Components of Quantum Computers and Quantum Algorithms

Alejandro Gallardo

April 22, 2019

1 Introduction

Quantum computing promises to greatly outperform traditional computers at specific task, including database searches and integer factorization. They consequently have the potential to solve problems that are currently beyond the reach of modern computation.

This paper has two goals: 1) describe how quantum computers work: their underlying technological and physical principles and 2) describe a specific quantum algorithm to further elucidate how quantum computers work. Let's begin by examining what quantum computers are and their core components. The audience is assumed to be students familiar with quantum mechanics.

2 Quantum Computers

Some core components of quantum computers are: qubits, quantum logic gates, and quantum registers. The below sections explore what these components are and how they work together.

2.1 Qubit

In quantum computers, the most basic unit of data is the qubit, analogous to the classical bit. Just like in traditional computers, the qubit can either be 0 or 1. More specifically, the qubit is a two state quantum system, which will result in a binary value after being measured. Therefore, particle properties, such as particle spin, can be used to represent qubits (since they are a two state system). For all intensive purposes, we can think of a qubit as the spin state of a particle.

The state of the qubit can be represented as basis vectors in a two dimensional vector space:

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

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2}$$

As we know from quantum mechanics, the overall state of the qubit will be a superposition of the two possible pure states, which can be described as:

$$|q\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \tag{3}$$

where $|q\rangle$ is the mixed state of the qubit, α_0 is the probability amplitude of the $|0\rangle$ pure state, and α_1 is the probability amplitude of the $|1\rangle$ pure state.

We can further define:

$$\begin{aligned} \alpha_0 &= \cos\frac{\theta}{2}, \\ \alpha_1 &= e^{i\phi}\sin\frac{\theta}{2} \end{aligned} \tag{4}$$

which we will later use for visualizing logic gates and qubit wave functions.

2.2 Quantum Logic Gates and Circuit Diagrams

Similar to how traditional computers use logic gates to produce a value depending on input data, quantum computers use quantum logic gates to produce a binary output based on an input.

Mathematically, quantum logic gates are transformations, represented with matrices, which act on the wavefunction of a qubit. Quantum computers can use these logic gates to change the probability distrubition of a qubit's basis states. In addition quantum logic gates can change the relative phase between the two basis states. In other words, qubits serve as the input for a logic gate. If the input to a logic gate is the qubit whose state is describe by $|q_0\rangle$, then the output $|q\rangle$ of the logic gate will be:

$$|q\rangle = \mathbf{T}|q_0\rangle \tag{5}$$

where $\mathbf{T}$ represents a logic gate transformation.

For those familiar with traditional logic gates— if a qubit has a very high probability of collapsing to state 0, a quantum NOT gate can be used to reduce the amplitude of the 0 basis state and greatly increase the probability of the 1 basis state, effectively implementing logical negation.

One logic gate that we will need when trying to understand algorithms is the Hadamard quantum logic gate. It transforms the wave function of a qubit so that each possible state has equal probability of occurring. For the transformation to function properly, the qubit needs to initially be in a collapsed basis state– either $|0\rangle$ or $|1\rangle$. The transformation has the following matrix:

$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{6}$$

Logic gate transformations can also be described using quantum circuit notation. For example, if we apply a Hadamard transformation on a qubit $|q_0\rangle$, we can describe the action with the following circuit diagram:

