Quantum Gates: The Building Blocks of Quantum Computing

Emir Kaan Özdemir

September 2024

Abstract

Quantum gates, pivotal to the functioning of quantum computers, represent the quantum counterpart to classical logic gates but operate on entirely different principles. Unlike classical gates that process binary bits, quantum gates manipulate qubits, enabling operations that harness quantum phenomena such as superposition and entanglement. These operations allow quantum computers to perform calculations that would be intractable for classical systems. This paper explores the significance of quantum gates in quantum algorithm development, their unique capabilities compared to classical gates, and their potential impact on various fields, including cryptography, optimization, and artificial intelligence. The content is structured to be accessible, but a solid understanding of advanced mathematical concepts may be necessary.

Contents

1 Introduction

Quantum gates are fundamental components in the realm of quantum computing, analogous to classical logic gates in traditional computers. However, their function and impact reach far beyond classical computation, enabling quantum computers to perform complex operations that are impossible for classical systems. This article explores the critical role quantum gates play in the development of quantum algorithms, their differences from classical gates, and their potential to revolutionize fields such as cryptography, optimization, and artificial intelligence. Although it is aimed to explain all of these in a simple and clear manner, more than a high school level of mathematics may be required to understand comfortably. Knowledge of complex numbers is required. Mathematical expressions and code snippets will be explained. The Q# programming language used for code blocks.

2 Necessary Mathematical Expressions

2.1 Matrices, Vectors and Scalars

2.1.1 Matrices(Matrix)

A matrix is set of numbers arranged in a rectangular grid. Here is a 2 by 2 matrix:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

The notation $A_{i,j}$ refers to the element in row i and column j of matrix A(all indices are 0-based). In the above example, $A_{0,1} = 2$.

An $n \times m$ matrix will have n rows and m columns:

$$\begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,m-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1,0} & x_{n-1,1} & \cdots & x_{n-1,m-1} \end{bmatrix}$$

Quantum computing uses complex-valued matrices: the elements of a matrix can be complex numbers. This, for example, is a valid complex-valued matrix:

$$\begin{bmatrix} 1 & i \\ -2i & 3+4i \end{bmatrix}$$

2.1.2 Vectors(Vector)

A vector is an $n \times 1$ matrix. Here, for example, is a 3×1 vector:

$$V = \begin{bmatrix} 1 \\ 2i \\ 3+4i \end{bmatrix}$$

Since vectors always have a width of 1, vector elements are sometimes written using only one index. In the above example, $V_0 = 1$ and $V_1 = 2i$ and $V_2 = 3+4i$.

2.1.3 Scalars(Scalar)

A 1×1 matrix is equivalent to a scalar:

$$\begin{bmatrix} 2 \end{bmatrix} = 2$$

2.2 Matrix Operations

2.2.1 Matrix Addition

Matrix addition is performed by adding the corresponding elements of the matrices together. Given two matrices A and B:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$B = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

The elements of the resulting matrix C are calculated using the formula:

$$C_{i,j} = A_{i,j} + B_{i,j}$$

Where $C_{i,j}$ represents the element in the ith row and jth column of matrix C.

The result is as follows:

$$C = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

2.2.2 Scalar Multiplication

The next matrix operation is scalar multiplication - multiplying the entire matrix by a scalar (real or complex number):

$$a \cdot \begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,m-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1,0} & x_{n-1,1} & \cdots & x_{n-1,m-1} \end{bmatrix} = \begin{bmatrix} a \cdot x_{0,0} & a \cdot x_{0,1} & \cdots & a \cdot x_{0,m-1} \\ a \cdot x_{1,0} & a \cdot x_{1,1} & \cdots & a \cdot x_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ a \cdot x_{n-1,0} & a \cdot x_{n-1,1} & \cdots & a \cdot x_{n-1,m-1} \end{bmatrix}$$

