Brief Introduction to Quantum Computing for Undergraduate Students: Lecture One

Li Chen
University of the District of Columbia

Abstract

This article provides an introductory overview of quantum computing for college students through intuitive examples. We explore key concepts, starting with the basic principles of quantum mechanics. The discussion covers essential elements such as Qubits, superposition, entanglement, and introduces quantum gates. The article concludes by examining the first quantum algorithm: the solution to Deutsch's Problem.

1 Introduction

The quantum computer uses the principles of quantum mechanics to perform computations. The related field is called quantum computing. It can be much more efficiently over classical computers when dealing with certain types of calculations. Quantum computers utilize quantum bits or qubits, which have the property of existing multiple states simultaneously, to leverage parallel-computation-like computations at quantum level. During a calculation, Qubits usually hold the simultaneous multiple states that is called superposition during calculations; two or more qubits can also make connections through so called entanglements, so that multiple qubits can work together as a bounded group. The results of a quantum computation usually obtained by quantum measurement that will make a superposition disappear to locate to a single state probabilistically.

In this article, we provide an introductory overview of quantum computing for college students through intuitive examples. We first explain the basic principles of quantum mechanics such as wave-particle duality. Then we extend the discussion to Qubits, superposition, entanglement, and introduces quantum gates. The article concludes by exploring the first quantum algorithm: the solution to Deutsch's Problem.

The first important concept of quantum computing is **Qubits**. In popular computers or classical computing, a bit can either represent 0 or 1. We can refer them to two states, State 0 and State 1. A qubit abbreviated from a quantum bit, can stay in a **Superposition** with both State 0 and State 1 simultaneously. This property enables the possibility of parallel-like computation when multiple qubits are involved in a system. In other words, the superposition empowers quantum computers to explore or examine multiple possible solutions to a problem at the same time.

Secondly, **Entanglement** among qubits means that a certain state of one qubit can be directly related to a particular state of another qubit, i.e. these two (or more) states are correlated regardless of the physical distance between them. Entanglements can significantly reduce the occurrence of meaningless outcomes in problem-solving, "fostering a heightened level of correlation between qubits and thereby facilitating more efficient information processing."

Similar to logic gates used in classical computers which perform logical operations on one or more binary inputs, **Quantum Gates** in quantum computers are used to manipulate qubits and perform quantum operations. However, Quantum gates , much different from classical logic gates, take advantage of superposition that explores multiple solutions at the same time—based on the properties of quantum mechanics.

Quantum Algorithms, at the center of quantum computing, serve an essential role to solve different problems using quantum computers. A quantum algorithm can be implemented in multiple stage or multiple layers by quantum gates. For instances, Shor's algorithm was developed to find factors of an

large integer number. and Grover's algorithm is to find a specific element in a database. Other quantum algorithms such as quantum annealing employ the different architectures in quantum computing. As we mentioned in above all of these algorithm use superposition for exploiting the capabilities of parallel computation to solve specific problems much more efficiently comparing to classical algorithms.

2 Introduction: What is Quantum and Quantum Mechanics

In science, matter is the substance that has mass and occupies space. It is composed of molecules, and these molecules are formed by atoms. As we know that the atom consists of electrons, protons and neutrons. See Fig. 1. [1] There are still more layers that physicists can decompose protons and neutrons into basic or smallest particles, these particles are called quantum.

A quantum, or quanta, represents the minimum amount of any physical entity. For instance, an electron or photon can be regarded as a quantum, as they cannot be subdivided into smaller pieces.

At the quantum level, typically on the order of 10^{-15} to 10^{-31}mm, the quantum (a photon or electron) displays wave-particle duality. This means that it exhibits characteristics of both a wave and a particle. This phenomenon has been experimentally confirmed over the past century.

One can think about a particle as a tiny dot or a tiny ball. Despite being throw (shot) out individually, such a tiny object exhibits wave-like properties. Its particle nature meaning that moves as a straightforward tiny ball, only becomes apparent when it is measured or observed. Quantum computing needs to use both of these natures.

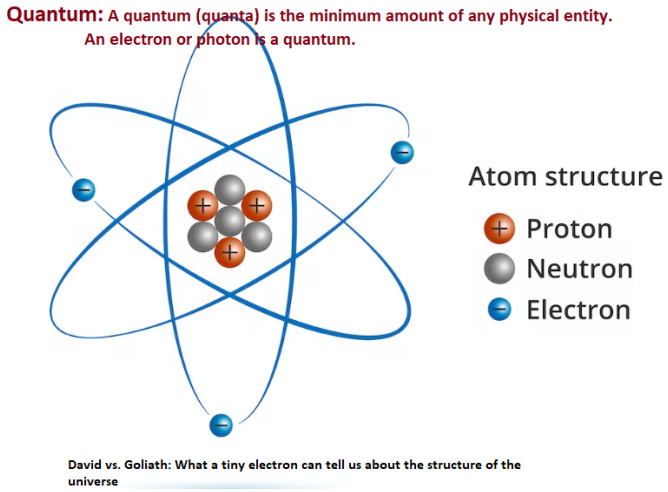


Figure 1: What is a quantum: the structure of an atom where electrons are quantum.

In theory, a quantum element, such as an electron, at a specific time is referred to a quantum **state** that can be described by the Schrödinger equation:

$$i\hbar \frac{d}{dt}\psi(t) = H\psi(t). \tag{1}$$

So $\psi(t)$ is the solution for the state at time t. We do not need to worry about how we get $\psi(t)$ analytically from the above equation in learning quantum computing. We only need to ensure that $\psi(t)$ is a wave as follows:

$$\psi(t) = e^{-iHt/\hbar}\psi(0). \tag{2}$$

The expression $e^{-iHt/\hbar}$ represents the time evolution operator in quantum mechanics and $\psi(0)$ is an initial state [1].

[1]Transferring a state to another in a quantum system of quantum mechanics must be unitary which we will define in next article. Note that, the exponential function e^A where A is an operator or a matrix, can be expressed by the Taylor

We observe that the solution of the Schrödinger equation appears as a wave. Consequently, it can be decomposed into two waves; simultaneously, two waves can be combined into a single wave. This theory aligns with the earlier discussion on superposition.

