to create a superposition of all possible states.
   - This results in an equal superposition of all 2^n states, representing all possible solutions.

2. **Oracle**:
   - The oracle is a quantum subroutine that marks the correct solution by flipping the sign of its amplitude. It is usually represented as a black-box function $f(x)$.

Steps of Grover's Algorithm

3. **Amplitude Amplification (Grover Diffusion Operator)**:
   - The Grover diffusion operator amplifies the amplitude of the correct state and reduces the amplitudes of incorrect states. This step increases the probability of measuring the correct solution.

4. **Measurement**:
   - After repeating the Oracle and Amplitude Amplification steps $\sqrt{N}$ times, measure the quantum state. The result will be the correct solution with high probability.

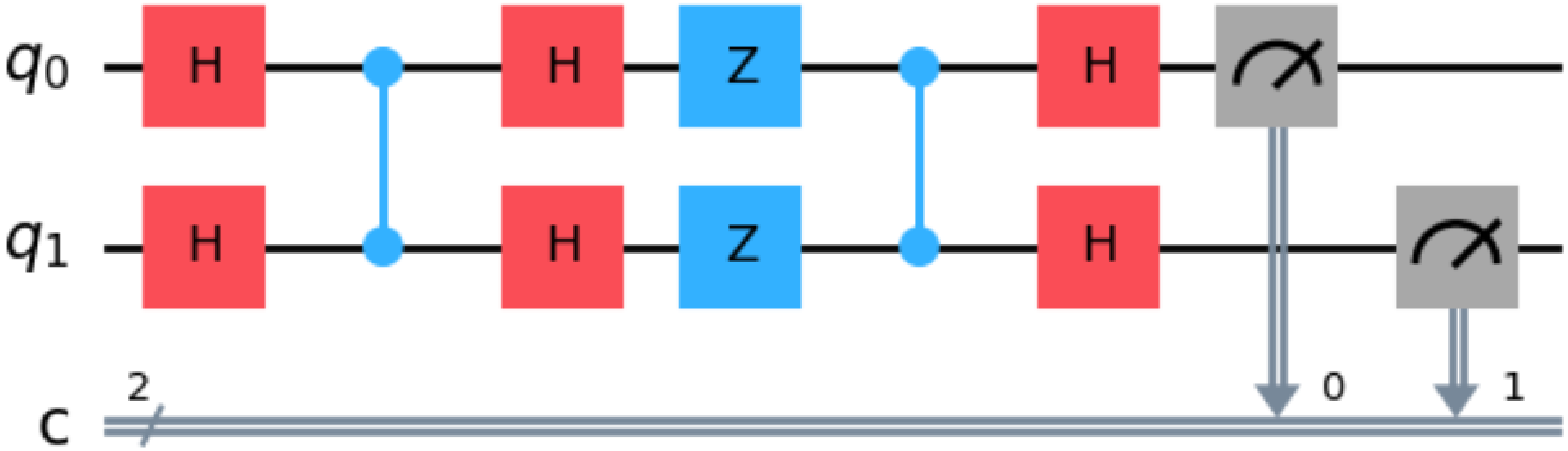
Example: Searching a Database

- Here's an implementation of Grover's Algorithm using Qiskit to search for a marked state in an unsorted database.
- The example circuit is designed to find the state $|11\rangle$ out of the four possible states $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ using Grover's algorithm.