2.2.3 Matrix Multiplication

Matrix multiplication is a very important and somewhat unusual operation. The unusual thing about it is that neither its operands nor its output are the same size: an $n \times m$ matrix multiplied by an $m \times k$ matrix results in an $n \times k$ matrix. That is, for matrix multiplication to be applicable, the number of columns in the first matrix must be equal to the number of rows in the second matrix. Here's how matrix product is calculated: if you're calculating $AB = C$, then

$$C_{i,j} = A_{i,0} \cdot B_{0,j} + A_{i,1} \cdot B_{1,j} + \cdots + A_{i,m-1} \cdot B_{m-1,j} = \sum_{t=0}^{m-1} A_{i,t} \cdot B_{t,j}$$

Here's a small example:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 3 \\ 4 \cdot 1 + 5 \cdot 2 + 6 \cdot 3 \end{bmatrix} = \begin{bmatrix} 14 \\ 32 \end{bmatrix}$$

2.3 Transpose

The transpose operation, denoted as A^T, is essentially a reflection of the matrix across the diagonal: $A^T_{i,j} = A_{j,i}$.

Given an $n \times n$ matrix A, its transpose is the $n \times n$ matrix A^T, such that if:

$$A = \begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,m-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1,0} & x_{n-1,1} & \cdots & x_{n-1,m-1} \end{bmatrix}$$

then:

$$A^T = \begin{bmatrix} x_{0,0} & x_{1,0} & \cdots & x_{n-1,0} \\ x_{0,1} & x_{1,1} & \cdots & x_{n-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{0,m-1} & x_{1,m-1} & \cdots & x_{n-1,m-1} \end{bmatrix}$$

For Example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

2.4 Conjugate

The next important single-matrix operation is the matrix conjugate, denoted as $\overline{A}$. This operation makes sense only for complex-valued matrices; as the name might suggest, it involves taking the complex conjugate of every element of the matrix. In matrix form, if

$$A = \begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,m-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1,0} & x_{n-1,1} & \cdots & x_{n-1,m-1} \end{bmatrix}$$

Then:

$$\overline{A} = \begin{bmatrix} \overline{x}_{0,0} & \overline{x}_{0,1} & \cdots & \overline{x}_{0,m-1} \\ \overline{x}_{1,0} & \overline{x}_{1,1} & \cdots & \overline{x}_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{x}_{n-1,0} & \overline{x}_{n-1,1} & \cdots & \overline{x}_{n-1,m-1} \end{bmatrix}$$

For Example:

$$\overline{A} = \begin{bmatrix} -1 & -2 \\ -3 & -4 \end{bmatrix}$$

2.5 Adjoint(Conjugate Transpose)

The final important single-matrix operation is a combination of the previous two. The conjugate transpose, also called the adjoint of matrix A, is defined as $A^\dagger = \overline{(A^T)} = (\overline{A})^T$. A matrix is known as Hermitian or self-adjoint if it equals its own adjoint: $A = A^\dagger$. For example, the following matrix is Hermitian:

$$\begin{bmatrix} 1 & i \\ -i & 2 \end{bmatrix}$$

The adjoint of a matrix product can be calculated as follows:

$$(AB)^\dagger = B^\dagger A^\dagger$$

2.6 Special Matrices

2.6.1 Identity Matrix

An identity matrix, often denoted as I_n for an $n \times n$ matrix, is a square matrix with ones on the diagonal and zeros elsewhere. It is called the identity matrix because it is the multiplicative identity in matrix algebra. This means that for any $n \times n$ matrix A:

$$A \times I_n = I_n \times A = A$$

For example, the identity matrix of size 2×2 is:

$$I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

And for a 3×3 matrix:

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2.6.2 Unitary Matrices

Unitary matrices are very important for quantum computing. A matrix is unitary when it's invertible, and its inverse is equal to its adjoint: $U^{-1} = U^\dagger$. That is, an $n \times n$ square matrix U is unitary if and only if $UU^\dagger = U^\dagger U = I_n$.