When dealing with two or more particles, such as two electrons, the Schrödinger equation remains valid, but obtaining the solution analytically becomes considerably challenging. Nonetheless, we can still treat the solution as a wave and interpret it as the superposition of combined waves originating from individual particles. Fortunately, in quantum computing, we only handle very simple cases that are not closely tied to the original Schrödinger equation.

There is a situation a superposition of two particles (two electrons), a wave, could not be separated into two waves. These two particles are called entangled as we mentioned before. Even thought that the two are separated to millions of miles in distance, the two particles (its superposition) will act as one. The change of one particle (e.g. measurement) will determine the state of the second particle.

3 An Intuitive Perspective on Quantum Computing

Quantum computing utilizes the principle of wave-particle duality to perform calculations akin to parallel computation at the quantum level. This is because the interference of two waves in physics allows "parallel processing" in computing.

Consider, for instance, a person tossing two stones into a pond at different nearby locations (Fig. 2). After a few seconds, the resulting wave is an interference of the two waves, and every few seconds, the combined wave transforms into a new shape (a new state). Consequently, the "calculation" from an old shape (old state) to a new shape (a new state) occurs in parallel computation alike. The combined wave can be conceptualized as a super-state, termed superposition in physics. This interpretation provides a tangible and popular explanation of the calculation phenomenon between two waves.

In contrast to classical computation, where the new "picture" of the combined wave requires calculating each amplitude value at individual physical points, this approach is sequential computation in classical computer algorithm analysis.

In other words, consider the scenario when two waves are combined, a process known as wave interference. How many computational steps does a computer need to complete this when both waves are arbitrary or random? This process is contingent on sampling—how many sample points we extract from the two waves. The time complexity is at least linear ($O(n)$).

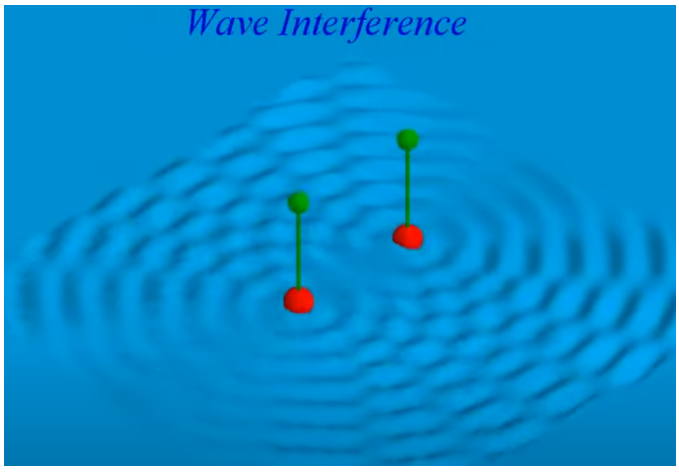


Figure 2: Interference of two water waves [2].

In quantum mechanics, leveraging this property for computation is the essence of quantum computing. It was proposed by R. Feynman in 1982. In the physical world, the interference of two waves typically occurs within a unit of time, determined by the speed of motion in experiments. This differs

series expansion:

$$e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!}.$$

from digital computers, which rely on sampling and addition of each samples. The following example provides more detailed insights into the interference of waves (Fig. 3):

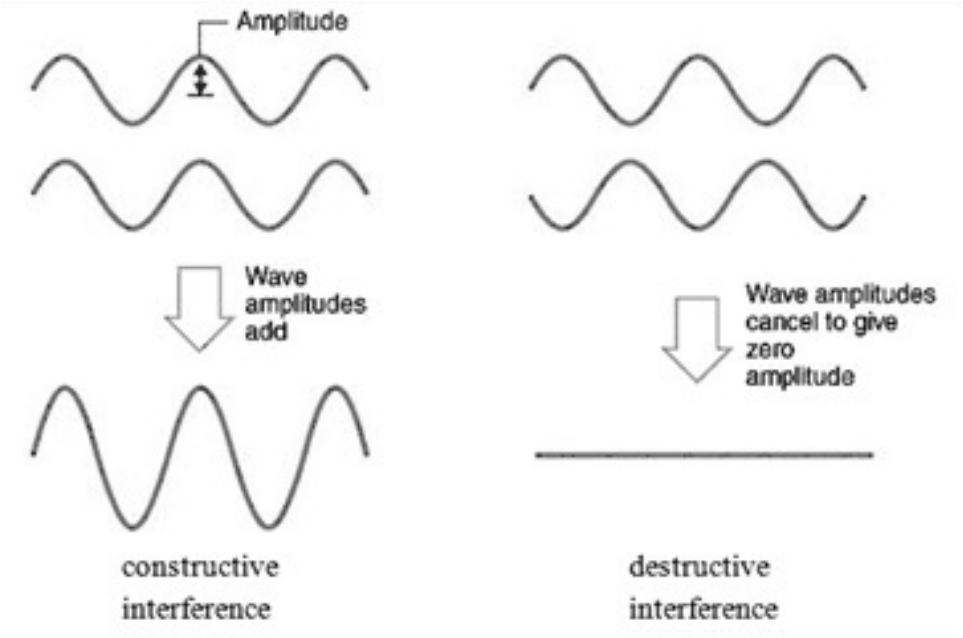


Figure 3: Accumulation and cancellation of two waves in interference. [3]

How to Achieve Quantum Computing?

Let us describe why it is possible that an exponential number of computations in classical computers becomes feasible (polynomial time) in quantum computing:

(a) At the quantum scale, the state of a particle (such as an electron) behaves like a wave.

(b) The interference of two waves (quantum computation) is analogous to the merging of two waves into one (in parallel), resulting in a single computational process. This phenomenon is known as quantum coherence. Quantum computations involve the superposition containing exponential numbers of (basic) states.

(c) A wave can represent the outcome of combining multiple waves, a state known as superposition.

(d) To extract specific information from a combined wave, we entangle it with another wave (a particle state). This entanglement ensures that changes in one will directly impact the other. This process is referred to as quantum decoherence.

(e) Measurement comes into play when we observe the entangled wave, causing it to collapse into a definite state. This state is the result we expected and wanted, even obtained with probabilities. This collapse, in turn, influences the state of the original wave, illustrating the complexities of quantum computing.