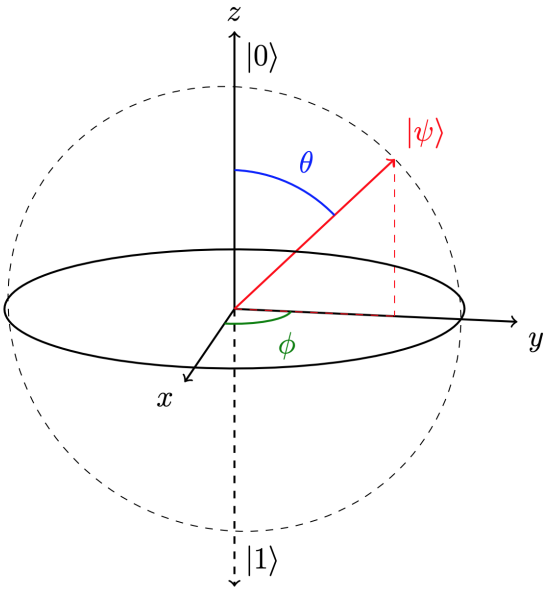


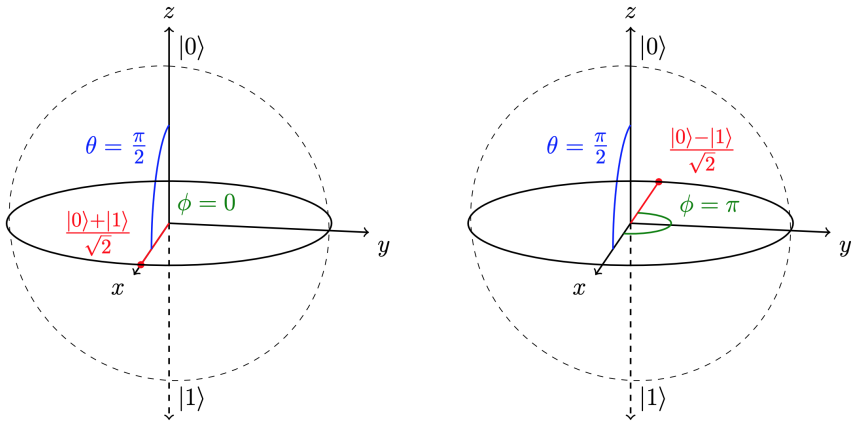
Figure 1: Bloch sphere

$$|q_0\rangle \text{—}\boxed{H}\text{—} \tag{7}$$

Quantum circuits are read like a timeline. From left to right, where left is the initial state of the qubit or system, where time progresses to the right. The Hadamard circuit— Eq. 7— reads as: qubit begins in state $|q_0\rangle$, and then it is transformed by the Hadamard transformation represented as a square bock labeled **H**. The right part of the circuit is open ended, implying further connections.

We can better understand how logic gates operate with other kinds of visualizations, such as the Bloch sphere. As shown in Fig. 1, the Bloch sphere visualizes the wave function of a qubit, providing information about the probability amplitudes of the basis states as well as information about the phase different between basis states. The wave function of a qubit is described as a ray in the sphere starting at the origin, and it indicates the state of the sphere, where θ encodes information about probability amplitudes of the basis states and ϕ encodes information relative phase, in accordance with Eq. 4.

In general, the Bloch sphere can describe any two state quantum system. We can gain an intuition for how we can use the Bloch sphere by applying it to a $\frac{1}{2}$ particle spin system. In this case, the shown diagram uses S_z as the basis states, where each state is denoted as $|+Z\rangle = |0\rangle$ and $|-Z\rangle = |1\rangle$. When θ is zero, the qubit is in the $|+Z\rangle$ state; when θ is π, the qubit is in the $|-Z\rangle$ state; when θ is $\frac{\pi}{2}$ and $\phi = 0$, the qubit is in the $|+X\rangle$ state, and therefore in a superposition of both $|+Z\rangle$ and $|-Z\rangle$ states with equal probability amplitudes. Therefore, the Hadamard transformation acts similar to a rotation in the Bloch sphere. For example, if we have a qubit that is initially in the $|0\rangle$, which corresponds to a wave function ray that points in the +Z direction, and then apply a Hadamard transformation, the qubit



(a) Basis state $|0\rangle$ (b) Basis state $|1\rangle$

Figure 2: Bloch sphere representation of the Hadamard operator applied to a) $|0\rangle$ and b) $|1\rangle$ [1]

state will rotate $\theta = \pi/2$ and be in the $|+X\rangle$ state. Consequently, the qubit will be described to be in equal superposition between the basis states $|0\rangle$ and $|1\rangle$.

2.3 Quantum Registers

So far, we've seen how quantum gates can affect a single qubit. In practice, quantum computers are made up of many qubits and quantum gates. A specific arrangement of qubits is called a quantum register, analogous to traditional processing registers.

In order to keep track of the overall state of a register, we need to keep track of the state of each individual qubit in the register. Recall that we can describe a single qubit as $|q_0\rangle$. For a two qubit register we can describe the register as: $|q_0\ q1\rangle$. Rinse lather repeat: for a 3 qubit register, we can describe the pure state of the register as:

$$|q_0\ q_1\ q_2\rangle \tag{8}$$

Just as for half spin particles, the mixed state of a quantum register can be described as a superposition of all the possible pure states. Mathematically, this results in:

$$|\Psi_N\rangle = \sum_{i=0}^{2^N-1} a_i|i\rangle \tag{9}$$

where $|\Psi_n\rangle$ is the mixed state of the register, $|i\rangle$ is a basis state of the register, and a_i is the probability amplitude of a particular basis state. For a 3 qubit system, the expanded sum becomes:

$$|\Psi_n\rangle = a_{000}|000\rangle + a_{001}|001\rangle + a_{010}|010\rangle + a_{011}|011\rangle + a_{100}|100\rangle + a_{101}|101\rangle + a_{110}|110\rangle + a_{111}|111\rangle \tag{10}$$

For Eq. 10, i is written as a base-2 integer to encode the state of each internal qubit in the register. For a 3 qubit system, we can rewrite Eq. 14 so that it better matches our base-2

integer notation:

$$|\Psi_N\rangle = \sum_{q_2=0}^{1} \sum_{q_1=0}^{1} \sum_{q_0=0}^{1} a_{q_0 q_1 q_2} |q_0\ q_1\ q_2\rangle \tag{11}$$

The summation is only from 0 to 1 because each qubit only has 2 possible states— $|0\rangle$ or $|1\rangle$. Recall that equation 11 is just defining the state of the register as the superposition of all the possible pure register states.

