Quantum Computing

Amir El-hamdy

September 26, 2016

Computers are ubiquitous in our modern world. We don't know how many there are in existence; billions at least. There are computers in our phones, tablets, cars and of course in our laptops and desktops, and they are getting more powerful each day. The processing power of computer chips has been doubling every couple of years, according to Moore's Law; however for the last decade or so this rate of increase has been slowing down as transistors are now reaching the nano-scale where quantum effects come into play. These quantum phenomena, such as quantum tunnelling, pose a danger to the development of the computers we are most familiar with. This is because the computers that run our laptops and phones are deterministic. Given the same initial conditions and the same operations to carry out, a deterministic computer should always produce the same result. Due to the probabilistic nature of quantum effects this poses a huge problem to the development of deterministic computers at smaller and smaller scales. However the quantum effects that are observed at the nano-scale can be used to create a new kind of computation altogether.

Where did the idea of quantum computers come from?

The idea of creating a device that exploited quantum mechanics to do computation was already being explored at around the 1970s. One of the first people to propose Quantum Computers was Yuri Manin in his 1980 paper "Computable and Uncomputable". Then in 1982 Richard Feynman proposed a basic model for a quantum computer that could simulate quantum processes[1]. Over the next couple of decades several algorithms were developed including Grover's algorithm, a quantum database search algorithm, and Shor's algorithm which would allow quantum computers to factor large integers quickly. Shor's algorithm really brought quantum computing into the limelight since much of the World Wide Web's security depends upon the fact that it is hard to factor integers on a classical computer[2].

How do quantum computers work?

To understand quantum computing first one must first understand the qubit. A 'bit' is simply a unit of information describing a two-dimensional classical

system, it can be either on $|1\rangle$ or off $|0\rangle$. However a 'qubit' or 'quantum bit' is a unit of information describing a two-dimensional quantum system, which can be in a superposition of the on and off states[2].
We can represent the classical bit states on and off with the following 2-by-1 matrix :

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

These states are known as computational basis states.
Similarly we can represent a qubit with the following 2-by-1 matrix with complex numbers:

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

where $|\alpha|^2 + |\beta|^2 = 1$. Here we see that the classical bit is a special case of the qubit. As soon as the qubit is measured, it collapses to either the on or off state. The quantities $|\alpha|^2$ and $|\beta|^2$ give the probability of measuring the qubit in states $|0\rangle$ and $|1\rangle$ respectively. Also note that the classical bit is a vector in two-dimensional real vector space $\mathbb{R}^2$ and the quantum bit is a vector in two-dimensional complex vector space $\mathbb{C}^2$. In addition, both vectors are normalised so we can think of bits and qubits as unit vectors in their respective vector space.

It turns out we can rewrite the expression for a qubit as:

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

since $|\alpha|^2 + |\beta|^2 = 1$.
We can ignore the factor of $e^{i\gamma}$ out the front because it doesn't have any observable effects. So we essentially have

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

[3]. This allows us to visualise the state space of a qubit on the unit three-dimensional sphere known as the Bloch sphere. For a given θ and ϕ we can define a point on the Bloch sphere as shown in Figure 1.

This seemingly creates a paradox. As we can see there are an infinite number of points on the Bloch sphere, so does this mean that a qubit can store an infinite amount of information in a single qubit? In theory this is true, however when the qubit is measured it must collapse to a $|0\rangle$ or $|1\rangle$ state just like a classical bit. Therefore we can only retrieve one bit of information about the state of the qubit on measurement, which resolves the paradox. So although there are an infinite number of states hidden within a qubit, if we cannot retrieve it upon measurement how could we quantify the 'information' stored within? Nonetheless, if we had an infinite number of identically prepared qubits and measured