These principles form the foundation of quantum computing, allowing for the execution of an exponential number of calculations in a parallel and interconnected manner.

4 Basic Elements of Quantum Computing: Qubits

The foundational element in quantum computing is known as the qubit. To better understand qubits, let's revisit the concept of bits in digital computers.

In digital computing, electronic circuits are designed to recognize only 0's and 1's at the lowest electronic level. For instance, 0's and 1's may represent low-voltage or high-voltage. Each of these binary values is termed a bit, capable of representing a single zero or one. Multiple bits can collectively form a $\{0, 1\}$-string or a binary number. We can view the value 0 stored in a bit as a state, as well as 1 as another state of the bit.

Similarly, in quantum computing, we work with quantum bits, or qubits. Physicists have observed that electrons exhibit a spin of their own (or at least demonstrate properties similar to self-spinning objects). Additionally, electrons can exist in a wave or particle format. Consequently, **spinning-up** can be designated as one state, and **spinning-down** as another. Thus, we can define two states: $<$ **State0** $>$ and $<$ **State1** $>$.

Figure 4: A Quantum Bit: the **spin-up** state and the **spin-down** state [4].

A significant difference from a classical bit is that a spinning electron can rotate at any angle, not restricted to UP or DOWN. The spinning at an angle constitutes a new state, and this state can undergo continuous changes. Mathematically, an arbitrary spin is a combination of two fundamental states with specific coefficients: $<$ **State0** $>$ and $<$ **State1** $>$, (considering $<$ **State0** $>$ and $<$ **State1** $>$ as two vectors. Consequently,) any state of an electron's spinning can be expressed using the two basic states: **Spin-Up** and **Spin-Down**.

$$\Psi = a < State0 > + b < State1 > \tag{3}$$

where a and b are two real numbers (or complex numbers).

In quantum mechanics, a crucial requirement is that $|a|^2+|b|^2 = 1$. This constitutes the probabilistic interpretation of quantum mechanics concerning the observation (measurement) of quantum particles.

This principle is another fundamental property of quantum mechanics, asserting that upon measurement, the spinning particle will collapse into either $<$ **State0** $>$ or $<$ **State1** $>$. The collapse is not definite; Ψ will collapse to $<$ **State0** $>$ with a probability of $|a|^2$ and to $<$ **State1** $>$ with a probability of $|b|^2$. Ensuring a total probability of 1, we have $|a|^2 + |b|^2 = 1$.

If we can establish a mathematical representation for spin-up and spin-down, the electron spinning at an angle becomes a combination of these two states. In simpler terms, the electron spinning at an angle is a superposition of the states $SpinUp$ and $SpinDown$. An electron in this type of superposition is referred to as a **QuBit**. Quantum computers are constructed with many QuBits.

Quantum computing opens up the possibility for a quantum bit to represent both a zero and a one simultaneously. When 10 or 100 qubits are organized within a system, the computing system can effectively represent 2^{10} or 2^{100} states simultaneously. This highlights the remarkable power of quantum computation, a concept introduced by David Deutsch in 1985.

In the upcoming subsections, we will delve into a method that can mathematically represent such a general quantum state in its superposition, i.e. a qubit.

4.1 How to Represent a Qubit Mathematically

Now, our task is to find a suitable mathematical representation for a quantum state, particularly a qubit. How should we represent it: as a (complex) number, a vector, or a matrix?

As we discussed earlier, there are two fundamental states of an electron spinning. But how should we represent them? Using just $< State0 >$ or $< State1 >$ won't suffice for further computations. The simplest approach is to represent them as 0 or 1, that are scalar values. However, using scalar values alone poses challenges when we need to handle addition and multiplication, crucial for describing superposition that needs to contain two states at the same time. Hence, we require a more sophisticated method for representing a state.

In artificial intelligence especially in pattern recognition, we commonly use a vector (a feature vector) to represent a state for the real world problems. Therefore, let's explore the possibility of using

a vector for a qubit. This is also a logical choice for the next step. Fortunately, when we represent $< State0 >= (1, 0)$ and $< State1 >= (0, 1)$, everything would naturally falls into place.

A general two-component vector can be expressed as follows:

$$\mathbf{v} = \begin{pmatrix} a \\ b \end{pmatrix} = a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{4}$$

Recall from linear algebra that $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ form a pair of the simplest unit vectors, serving as a basis for the 2D plane. This implies that this basis can represent any 2D vectors. Therefore, we can interpret Ψ in (3) as a 2D vector $\mathbf{v}$ in (4).

This representation draws attention to that a superposition is the addition of two basic unit vectors with coefficients. This emphasizes why researchers consider linear algebra as one of the foundations of quantum computing.

Let us continue the discussion of a qubit, a superposition of an electron, that can be represented by two two basic states or two (orthogonal) unite vectors:

$$\begin{aligned} \Psi &= a< \mathbf{State0} > + b< \mathbf{State1} > \\ &= a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} a \\ b \end{pmatrix} \end{aligned} \tag{5}$$

where

$$\mathbf{SpinUp} \rightarrow < \mathbf{State0} > \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$\mathbf{SpinDown} \rightarrow < \mathbf{State1} > \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

In quantum physics or quantum computing, a quantum state is usually represented as $|\Psi\rangle$, called Dirac notation or Bra-Ket notation. For instance , $< State0 >= |0\rangle$ and $< State1 >= |1\rangle$. Since this notation carries more insight meanings as we encounter more profound contents or knowledge, we now better to avoid this notation but focusing on the real quantum computing activities. For simplicity, we can just use ,

$< State0 >= \mathbf{0_v} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $< State1 >= \mathbf{1_v} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$

for easier understanding for our next discussion of the quantum algorithms. However, as time goes on, when we are getting more knowledge in quantum computing, we will come back to use Dirac notation. Using $\mathbf{0}_v$ we can immediately imagine the vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ so as $\mathbf{1}_v$.