2.7 Inner and Outer Products

2.7.1 Inner Product

The inner product, also known as the dot product in the context of vectors, is a fundamental concept in linear algebra. It provides a way to multiply two vectors, resulting in a scalar. The inner product of two vectors $\mathbf{u}$ and $\mathbf{v}$, denoted by $\mathbf{u} \cdot \mathbf{v}$ or $\langle \mathbf{u}, \mathbf{v} \rangle$, is calculated as follows:

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n \tag{1}$$

For example, given two vectors $\mathbf{u} = \begin{bmatrix} u_1 & u_2 & \cdots & u_n \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}$, the inner product is:

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n \tag{2}$$

In the case of complex vectors, the inner product involves the conjugate of the first vector:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^{n} \overline{u_i} v_i \tag{3}$$

The inner product is essential in defining orthogonality, norms, and angles between vectors, making it a cornerstone of vector space theory and applications in quantum computing.

2.7.2 Outer Product

The outer product of two vectors $\mathbf{u}$ and $\mathbf{v}$ is a matrix operation that results in a matrix. It is defined as:

$$\mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix} \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \cdots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \cdots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \cdots & u_m v_n \end{bmatrix} \tag{4}$$

For example, if:

$$\mathbf{u} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{5}$$

$$\mathbf{v} = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \tag{6}$$

Then the outer product $\mathbf{uv}^T$ is:

$$\mathbf{uv}^T = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 6 & 8 \end{bmatrix} \tag{7}$$

The outer product results in an $m \times n$ matrix, where each element is the product of the corresponding elements of the two vectors.

2.8 Tensor Product

The tensor product is a different way of multiplying matrices. Rather than multiplying rows by columns, the tensor product multiplies the second matrix by every element of the first matrix.

Given an $n \times n$ matrix A and $k \times l$ matrix B, their tensor product $A \otimes B$ is an $(n \cdot k) \times (m \cdot l)$ matrix defined as follows:

$$A \otimes B = \begin{bmatrix} A_{0,0} \cdot B & A_{0,1} \cdot B & \cdots & A_{0,m-1} \cdot B \\ A_{1,0} \cdot B & A_{1,1} \cdot B & \cdots & A_{1,m-1} \cdot B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-1,0} \cdot B & A_{n-1,1} \cdot B & \cdots & A_{n-1,m-1} \cdot B \end{bmatrix} =$$

$$\begin{bmatrix} A_{0,0} \cdot \begin{bmatrix} B_{0,0} & \cdots & B_{0,l-1} \\ \vdots & \ddots & \vdots \\ B_{k-1,0} & \cdots & b_{k-1,l-1} \end{bmatrix} & \cdots & A_{0,m-1} \cdot \begin{bmatrix} B_{0,0} & \cdots & B_{0,l-1} \\ \vdots & \ddots & \vdots \\ B_{k-1,0} & \cdots & B_{k-1,l-1} \end{bmatrix} \\ \vdots & \ddots & \vdots \\ A_{n-1,0} \cdot \begin{bmatrix} B_{0,0} & \cdots & B_{0,l-1} \\ \vdots & \ddots & \vdots \\ B_{k-1,0} & \cdots & B_{k-1,l-1} \end{bmatrix} & \cdots & A_{n-1,m-1} \cdot \begin{bmatrix} B_{0,0} & \cdots & B_{0,l-1} \\ \vdots & \ddots & \vdots \\ B_{k-1,0} & \cdots & B_{k-1,l-1} \end{bmatrix} \end{bmatrix} =$$

$$\begin{bmatrix} A_{0,0} \cdot B_{0,0} & \cdots & A_{0,0} \cdot B_{0,l-1} & \cdots & A_{0,m-1} \cdot B_{0,0} & \cdots & A_{0,m-1} \cdot B_{0,l-1} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ A_{0,0} \cdot B_{k-1,0} & \cdots & A_{0,0} \cdot B_{k-1,l-1} & \cdots & A_{0,m-1} \cdot B_{k-1,0} & \cdots & A_{0,m-1} \cdot B_{k-1,l-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ A_{n-1,0} \cdot B_{0,0} & \cdots & A_{n-1,0} \cdot B_{0,l-1} & \cdots & A_{n-1,m-1} \cdot B_{0,0} & \cdots & A_{n-1,m-1} \cdot B_{0,l-1} \\ \vdots & \ddots & \vdots & \dots & \vdots & \ddots & \vdots \\ A_{n-1,0} \cdot B_{k-1,0} & \cdots & A_{n-1,0} \cdot B_{k-1,l-1} & \cdots & A_{n-1,m-1} \cdot B_{k-1,0} & \cdots & A_{n-1,m-1} \cdot B_{k-1,l-1} \end{bmatrix}$$