The probability amplitude $a_{q_0 q_1 q_2}$ of a register state will be the product of the probability amplitudes of the relevant qubit probability amplitudes: a_{q_0}, a_{q_1}, a_{q_2}. For example, in a register composed of qubits q_0, q_1, and q_2, the probability amplitude a_{010} of the register state $|0\,1\,0\rangle$ will be the product of the probability amplitude of the $|0\rangle$ basis state of the q_0 qubit, the amplitude of the $|1\rangle$ basis state of the q_1 qubit, and the amplitude of the $|0\rangle$ basis state of the q_2 qubit. For a 3 qubit system, we can mathematically describe this as:

$$a_{q_0 q_1 q_2} = a_{q_0} * a_{q_1} * a_{q_2} \tag{12}$$

where q_x is either 0 or 1, corresponding to the $|q_x\rangle$ qubit. Equations 11 and 12 can be generalized for registers with more than 3 qubits, but will be left as an exercise for the reader.

3 Grover's Algorithm

Now that we have an understanding of how quantum computers function, let's turn our attention to a specific algorithm that can further our understanding.

Grover's algorithm is a search algorithm that can locate an element in an unordered data set of size N with a time complexity of just $O(\sqrt{N})$. The best time complexity for traditional searching algorithm is O(N). In practice, Grover's algorithm has been predominantly achieved for 2 and 3 qubits. In 2018, a team of scientist achieved a 4 qubit algorithm that uses IBM's QX5 quantum computer.

For a 4 qubit register, there are $2^4 = 16$ possible states that the register can be in, and the state of the register can be represented as shown in expression 14. For Grover's algorithm, each of the 16 possible register states represents a data value within our dataset. The goal of the algorithm is to increase the probability amplitude of the state that corresponds to the data value we seek.

Conceptually, Grover's algorithm has four major components:

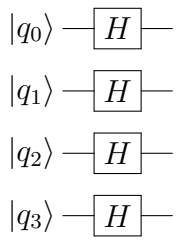
- Initialization: all qubits are set to be in superposition with equal amplitudes $1/sqrt(2^n)$ where n is number of qubits. This means that all register states will also have equal probability amplitudes.

- Oracle phase: The algorithm "marks" a specified register state by flipping the phase of the corresponding probability amplitude (phase rotation of pi). The algorithm marks states depending on whether the state satisfied a condition, namely: f(x') = 1, where x' is a particular state, and f is a function. This step runs in constant time.

- Amplification phase: The amplitudes of the states are reflected over the mean value of the probability amplitudes of the register states.
- Measurement phase: system is collapsed and a measurement is read.

3.1 Initialization

In order to prepare a register for executing Grover's algorithm, each possible qubit state needs to have equal probability of being measured. To achieve this, we can apply the Hadamard logic gate to all the relevant qubits in a register. For a Grover's algorithm that uses a four qubit register, the circuit diagram for initialization can be represented as:



(13)

where square labeled H represents the Hadamard transformation.

For a four qubit register, each basis state will have a probability amplitude of $\frac{1}{4}$ after applying the Hadamard gates as shown in circuit 13. Mathematically this translates to:

$$|\Psi_4\rangle = \sum_{i=0}^{2^4-1} \frac{1}{4}|i\rangle \tag{14}$$

3.2 Oracle Phase

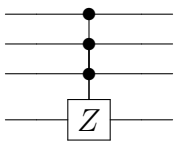
In the oracle phase, the algorithm searches for a state that satisfies some predetermined condition, which we can mathematically represent as f(x') = 1, where x' is a particle state for the register. When a particular state satisfies the condition, the algorithm will "mark" the state by applying a phase flip to the state's probability amplitude.