Now we have

$$\begin{aligned} \Psi &= \begin{pmatrix} a \\ b \end{pmatrix} \\ &= a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= a\mathbf{0_v} + b\mathbf{1_v}. \end{aligned} \tag{6}$$

4.2 Representations of Two and More Qubits

Mathematicians and scientists often seek the simplest approach to model or express real-world problems, avoiding unnecessary complexity. For example, if scalar variables suffice for representing elements in a set, vectors are not employed. Similarly, if vectors are adequate, matrices may be unnecessary, and so on.

As previously discussed, spin-up and spin-down cannot be adequately represented using scalar values alone, hence the utilization of vectors, specifically the simplest two-dimensional vectors.

When dealing with two or more electrons within a quantum system, we must consider their mathematical representation. As mentioned earlier, a single electron can simultaneously exhibit spin-up and spin-down states with different probabilities. Consequently, with two electrons, we encounter four distinct states:

- Spin-up for both electrons
- Spin-up for the first electron and spin-down for the second
- Spin-down for the first electron and spin-up for the second
- Spin-down for both electrons

For simplicity, we can denote these 4 states as $< State_0 >< State_0 >$, $< State_0 >< State_1 >$, $< State_1 >< State_0 >$, and $< State_1 >< State_1 >$. Correspondingly, the indices are represented as 00, 01, 10, and 11; these are just the binary representations from 0 to 3.

When three electrons are considered, we will have 8 basic states. See the following figure. See the following figure.

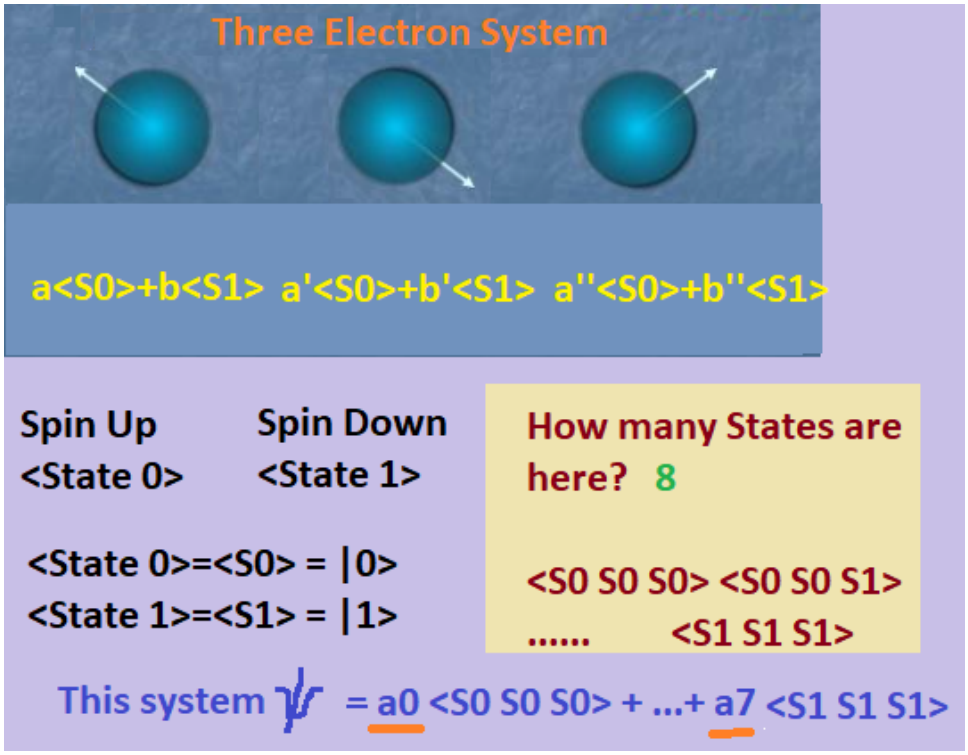


Figure 5: Three qubits generate 8 basic states (basis) [4,5]

What the above observations tell us, 10 qubits system will have 2^{10} states in combination and n qubits would have 2^n states. Therefore, in order to represent a big number represented by quantum states, we can just add some qubits to the system.

To obtaining these states mathematically, we can see that it is similar to so called the Cartesian product of two sets, three sets, and 10 sets. Actually, the proper name for this type of product is called **Tensor Product**. and we will specifically formally introduce this concept in next article. We put an appendix for the example of the Tensor product relating to two qubit states.

We also need keep in mind that a superposition of a quantum state could be the combination of all theses states with coefficients. According to the principle of quantum mechanics, the sum of the square of each coefficient will be 1, the total probability.

4.3 Optional Example: Tensor Product of Two quantum bit states

If you prefer not to delve into unnecessary mathematics in this simple introduction, feel free to skip this subsection.

When dealing with two qubits (states), we use the tensor product $\otimes$ to combine quantum states. We can still represent them using the vector notation for two qubits. For example, if

$|\alpha\rangle = a|0\rangle + b|1\rangle$

$|\beta\rangle = c|0\rangle + d|1\rangle$

The combined state of $|\alpha\rangle$ and $|\beta\rangle$ will be $|\alpha\rangle \otimes |\beta\rangle$, that is:

$|\alpha\rangle \otimes |\beta\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle) = ac|0\rangle \otimes |0\rangle + ad|0\rangle \otimes |1\rangle + bc|1\rangle \otimes |0\rangle + bd|1\rangle \otimes |1\rangle$

We can easily prove that $(ac)^2 + (ad)^2 + (bc)^2 + (bd)^2 = 1$ when $||a||^2 + ||b||^2 = 1$ and $||c||^2 + ||d||^2 = 1$. Since $\otimes$ appeared in every combined states, so we can omit it as just write $|\alpha\rangle \otimes |\beta\rangle = |\alpha\rangle|\beta\rangle = |\alpha\beta\rangle$ for simplification. Thus:

$|\alpha\beta\rangle = ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle$

Since we have 4 base states now ($|00\rangle, |01\rangle, |10\rangle$, and $|11\rangle$) . We can use $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$ to represent $|00\rangle$ and so on. We have:

$$|\alpha\beta\rangle = v_1 \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + v_2 \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} + v_2 \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} + v_4 \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

where $v_1 = ab$, etc.