3 Dirac Notation(Ket-Bra Representation)

Dirac notation is a shorthand notation that eases writing quantum states and computing linear algebra. In Dirac notation, a vector is denoted by a symbol called a ket. For example, a qubit in state 0 is represented by the ket $|0\rangle$, and a qubit in state 1 is represented by the ket $|1\rangle$.

Info: The reason for some things will be explained in the future, only the impressions were shown.

3.1 Ket

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

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

The kets $|0\rangle$ and $|1\rangle$ represent basis states, so they can be used to represent any other state:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha\,|0\rangle + \beta\,|1\rangle$$

Dirac notation isn't restricted to vectors 0 and 1; it can be used to represent any vector, similar to how variable names are used in algebra. For example, you can call the above state "ψ" and write it as:

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

3.1.1 Special Ket States

Several ket symbols have a generally accepted use, so you will see them often.

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

$$|-\rangle = \frac{1}{\sqrt{2}}\big(\,|0\rangle - |1\rangle\,\big)$$

$$|i\rangle = \frac{1}{\sqrt{2}}\big(\,|0\rangle + i\,|1\rangle\,\big)$$

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

3.2 Bra

Although bras are used less than kets, learning them is necessary in terms of formulation. Bras are also called row vectors. $\langle 0|$ is read as "Bra 0". $\langle 1|$ is read as "Bra 1".

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

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

4 The Qubit

4.1 The Concept of Qubit

The basic building block of a classical computer is the bit - a single memory cell that is either in state 0 or in state 1. Similarly, the basic building block of a quantum computer is the quantum bit, or qubit. Like the classical bit, a qubit can be in state 0 or in state 1. Unlike the classical bit, however, the qubit isn't limited to just those two states - it may also be in a combination, or superposition of those states.

4.1.1 Representation

The state of a qubit is represented by a complex vector of size 2:

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

Here α and β are complex numbers. α represents how "close" the qubit is to state 0, and β represents how "close" the qubit is to state 1. This vector is normalized if:

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

α and β are known as the probability amplitudes of states 0 and 1, respectively.

4.2 Qubit Data Type in Q#

This part is shown to understand the logic of using data qubit. The some syntax rules of the Q# language will be skipped.

```
namespace Article{
open Microsoft.Quantum.Diagnostics;
@EntryPoint()
operation HelloQubitDataType() : Unit {
    use q = Qubit();
    Message("I defined a single qubit in |0> state.");
    DumpMachine();
    // This line prints out the state of the quantum computer.
    // Since only one qubit is allocated, only its state is printed.
}
}
```

5 Multi-Qubit Systems

A multi-qubit system is a collection of multiple qubits, treated as a single system. Let's start by examining a system of two classical bits. Each bit can be in two states: 0 and 1. Therefore, a system of two bits can be in four different states: 00, 01, 10, and 11. Generally, a system of N classical bits can be in any of the 2^N states. A system of N qubits can also be in any of the 2^N classical states, but, unlike the classical bits, it can also be in a superposition of all these states. Similarly to single-qubit systems, a state of an N-qubit system can be represented as a complex vector of size 2^N:

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{2^N-1} \end{bmatrix}$$

5.1 Seperable States

This is where tensor multiplication comes in handy. Sometimes the global state of a multi-qubit system can be separated into the states of individual qubits or subsystems. To do this, you would express the vector state of the global system as a tensor product of the vectors representing each individual qubit/subsystem. Here is an example of a two-qubit state:

Examples:

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

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

$$\begin{bmatrix}\frac{1}{\sqrt{2}}\\0\\\frac{1}{\sqrt{2}}\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \frac{1}{\sqrt{2}}\big(\,|0\rangle + |1\rangle\,\big) = |+\rangle$$

The third states first qubit is in a superposition. The probability of two basis are both equal. If we check that the absolute values of the squares of the coefficients of the bases are equal to 1, we realize that the system is normalized.