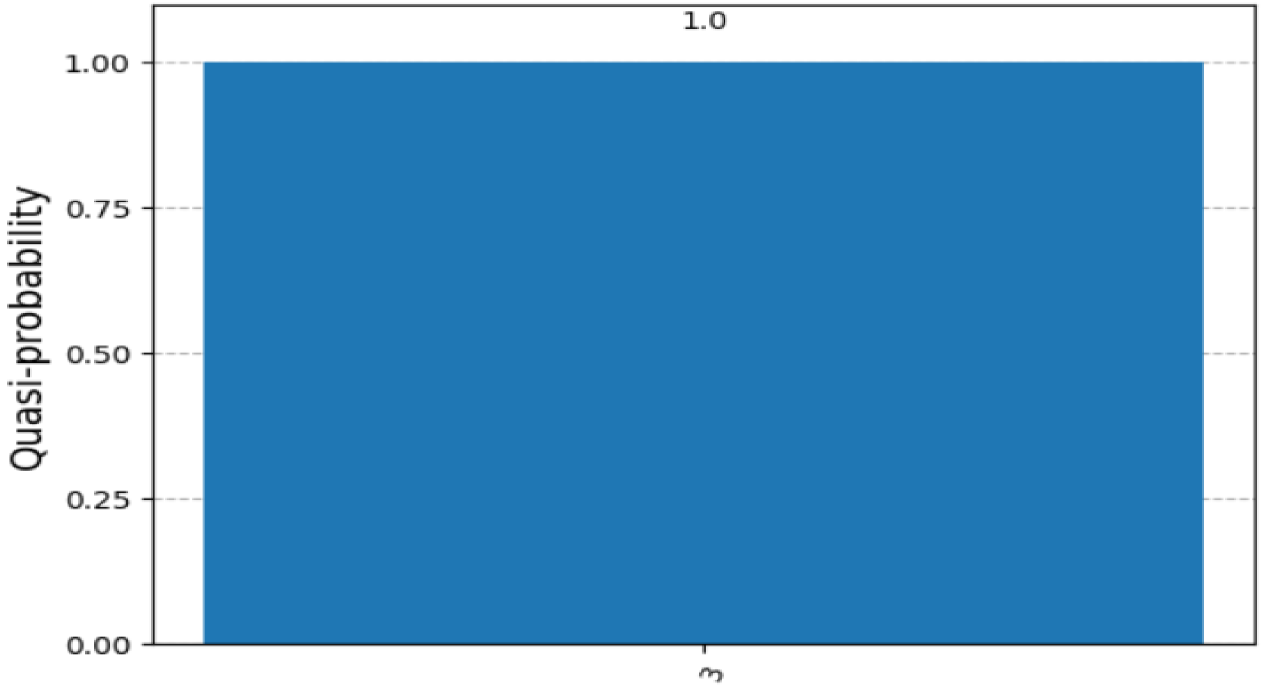
```python
from qiskit import QuantumCircuit, QuantumRegister, ClassicalRegister
from qiskit.primitives import Sampler
from qiskit.visualization import plot_histogram, circuit_drawer
from IPython.display import Image

# Quantum and Classical Registers
qreg = QuantumRegister(2, 'q')
creg = ClassicalRegister(2, 'c')
qc = QuantumCircuit(qreg, creg)

# Initialize Superposition
qc.h(qreg[0])
qc.h(qreg[1])

# Oracle for |11>
qc.cz(qreg[0], qreg[1])
```

```python
# Grover Diffusion Operator
qc.h(qreg[0])
qc.h(qreg[1])
qc.z(qreg[0])
qc.z(qreg[1])
qc.cz(qreg[0], qreg[1])
qc.h(qreg[0])
qc.h(qreg[1])

# Measurement
qc.measure(qreg, creg)
```

```python
# Save and display the circuit visualization
circuit_image = circuit_drawer(qc, output='mpl')
circuit_image.savefig('grovers_algorithm_circuit.png')
display(Image(filename='grovers_algorithm_circuit.png'))

# Simulate and Plot
sampler = Sampler()
job = sampler.run(circuits=qc)
result = job.result()
counts = result.quasi_dists[0]
plot_histogram(counts)
```



q_0
q_1
H
H
H
H
Z
Z
H
H
c
2
0
1



1.0
1.00
0.75
0.50
0.25
0.00
Quasi-probability

Quiz 5

Future Directions and Research Opportunities in Quantum Computing

Dr. Thyagaraju G S

Professor and HoD, Department of CSE,

SDM Institute Of Technology, Ujire-574240

Key areas

1. Quantum Algorithms
2. Quantum Error Correction
3. Quantum Hardware Development
4. Quantum Communication and Cryptography
5. Quantum Simulation
6. Quantum Machine Learning
7. Topological Quantum Computing
8. Quantum Metrology and Sensing
9. Quantum Ethics and Governance
10. Quantum Education and Workforce Development