To use vector's tensor product, we can get the similar representation:

$$|\alpha\rangle = a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

$$|\beta\rangle = c \begin{pmatrix} 1 \\ 0 \end{pmatrix} + d \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

$$|\alpha\rangle \otimes |\beta\rangle = [a \begin{pmatrix} 1 \\ 0 \end{pmatrix} + b \begin{pmatrix} 0 \\ 1 \end{pmatrix}] \otimes [c \begin{pmatrix} 1 \\ 0 \end{pmatrix} + d \begin{pmatrix} 0 \\ 1 \end{pmatrix}]$$

$$|\alpha\rangle \otimes |\beta\rangle = ac \begin{pmatrix} 1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} + ad \begin{pmatrix} 1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 0 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix} + bc \begin{pmatrix} 0 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ 1 \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} + bd \begin{pmatrix} 0 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ 1 \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix}$$

That is

$$ac \begin{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{pmatrix} + ad \begin{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{pmatrix} + bc \begin{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} + bd \begin{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix}$$

Therefore, the two representations will result in (almost) the same vector expressions. Now, you would understand the reason why we use $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ for $|0\rangle$ and $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ for $|1\rangle$.

5 Quantum Gates and Hadamard Gates

Now it's time to introduce the actual components of quantum computing: the Quantum Gates. Since this article provides only an introduction, we will introduce just two quantum gates here. These gates will be used in the quantum algorithm discussed in the final section.

5.1 Three Classical Circuit Gates: AND, OR, NOT Gates

For classical digital computer, there are the three basic logic gates: "AND", "OR", "NOT" (See Wiki):

All computations can be implemented by these three gates. Sometimes, in order to implement an algorithm, their layouts might be very complex. For instance, an **Exclusive-OR**(XOR or $\oplus$) operation—the result is true if two input bits are different. See (*Http: allaboutcircuits.com/textbook/digital/chpt-7/the-exclusive-or-function-xor/*).

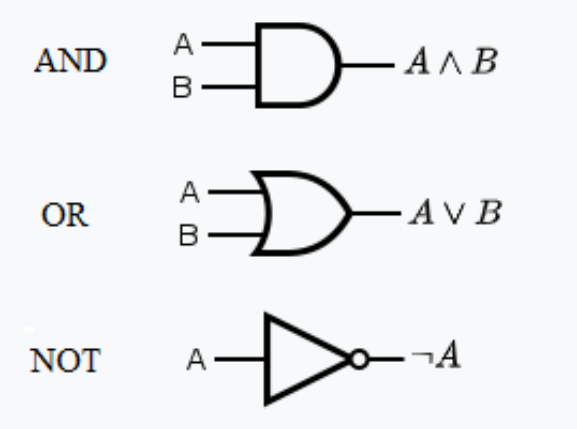


Figure 6: Three Basic Logic Gates. The image is from Wikipedia.

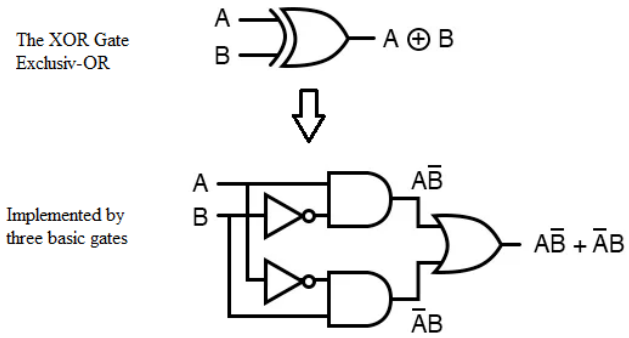


Figure 7: The Exclusive-OR gate and its implementation by three Basic Logic Gates. The image is from Wikipedia.

There is a concept that is called the universal logic gate: This type of the logic gates can construct all other logic gates. We are not going to discuss here.

5.2 Hadamard Gates in Quantum Computing

We have discussed two basic states $< State_0 >$ and $< State_1 >$ ($|0\rangle$ and $|1\rangle$) and the superposition of a qubit. Let us rewrite the general superposition of a qubit:

$$\Psi = a < State_0 > +b < State_1 >$$

We can ask a small question: What would be the simplest superposition containing both $< State_0 >$ and $< State_1 >$? The answer should be that if we select $a = b$. consequently, we have $a = b = 1/\sqrt{2}$ since $a^2 + b^2 = 1$. Further, we also could select or $b = -a$. We have two special states:

$$\Psi_+ = \frac{1}{\sqrt{2}} < State_0 > +\frac{1}{\sqrt{2}} < State_1 >$$

$$\Psi_- = \frac{1}{\sqrt{2}} < State_0 > -\frac{1}{\sqrt{2}} < State_1 >$$

or,

$$\Psi_+ = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{\sqrt{2}}(\mathbf{0_v} + \mathbf{1_v}) \tag{7}$$

$$\Psi_- = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}(\mathbf{0_v} - \mathbf{1_v}) \tag{8}$$

These two superposition states have essential importance in quantum computing. Luckily, it can be made just use a simple quantum gate to achieve them. This gate is called the **Hadamard Gate**, the H-Gate.

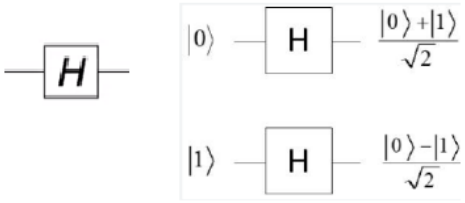


Figure 8: the Hadamard Gate

When input of the Hadamard Gate is $< State_0 >$ we will get Ψ_+ as output, similarly, we can get Ψ_- from $< State_1 >$ if applying the Hadamard Gate. See Fig. 8.

Mathematically, $H(< State_0 >) = \Psi_+$ and $H(< State_1 >) = \Psi_-$. We can also write $H(|0\rangle) = \Psi_+$ and $H(|1\rangle) = \Psi_-$ or $H(\mathbf{0_v}) = \Psi_+$ and $H(\mathbf{1_v}) = \Psi_-$

When input is not a single basic state, we can use a matrix multiplications to calculate the output where the input is $(a, b)^T$, where T is the transpose.

$$H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{9}$$

This is a linear transformation, (a matrix multiplying by a vector always generates another vector,) that is called the Hadamard transformation.

Hadamard transformation has such a good properties that is $HH(\Psi) = \Psi$. In fact , we can verify easily that $H \cdot H = I$ where I is an identity matrix.