5.2 Entanglement

Sometimes, quantum states cannot be written as individual qubit states. Quantum systems that are not separable are called entangled systems. If a state can be written as the product state of the individual subsystems, that state is not entangled.

Entanglement is a quantum correlation, which is very different from classical correlations. In entanglement, the state of the subsystems isn't determined, and you can talk only about the probabilities associated with the outcomes. The global system must be considered as one.

Entanglement is a huge part of what makes quantum computing so powerful. It allows us to link the qubits so that they stop behaving like individuals and start behaving like a large, more complex system. In entangled systems, measuring one of the qubits modifies the state of the other qubits, and tells us something about their state. For example, consider two qubits A and B in superposition such that the state of the global system is

$$|\psi\rangle_{AB} = \frac{1}{\sqrt{2}}\,|00\rangle + \frac{1}{\sqrt{2}}\,|11\rangle$$

In such a state, only two outcomes are possible when you measure the state of both qubits in the standard basis: $|00\rangle$ and $|11\rangle$. Notice that each outcome has the same probability of $\frac{1}{2}$. There's zero probability of obtaining $|01\rangle$ and $|10\rangle$. If you measure the first qubit and you get that it is in $|0\rangle$ state, then you can be positive that the second qubit is also in $|0\rangle$ state, even without measuring it. The measurement outcomes are correlated, and the qubits are entangled.

5.3 Defining Multi-Qubit Systems in Q#

It has already been explained that systems can have more than one qubit. However, for a more concrete demonstration:

```
namespace Article{
open Microsoft.Quantum.Diagnostics;
@EntryPoint()
operation DefiningMultiQubitSystems() : Unit{
// This allocates an array of 2 qubits, each of them in state |0⟩.
// The overall state of the system is |00⟩.
use qs = Qubit[2];
Message("States of the qubits");
DumpMachine();
}
}
```

6 Quantum Gates

6.1 Single-Qubit Gates

Quantum gates are the fundamental building blocks of quantum computing, similar to how classical logic gates are the basis of classical computing. However, quantum gates operate on quantum bits, or qubits, which can exist in multiple states simultaneously thanks to the principles of quantum mechanics, such as superposition and entanglement. This section was written to explain the working principles of the gates. Only the most used ones will be mentioned. **All gates are a unitary matrix.**

$$UU^\dagger = U^\dagger U = I_n$$

6.1.1 Applying Quantum Gates to State Vectors

Quantum gates represented as matrices. We already said that matrices are made up of complex numbers.(Remember $3 = 3 + 0i$) Applying quantum gates to state vectors means: multiplying the matrices representing the gates by the state vectors.

6.1.2 Pauli Gates

X Gate The X gate also known as the NOT gate. It reverses the state.

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

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

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

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

When gates are applied to vectors, the result is found like this and this process will not be shown again.

And this is the proof of the rule: all gates are a unitary matrix.

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

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

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

$$X^\dagger X = \begin{pmatrix} 0 \cdot 0 + 1 \cdot 1 & 0 \cdot 1 + 1 \cdot 0 \\ 1 \cdot 0 + 0 \cdot 1 & 1 \cdot 1 + 0 \cdot 0 \end{pmatrix}$$

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

$$X^\dagger X = I$$

Y Gate

$$Y \Rightarrow \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i(|1\rangle\langle 0| - |0\rangle\langle 1|)$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \Rightarrow Y|\psi\rangle = i\big(\alpha|1\rangle - \beta|0\rangle\big)$$

Z Gate The Z gate is sometimes referred to as the phase flip gate.

$$Z \Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \Rightarrow Z|\psi\rangle = \alpha|0\rangle - \beta|1\rangle$$

6.1.3 Hadamard Gate

Hadamard gate is one of the most important gates. It uses to make a superposition and this superpositions are already familiar: $|+\rangle$ and $|-\rangle$.

$$H \Rightarrow \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = |0\rangle\langle +| + |1\rangle\langle -|$$