1. Quantum Algorithms

- **New Algorithms Development**
  - Focus on optimization, machine learning, cryptography
- **Hybrid Classical-Quantum Algorithms**
  - Benefits of combining classical and quantum computing

2. Quantum Error Correction

- **Improving Error Correction Codes**
  - Importance of mitigating quantum noise
- **Fault-Tolerant Quantum Computing**
  - Steps toward reliable quantum computation

3. Quantum Hardware Development

- **Scalability of Quantum Processors**
  - Challenges and research focus
- **Exploring Qubit Technologies**
  - Superconducting, trapped ions, topological qubits
- **Quantum Interconnects**
  - Connecting multiple quantum processors

4. Quantum Communication and Cryptography

- **Quantum Internet**
  - Secure long-distance communication
- **Quantum Key Distribution (QKD)**
  - Integration with classical networks

5. Quantum Simulation

- **Simulating Complex Systems**
  - Applications in materials science, chemistry, drug discovery
- **Digital vs. Analog Simulations**
  - Comparison of approaches

6. Quantum Machine Learning

- **Quantum-enhanced Machine Learning**
  - Advantages for large datasets, complex problems
- **Variational Quantum Algorithms**
  - Optimization of machine learning models

7. Topological Quantum Computing

- **Topological Qubits**
  - Error resistance and stability
- **Majorana Fermions**
  - Research in creating protected qubits

8. Quantum Metrology and Sensing

- **High-Precision Measurements**
  - Leveraging quantum properties for accuracy
- **Applications in Medicine and Environmental Science**
  - Early disease detection, environmental monitoring

9. Quantum Ethics and Governance

- **Ethical Considerations**
  - Privacy, cybersecurity, and societal impact
- **Policy and Regulation**
  - Frameworks for responsible development

10. Quantum Education and Workforce Development

- **Expanding Educational Programs**
  - Preparing the next generation of quantum experts
- **Interdisciplinary Research**
  - Collaboration across physics, computer science, and engineering

Questions?

References

- ChatGPT
- Perplexity
- Claude
- Qiskit | IBM Quantum Computing
- Qubits: What is a Qubit and How Qubits Work? - MAKB Tech (makb183.com)
- Qubit (devopedia.org)
- Cryogenic measurements of semiconductor devices - NPL

References

- Google's Quantum Lab in California holds the future of computing (youtube.com)
- Quantum Computers, explained with MKBHD (youtube.com)
- The Map of Quantum Computing - Quantum Computing Explained (youtube.com)
- Quantum Computing Hardware - An Introduction (youtube.com)
- Decoded: How Does a Quantum Computer Work? (youtube.com)
- Quantum Computing: Algorithm, Programming and Hardware, an Introduction (youtube.com)
- Inside a Quantum Computer! with Andrea Morello (Part 1 of 2) (youtube.com)
- L2-1 Quantum Computing Hardware: An Overview (youtube.com)
- What Would a Quantum Internet Look Like? (youtube.com)

References

- Mapping the qubit state onto the Bloch Sphere (youtube.com)
- qiskit-tutorials/reference/tools/getting_started.ipynb at 66ab9961e5e25f8712819cd212351ee077a811ae · Qiskit/qiskit-tutorials · GitHub
- qiskit-tutorials/reference/tools/quantum_gates_and_linear_algebra.ipynb at 66ab9961e5e25f8712819cd212351ee077a811ae · Qiskit/qiskit-tutorials · GitHub
- textbook/notebooks/ch-states/single-qubit-gates.ipynb at main · Qiskit/textbook · GitHub
- textbook/notebooks/ch-prerequisites at main · Qiskit/textbook · GitHub

Thank you