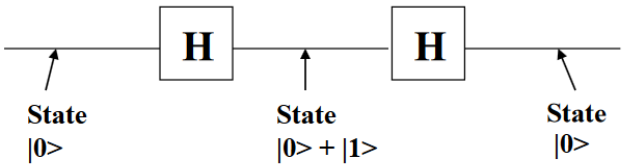


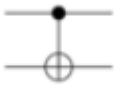
Figure 9: The doubled Hadamard Gates

This properties will assist us to de-coherent the multiple states to a single state. To keep this introduction simple, we will present high order of the Hadamard transformation in later articles.

There are many other types of quantum gates. We have the AND gate, the OR gate, and NOT gate. However, we will discuss in the following article. We want to present a special get here that is called the Controlled-NOT Gate since we will use the idea in our simple quantum algorithm that will be introduced next.

Here $CNOT(x, y) = (x, x \oplus y)$ means when x is 0 then y has no change, when x is 1, y will be flipped the value. In fact, the value here is a quantum state. Basically, $CNOT(x, y)$ preforms a simple entanglement.

Controlled Not (CNOT, CX)



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

Figure 10: The Controlled-NOT Gate: $CNOT(x, y) = (x, x \oplus y)$. The image is from Wikipedia.

6 Deutsch's Algorithm: The First Algorithm of Quantum Computing

The first algorithm of Quantum Computing was developed by D. Deutsch in 1985, for solving the following problem:

Given a function $f : 0, 1 \to \{0, 1\}$, determine whether f is "constant" which means $f(0) = f(1)$, or "balanced" which means $f(0) \neq f(1)$.

This problem, now known as Deutsch's problem, is to distinguish between functions that always yield the same value (constant) and those that yield different values (balanced).

For example: If $f(0) = 1$ and $f(1) = 0$, then f is balanced. If $f(0) = 1$ and $f(1) = 1$, then f is constant.

In classical computation, both $f(0)$ and $f(1)$ must be checked (calculated) to determine the answer. However, Deutsch demonstrated that using quantum computing techniques, only one "calculation" is required to find the answer.

It's important to note that the preparation time, such as placing $f(0)$ and $f(1)$ into the system, is not counted as part of the computation time. In other words, the answer can be obtained without knowing the specific values of $f(0)$ and $f(1)$ as long as they are stored in a black-box beforehand.

While this problem may seem unconventional compared to traditional problem-solving approaches, it marked a groundbreaking discovery that paved the way for future developments in quantum computing.

6.1 Deutsch's Algorithm

In order to simplify the understanding of this algorithm, we extract the core component the the algorithm to explain first. Then we add more steps to it. In order to explain the centrial idea of Deutsch's Algorithm, we will be a little bit tidies. Hope the reader could be a little bit patience. We think everyone will get the idea of the important algorithm after reading this subsection.

Let us assume that H_0 is the Ψ_+ meaning that H_0 is the output of the Hadamard Gate acting on $< State0 >$, i.e. $H_0 = H(\mathbf{0_v}) = H(|0\rangle)$ as well as we let $H_1 = H(\mathbf{1_v}) = H(|1\rangle)$.

Thinking about G is a function of a quantum state, or simply a qubit that is just H_0, then, based on Formula (7), in order to match the function with Deutsch's problem, we can define:

$$\begin{aligned} G(H_0) &= G(\frac{1}{\sqrt{2}}[\mathbf{0_v} + \mathbf{1_v}]) \\ &= \frac{1}{\sqrt{2}}(\mathbf{G(0)_v} + \mathbf{G(1)_v}), \end{aligned} \tag{10}$$

where G is a function from $\{\mathbf{0_v}, \mathbf{1_v}\}$ to $\{\mathbf{0_v}, \mathbf{1_v}\}$.

In other words, if we input $\frac{1}{\sqrt{2}}\mathbf{0_v} + \frac{1}{\sqrt{2}}\mathbf{1_v}$ to a function f in Deutsch's problem , we would have $\frac{1}{\sqrt{2}}\mathbf{f(0)_v} + \frac{1}{\sqrt{2}}\mathbf{f(1)_v}$ as output. Note that the function f could act on the quantum state not only the scalar numbers. That is another characteristic of quantum computing.

Here is the idea, only an idea not a correct answer, check if $f(0) \neq f(1)$ (balanced), we will have the output the same as input; if $f(0) = f(1)$, we will have either $\mathbf{0_v}$ or $\mathbf{1_v}$ with bigger coefficient. Therefore, we see the hope that we could solve Deutsch's problem without calculating both $f(0)$ and $f(1)$. The real drawback of this one qubit input is that when $f(0) = f(1)$, the coefficient would be $2/\sqrt{2}$. It does not satisfy the principles of quantum mechanics that require the sum of squares of all

coefficients must be 1. (Using physics terminology, the quantum transformation must be unitary. but we are not introduce here.)

Now we introduce a new function U_f that takes two inputs $\mathbf{x}$,$\mathbf{y}$ which are two quantum states, the output will be $\mathbf{x}$,$\mathbf{y} \oplus \mathbf{f}(\mathbf{x})$ where $\oplus$ is Exclusive-OR , i.e.

$$U_f(\mathbf{x}, \mathbf{y}) = \mathbf{x}, \mathbf{y} \oplus \mathbf{f}(\mathbf{x}). \tag{11}$$

Note that $\mathbf{y} \oplus \mathbf{f}(\mathbf{x})$ may change the output of $\mathbf{x}$ after U_f. The equation (11) just means that the operator U_f does not act on the first parameter or the first line (See Fig. 11). It does not mean that it will not change the value of the first line. This expression could often confuse first time readers.

To solve Deutsch's problem, we let $\mathbf{x} = H_0 = \frac{1}{\sqrt{2}}\mathbf{0_v} + \frac{1}{\sqrt{2}}\mathbf{1_v}$ and another one is $\mathbf{y} = H_1 = \frac{1}{\sqrt{2}}\mathbf{0_v} - \frac{1}{\sqrt{2}}\mathbf{1_v}$.

Here is the illustration of U_f: The input of the first line is H_0 and the second line is H_1.