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \Rightarrow H|\psi\rangle = \alpha|+\rangle + \beta|-\rangle = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle$$

For Example:

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

The process created a superposition where the probabilities are $\frac{1}{\sqrt{2}}$.

$$H|0\rangle = |+\rangle$$

It can be checked that it is normalized:

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

$|-\rangle$ can be obtained by performing similar operations.

$$H|1\rangle = |-\rangle$$

6.2 Applying Gates to State Vectors in Q#

```
namespace Article {
    open Microsoft.Quantum.Diagnostics;

    @EntryPoint()
    operation ApplyingGates(q : Qubit) : Unit {
        Message("Now the qubit is in |0> state");
        DumpMachine();

        X(q);
        Message("Now it is |1>");
        DumpMachine();

        Y(q);
        Message("Now it is -i|0>");
        DumpMachine();

        Reset(q); // This makes it |0>
        Z(q); // This makes no change, as the qubit is already in |0>
        Message("Now it is |0> again after Z(q)");
        DumpMachine();
        X(q);
        Z(q);
        Message("Now it is -|1>");
        DumpMachine();
        Reset(q); // |0>

        // Apply Hadamard gate
        H(q);
        Message("Now it is in a superposition -- |+>");
        DumpMachine();

        Reset(q);
    }
}
```

6.3 Multi-Qubit Gates

Single-qubit gates, as the name suggests, apply only to single qubits. But that does not mean that single-qubit gates cannot use with multi-qubit systems.

Remember:

$$|10\rangle = |1\rangle \otimes |0\rangle$$

For example in this equation if Hadamard gate applied to first qubit:

$$\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle)$$

This operations can be done with other systems which has more qubits and the other gates such as Pauli gates or phase shift gates(did not shown).

6.3.1 The Most Known Multi-Qubit Gates

SWAP Gate

$$SWAP \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = |00\rangle\langle 00| + |01\rangle\langle 10| + |10\rangle\langle 01| + |11\rangle\langle 11|$$

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \Rightarrow SWAP|\psi\rangle = \alpha|00\rangle + \gamma|01\rangle + \beta|10\rangle + \delta|11\rangle$$

$$\begin{aligned} SWAP|00\rangle &= |00\rangle \\ SWAP|10\rangle &= |01\rangle \\ SWAP|01\rangle &= |10\rangle \\ SWAP|11\rangle &= |11\rangle \end{aligned}$$

6.3.2 Controlled Gates

Controlled gates can only use in the systems which have 2 or more qubits, because controlled gates takes a qubit as the control qubit and the other qubit as the target qubit. Controlled gates can take more than 1 qubit as the control and target qubit and that will be shown.

The Most Known Controlled Gates

CNOT Gate CNOT known as the controlled-NOT gate and it is equal to controlled-X gate. It flips the target qubit if the control qubit is in $|1\rangle$ state. Usually the first qubit is the control, and the second qubit is the target.

$$CNOT \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 11| + |11\rangle\langle 10|$$

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \Rightarrow CNOT|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \delta|10\rangle + \gamma|11\rangle$$

For Example:

$$\begin{aligned} CNOT|10\rangle &= |11\rangle \\ CNOT|01\rangle &= |01\rangle \end{aligned}$$

CZ Gate The CZ("controlled-Z") gate is a two-qubit gate, with one qubit referred to as the control qubit, and the other as the target qubit. For the CZ gate it does not matter which qubit is the control and target qubit.

$$CZ \Rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} = |00\rangle \langle 00| + |01\rangle \langle 01| + |10\rangle \langle 10| + |11\rangle \langle -11|$$

$$|\psi\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle \Rightarrow CZ |\psi\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle - \delta |11\rangle$$