Consider a four qubit register as an example. The first iteration of the oracle phase will change the original amplitude to $-\frac{1}{4}$ for the target register state. The phase flip can be achieved using a combination of logic gates called the Pauli Matrices. The effect of the Pauli matrices can be easily visualized on the Bloch sphere– they will transform a qubit by rotating its wave function about a specific axis in the Bloch sphere by π radians. For example, the Pauli X matrix will rotate a qubit about the X-axis in the Bloch sphere by π, while the Pauli Z matrix will rotate the qubit's wave function about the Z-axis in the Bloch sphere by π. The Pauli Z logic gate can be represented as:

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

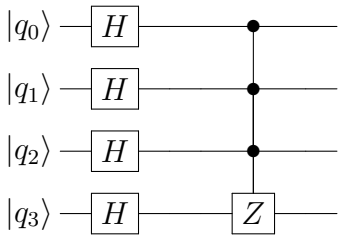
$$|q_0\rangle \text{—} \boxed{Z} \text{—} \tag{15}$$

We can use the Pauli Z gate to create a very simple logic sequence that will "mark" our signal as desired– in other words perform a phase flip for one of the register basis states. So how can we flip the probability amplitude of a particular register state? Eq. 12 gives us an intuition for how we can do so. If we can flip the sign of the probability amplitude of a relevant qubit basis state, then we will have flipped the overall sign of the register state's probability amplitude. For example, if we wanted to flip the probability amplitude of state $|1\,1\,1\,1\rangle$, we could apply a transformation such that the probability amplitude of the $|1\rangle$ basis state for one of the qubits flips signs. We can construct such a circuit using a single Paul Z logic gate. The circuit diagram can take the form:



(16)

If we combine the circuits from the Initialization (circuit 13) and the Oracle phase (circuit 16), we get a sample circuit that looks like:



(17)

Circuit 17 shows how the q_3 qubit will first undergo a Hadamard transformation, and will therefore be in the $|+X\rangle$ state. We can visualize this using the Bloch sphere as shown in Fig. 2. Then, the Pauli Z transformation will rotate the $|+X\rangle$ state around the Z access by π. Therefore, the resulting state for the q_2 qubit will be $|-X\rangle$. Recall, as shown in Fig. 2, the $|1\rangle$ pure state of the $|-X\rangle$ state will have a probability amplitude of $-\frac{1}{\sqrt{2}}$ as opposed to positive $\frac{1}{\sqrt{2}}$! Therefore, the probability amplitude of state $|1\,1\,1\,1\rangle$ will also have flipped signs since: $\frac{1}{\sqrt{2}} * \frac{1}{\sqrt{2}} * \frac{1}{\sqrt{2}} * \frac{-1}{\sqrt{2}} = \frac{-1}{4}$. In this example, oracle performed the desired function for a particular state. Note that the circuit necessary to flip probability amplitudes will change depending on what the target state is. For example, we will need a different circuit to mark the $|0\,0\,0\,0\rangle$ state, compared to the $|1\,1\,1\,1\rangle$ state.

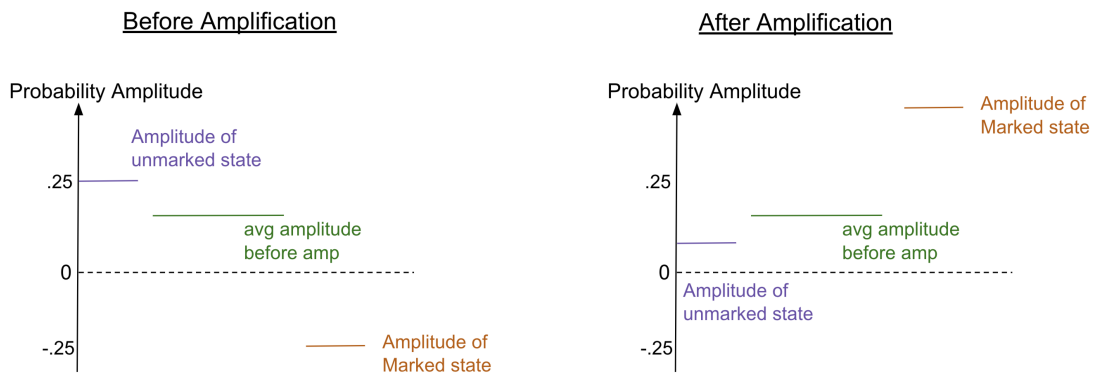


Figure 3: Diagram depicting the amplification stage of Grover's algorithm. In particular, it shows the very first amplification on a four qubit register.

3.3 Amplification

Before the phase flip from the oracle phase, the average probability amplitude will be $\frac{1}{4}$, but after phase flip, the new average will be lower than before. This is because the states that were marked by the oracle (with a phase flip) will have a negative amplitude, consequently lowering the average probability amplitude. Since the amplification stage flips all the probability amplitudes over the average value, all the marked states will have a considerably larger amplitude after amplification, as shown in Fig. 3.

3.4 Measurement

The qubits are then measured, collapsing all the qubit states.

4 Conclusion

This paper briefly discussed the various components of quantum computers as well as a specific search algorithm.

References

[1] Emma Strubell *Introduction to Quantum Algorithms*. 2011.

[2] Philip Stromberg *4-qubit Grover's Algorithm Implemented for the IBM QX5 Architecture*. 2018.

[3] Ryan O'Donnell *Lecture 1: Introduction to the Quantum Circuit Model*. 2015.