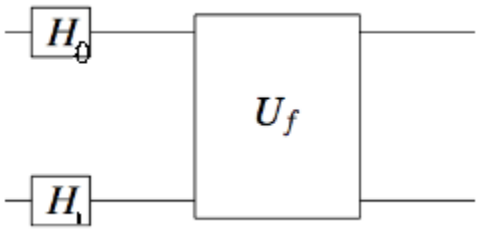


Figure 11: The Core Flowchart for the Deutsch's Algorithm.

Next, we will try to get $U_f(H_0, H_1)$ when f is defined differently.

Note that: the comma "," inside of the parameter really means the Tensor Product (or the Cartesian product alike) since every possible basic state of $\mathbf{x}$ will be combined with every possible basic state of $\mathbf{y}$. The input of U_f will be a superposition $H_0 \times H_1$. That forms the quantum parallel input. So the input for $U_f(H_0, H_1) = U_f(H_0 \times H_1)$ will be:

$$\begin{aligned} H_0 \times H_1 &= (\frac{1}{\sqrt{2}}\mathbf{0_v} + \frac{1}{\sqrt{2}}\mathbf{1_v})(\frac{1}{\sqrt{2}}\mathbf{0_v} - \frac{1}{\sqrt{2}}\mathbf{1_v}) \\ &= \frac{1}{2}(\mathbf{0_v0_v} - \mathbf{0_v1_v} + \mathbf{1_v0_v} - \mathbf{1_v1_v}) \end{aligned} \tag{12}$$

we can express $\mathbf{0_v0_v}$ as $(\mathbf{0_v}, \mathbf{0_v})$ (or $\mathbf{0_v} \otimes \mathbf{0_v}$, as tensor product, but to keep simplicity we did not put emphasis on it).

Since $U_f(x, y)$ only act on the second parameter ($U_f(x, y)$ is a linear function too), therefore:

$U_f : \mathbf{0_v}$,$\mathbf{0_v} \to \mathbf{0_v}, (\mathbf{0_v} \oplus \mathbf{f}(\mathbf{0_v}))$
$U_f : \mathbf{0_v}, \mathbf{1_v} \to \mathbf{0_v}, (\mathbf{1_v} \oplus \mathbf{f}(\mathbf{0_v}))$
$U_f : \mathbf{1_v}, \mathbf{0_v} \to \mathbf{1_v}, (\mathbf{0_v} \oplus \mathbf{f}(\mathbf{1_v}))$
$U_f : \mathbf{1_v}, \mathbf{1_v} \to \mathbf{1_v}, (\mathbf{1_v} \oplus \mathbf{f}(\mathbf{1_v}))$

Thus, the output of U_f ($U_f : x, y \to x, y \oplus f(x)$) will be:

$$\begin{aligned} U_f &= \frac{1}{2}(\mathbf{0_v}(\mathbf{0_v} \oplus \mathbf{f}(\mathbf{0_v}) - \mathbf{0_v}(\mathbf{1_v} \oplus \mathbf{f}(\mathbf{0_v})) + \mathbf{1_v}(\mathbf{0_v} \oplus \mathbf{f}(\mathbf{1_v})) - \mathbf{1_v}(\mathbf{1_v} \oplus \mathbf{f}(\mathbf{1_v}))) \\ &= \frac{1}{2}(\mathbf{0_v}(\mathbf{0_v} \oplus \mathbf{f}(\mathbf{0_v}) - \mathbf{1_v} \oplus \mathbf{f}(\mathbf{0_v})) + \mathbf{1_v}(\mathbf{0_v} \oplus \mathbf{f}(\mathbf{1_v}) - \mathbf{1_v} \oplus \mathbf{f}(\mathbf{1_v})) \end{aligned} \tag{13}$$

Now, we just need a bit more patience to get to go through each of the 4 case of function f to obtain the solution for this Deutsch's dilemma.

Let us perform the analysis directly as follows (other simplified version can be found in [5-8]):

Case 1: If $f(0) = f(1) = 0$, (f is constant), we want to know what happens after U_f? Based on Formula (13),

$$U_f = 1/2(\mathbf{0_v}(\mathbf{0_v} \oplus \mathbf{0_v} - \mathbf{1_v} \oplus \mathbf{0_v})) + \mathbf{1_v}(\mathbf{0_v} \oplus \mathbf{0_v} - \mathbf{1_v} \oplus \mathbf{0_v}))$$

Since $\mathbf{0_v} \oplus \mathbf{0_v} = \mathbf{0_v}$ and $\mathbf{1_v} \oplus \mathbf{0_v} = \mathbf{1_v}$, we have

$$U_f = 1/2(\mathbf{0_v}(\mathbf{0_v} - \mathbf{1_v}) + \mathbf{1_v}(\mathbf{0_v} - \mathbf{1_v})) = 1/2(\mathbf{0_v} + \mathbf{1_v})(\mathbf{0_v} - \mathbf{1_v})$$

$$U_f = (\tfrac{1}{\sqrt{2}}(\mathbf{0_v} + \mathbf{1_v}))(\tfrac{1}{\sqrt{2}}(\mathbf{0_v} - \mathbf{1_v})) = H_0 H_1$$

Case 2: If $f(0) = 0$, $f(1) = 1$, (f is balanced), then

$$U_f = 1/2(\mathbf{0_v}(\mathbf{0_v} \oplus \mathbf{0_v} - \mathbf{1_v} \oplus \mathbf{0_v}) + \mathbf{1_v}(\mathbf{0_v} \oplus \mathbf{1_v} - \mathbf{1_v} \oplus \mathbf{1_v})$$

$$U_f = 1/2(\mathbf{0_v}(\mathbf{0_v} - \mathbf{1_v}) + \mathbf{1_v}(\mathbf{1_v} - \mathbf{0_v})\ .\ (\text{Note: } \mathbf{1_v} \oplus \mathbf{1_v} = \mathbf{0_v}.\ )$$