$$CZ |00\rangle = |00\rangle$$

$$CZ |01\rangle = |01\rangle$$

$$CZ |10\rangle = |10\rangle$$

$$CZ |11\rangle = - |11\rangle$$

6.3.3 Gates Can Have Multiple Control and/or Target Qubits

This section does not contribute much to the article, that is, it will only be explained to create an understanding and no examples will be given. Quantum gates are not limited to operating on just one or two qubits; many gates can work with multiple control and/or target qubits. These types of gates are used to perform complex quantum operations and algorithms. For example, **controlled gates** operate with one or more control qubits and perform an operation on a specific target qubit. The **controlled-Toffoli** (CCNOT) gate includes two control qubits and one target qubit; if both control qubits are in the $|1\rangle$ state, a NOT gate is applied to the target qubit. The **controlled-SWAP** (Fredkin) gate, on the other hand, includes one control qubit and two target qubits; when the control qubit is in the $|1\rangle$ state, it swaps the states of the two target qubits. Additionally, **multi-target gates** are available, which perform the same operation on multiple target qubits. For instance, **controlled-unitary** gates apply a specific unitary transformation to multiple target qubits using one or more control qubits. These multi-control and multi-target qubits provide a broader and more flexible range of operations in quantum computing, enabling the design of more complex and powerful quantum algorithms and protocols.

7 Why Quantum Gates so Important?

The importance of quantum gates was explained with concrete mathematical proofs and calculations. This section will explain quantum qates' importance in quantum computing using thought experiments and examples. Also applications of quantum computers will be mentioned.

7.1 Understanding Superposition with Schrödinger's Cat

An English proverb states, "A cat has nine lives. For three he plays, for three he strays, and for the last three he stays." In the quantum world, however, objects can be in a superposition of states simultaneously. Therefore, a quantum cat could exist in a superposition of playing, straying, and staying all at once.[1]

Schrödinger's cat is a famous thought experiment that illustrates the concept of superposition in quantum mechanics. Superposition refers to a quantum system's ability to exist in multiple states simultaneously, with these states remaining undefined until an observation is made.

In this thought experiment, a cat is placed inside a sealed box along with a radioactive atom, a Geiger counter, a vial of poison, and a mechanism that releases the poison if the Geiger counter detects radiation. The radioactive atom has a 50% chance of decaying within an hour, triggering the release of the poison and killing the cat, and a 50% chance of not decaying, allowing the cat to remain alive.

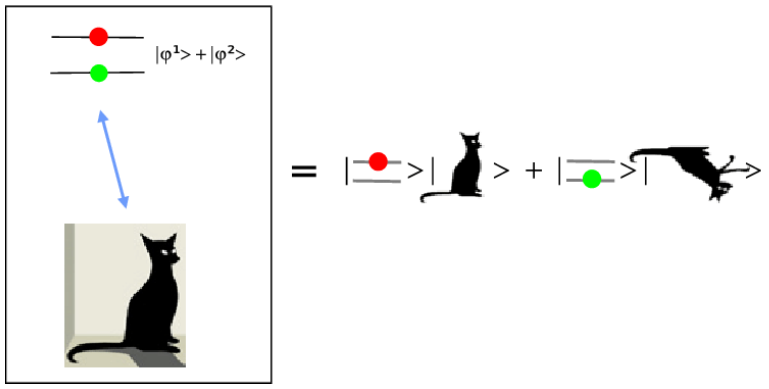

Figure 1: Schrödinger's Cat's Probabilities Representation[2]

Before the box is opened and the system is observed, the atom exists in a superposition of both decayed and undecayed states. According to the principles of quantum mechanics, this superposition extends to the entire system, meaning that the cat is simultaneously alive and dead until observed. Only when the box is opened and the system is measured does the cat's state collapse into one of the two possibilities—alive or dead.

This thought experiment highlights the peculiar nature of quantum superposition, where particles can exist in multiple states at once, a phenomenon not observed in the classical world. The Schrödinger cat is therefore a metaphor to illustrate the counterintuitive aspects of quantum mechanics, particularly the role of measurement in determining the state of a quantum system.

7.2 The Role of Superposition in Quantum Computing

In a classical system, a state can exist only or not exist. It is about living or dying, sleeping or being awake, going to work or not, etc. In other words, in the world, which is a classical system, something can either exist or not exist. This situation is the same as in the classical computers. However, in quantum computers, superposition makes both situations possible at the same time. The best example that explains the contribution of this situation is the labyrinth example. In classical computers, finding the exit of a maze involves progressing step by step through a set of possibilities. The computer tries one path at a time and if it encounters a dead end, it backtracks and tries the next possible path. This process can take a long time if the maze is complex and branching because the classical computer must test each path sequentially. This represents the limitation of a classical computer that can operate only on one possibility at a time.

However, in **quantum computers**, things work very differently. Thanks to quantum superposition, a qubit can exist in multiple states at once. This means that a quantum computer can explore multiple paths of the maze simultaneously. In superposition, the quantum computer can attempt both correct and incorrect paths at the same time. Therefore, it can effectively search the entire maze in parallel, dramatically reducing the time required to find the exit.

Difference Between Classical and Quantum Computers

- **Classical Computer**: The classical computer tries each path in the maze one by one. At each step, it selects a path, and if it reaches a dead end, it backtracks and tries the next possibility. This process is sequential.
- **Quantum Computer**: Due to the superposition property, a quantum computer can test all paths simultaneously. This allows it to explore the maze in parallel and find the exit much faster.

This example demonstrates how superposition allows quantum computers to perform **parallel computations**, unlike the **serial processing** of classical computers. While a classical computer must try every possibility one after the other, a quantum computer can explore multiple possibilities at once, leading to much faster problem solving capabilities.

7.3 Applications of Quantum Computers

Quantum computers hold the potential to revolutionize various fields through their unique computational abilities. Here are some key applications:

- **Cryptography:** Quantum computers can potentially break widely used encryption schemes, such as RSA, by efficiently factoring large numbers. However, they also enable the development of quantum-resistant cryptographic methods, which are essential for securing future communications.
- **Optimization:** Many real-world problems, such as route planning, scheduling, and resource allocation, involve finding the optimal solution among many possibilities. Quantum computers can solve these problems more efficiently using algorithms like the Quantum Approximate Optimization Algorithm (QAOA).
- **Drug Discovery:** Quantum computers can simulate molecular structures and interactions with high precision, which is valuable for discovering new drugs and materials. This capability could accelerate the development of treatments for complex diseases.
- **Machine Learning:** Quantum machine learning algorithms can process and analyze large datasets more efficiently than classical algorithms. Techniques such as quantum support vector machines and quantum neural networks offer the potential to enhance pattern recognition and data analysis.
- **Material Science:** Quantum computers can model complex materials and chemical reactions, enabling the discovery of new materials with specific properties. This has implications for developing advanced materials for electronics, energy storage, and more.
- **Security Simulations:** The vehicles that are manufactured (planes, cars, trains, etc.) need to be tested for safety. However, companies have calculated that doing these on computers is more expensive than doing them physically. If they could use quantum computers, they would have much shorter calculation times with more parameters and much lower costs.

These are just some of the uses that are currently being anticipated. But it is difficult to predict what the future holds. For example, when the first classic computers were produced, it was not thought that they could play thousands of games.

Quantum gates are pivotal in achieving the remarkable performance of quantum computers in various applications. They enable the manipulation and control of qubits, the fundamental units of quantum information, through unitary transformations. This capability is crucial for implementing quantum algorithms that take advantage of the unique properties of quantum mechanics, such as superposition and entanglement. Quantum gates facilitate the creation and maintenance of these quantum states, allowing quantum computers to perform complex calculations that classical computers struggle with. By orchestrating sequences of quantum gates, quantum algorithms can efficiently tackle problems in cryptography, optimization, drug discovery, machine learning, and material science. The efficiency and power of quantum computers in these domains are fundamentally rooted in the precise operation of quantum gates, which ensure that quantum mechanical principles are accurately harnessed to achieve unparalleled computational performance.

8 References and Citations

- [**1**]: https://physicsworld.com/a/schrodingers-cat-makes-a-better-qubit-in-critical-regim
- [**2**]: https://www.researchgate.net/figure/Illustration-of-the-Schroedingers-cat-paradox-a fig2_264550002

9 Contact

Feel free to contact for suggestions and questions.

- **LinkedIn** https://www.linkedin.com/in/emir-kaan-%C3%B6zdemir-8016442b9/
- **GitHub** https://github.com/emirkaanozdemr