We can see the difference comparing to Case 1:

$$U_f = 1/2(\mathbf{0_v}(\mathbf{0_v} - \mathbf{1_v}) - \mathbf{1_v}(\mathbf{0_v} - \mathbf{1_v})$$

$$U_f = H_1 H_1.$$

Case 3: If $f(0) = 1$, $f(1) = 0$, (f is balanced), then

$$U_f = 1/2(\mathbf{0_v}(\mathbf{0_v} \oplus \mathbf{1_v} - \mathbf{1_v} \oplus \mathbf{1_v}) + \mathbf{1_v}(\mathbf{0_v} \oplus \mathbf{0} - \mathbf{1_v} \oplus \mathbf{0_v})$$

$$U_f = 1/2(\mathbf{0_v}(\mathbf{1_v} - \mathbf{0_v}) + \mathbf{1_v}(\mathbf{0_v} - \mathbf{1_v})$$

So :

$$U_f = -1/2(\mathbf{0_v}(\mathbf{0_v} - \mathbf{1_v}) - \mathbf{1_v}(\mathbf{0_v} - \mathbf{1_v})$$

$$U_f = -H_1 H_1 = H_1(-H1)$$

Case 4: If $f(0) = 1$, $f(1) = 1$, (f is constant), then

$$U_f = 1/2(\mathbf{0_v}(\mathbf{0_v} \oplus 1 - \mathbf{1_v} \oplus \mathbf{1_v}) + \mathbf{1_v}(\mathbf{0_v} \oplus 1 - \mathbf{1_v} \oplus \mathbf{1_v})$$

$$U_f = 1/2(\mathbf{0_v}(\mathbf{1_v} - \mathbf{0_v}) + \mathbf{1_v}(\mathbf{1_v} - \mathbf{0_v})$$

The result is changed to: $U_f = -1/2(\mathbf{0_v}(\mathbf{0_v} - \mathbf{1_v}) + \mathbf{1_v}(\mathbf{0_v} - \mathbf{1_v})$; therefore,

$$U_f = -H_0 H_1 = H_0(-H_1)$$

These four cases provide us with a definitely clear clue (in Fig. 11):

If we obtain an output H_0 at the first output line, then f is constant; if we get H_1 at the first line, then f will be balanced.

The remaining steps of the algorithm involve only minor additions: As we know, H_0 is a superposition containing $\mathbf{0_v}$ and $\mathbf{1_v}$, and the same applies to H_1; however, our objective is to transform them into the basic state $\mathbf{0_v}$ or $\mathbf{1_v}$. Referring back to the previous information presented in Fig. 9, we can easily find the solution by adding a Hadamard gate H at the end of the first line. This allows us to obtain $\mathbf{0_v}$ if we receive H_0 or $\mathbf{1_v}$ if we receive H_1. This is because $H(H_0) = \mathbf{0_v} = |0\rangle$ and $H(H_1) = \mathbf{1_v} = |1\rangle$.

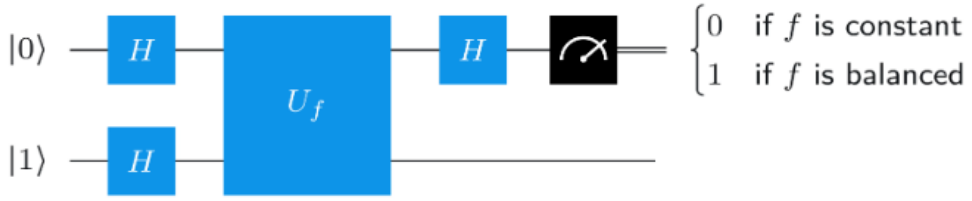


Figure 12: The Deutsch's Algorithm. The image is from IBM [6].

In simpler terms, with the additional H-gate at the first line, we can perform a measurement: if the result is $\mathbf{0_v}$ ($|0\rangle$), then f is constant; meanwhile, if we obtain $\mathbf{1_v}$ ($|1\rangle$), then f is balanced.

The final small adjustment is to obtain H_0 and H_1 for the input of the two lines. This can be achieved by simply adding an H-gate to the beginning of the first and second lines with an input of $|0\rangle$ and $|1\rangle$, respectively. See Fig. 12. [6]

With this, we conclude the entirety of Deutsch's Algorithm.

A question could be raised here: where does the power of quantum computation, or quantum supremacy, manifest when we've dedicated several pages to explaining just a small example in this paper? Answering this question isn't particularly difficult in theory. Consider a function f that doesn't just take a single bit as a parameter, but instead maps an n-bit string to a binary value. How many "calculations" would be required to determine if this new function is constant or not? In classical computers, this would necessitate an exponential number of calculation steps. However, theoretically, we can apply the same strategy as Deutsch's algorithm, which requires only linear time at most. This algorithm is known as the Deutsch-Jozsa Algorithm.

7 Conclusion

In summary, a quantum computer is a device that leverages principles of quantum mechanics to perform quantum computations. These principles include wave-particle duality and the probabilistic interpretation of measurements. There is no question that quantum computers are still in the very early stages of development. They face various technical challenges, especially quantum system errors during computation. Nevertheless, quantum computers still hold great potential to revolutionize certain areas of computation, such as factoring large numbers, optimizing solutions, and simulating complex problems in scientific fields. Many scientists often emphasize that quantum computers are not intended to replace classical computers but rather to contribute to solving specific types of problems much more efficiently.

References

[1] The source of the picture: *https://www.shalom-education.com/wp-content/uploads/2021/02/image-28.png.*

[2] The source of the picture: Wave Interference *https://www.youtube.com/watch?v=AHuhg7QdWZE.*

[3] The source of the picture: *http://labman.phys.utk.edu/phys136core/modules/m9/interference.htmll.*

[4] Joseph Stelmach, Quantum Computing, *https://www.coursehero.com/file/54133623/quantumComputersppt/.*

[5] Peter Young, An Undergraduate Course on Quantum Computing, 2024, *https://young.physics.ucsc.edu/150/phys_150_all.pdf.*

[6] IBM, Quantum query algorithms. *https://learning.quantum.ibm.com/course/fundamentals-of-quantum-algorithms/quantum-query-algorithms*

[7] N. D. Mermin, Quantum Computer Science, Cambridge University Press, Cambridge, 2007.

[8] Ronald de Wolf, Quantum Computing: Lecture Notes, 2023, *https://homepages.cwi.nl/ rdewolf/qcnotes.pdf.*