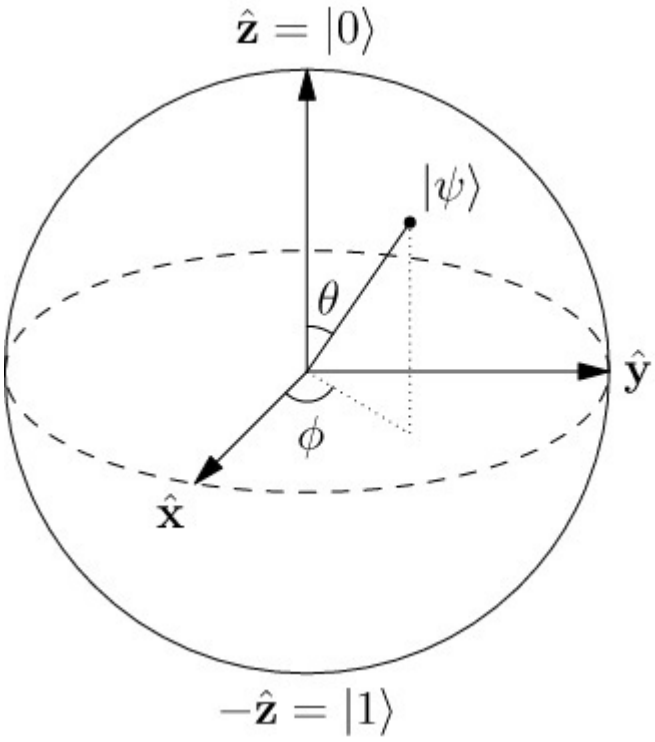

Figure 1: Bloch sphere showing the state space for a qubit[4].

them all, then we would be able to determine the coefficients α and β mentioned earlier. An important point to make however, is that during the evolution of a closed quantum system Nature keeps track of all this 'hidden information'. In addition as we add more qubits together in a system the potential amount of this hidden information grows exponentially[3]!

So what happens when we create systems of multiple qubits? Unfortunately there are no known simple generalisations of the Bloch sphere for multiple qubits [3] so we must proceed without visual aid. Each qubit alone has two possible computational basis states, $|0\rangle$ and $|1\rangle$, as we saw earlier. Thus for an N qubit system, there are 2^N possible computational basis states. For example, if we have a two-qubit system classically there would be four possible states: 00, 01, 10, 11. This corresponds to the following four computational basis states: $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. Qubits can exist in superpositions of these computational basis states therefore we need complex coefficients to associate an amplitude with each basis state as before. So we could describe pair of qubits with a state vector of the form

$$|\psi\rangle = \alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle$$

.
Of course the sum of these coefficients must sum to 1 due to the normalisation condition which can be succinctly written as:

$$\sum_{x \in \{0,1\}^N} |\alpha_x|^2 = 1$$

where N is the number of qubits in the system and $\{0,1\}^N$ denotes the set of strings with length N composed only from the characters '1' and '0' [3].
When we measure any subset of the qubits we collapse the wavefunction and in its place obtain a new one. For example, if we took a measurement of the first qubit in the system above and found that it was in state '0' that qubit is now fixed and we would now obtain the following new wavefunction:

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}$$

Note the wavefunction has to be renormalised after measurement[3].

In order to build any meaningful quantum circuits we need to manipulate these qubits. Classically we could manipulate bits with logic gates thus we will need quantum logic gates for a quantum computer. In ordinary computers we can represent all logical circuits as a combination of the AND and NOT gates as they form what is known as a set of universal logical gates. Just like in classical computing there are sets of universal logical gates one of which is composed of the Hadamard gate, the controlled-NOT gate and a phase shift gate[2]. For the remainder of this essay we will only need to look at the first of these.

We can represent the classical logical gates using matrices. For example the classical NOT gate could be represented by the matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.
Similarly we can also represent quantum logic gates using matrices, however there are some constraints. The matrices need to be reversible and preserve the geometry of the space they are acting on[2]. This means we must use unitary matrices to describe quantum logic gates but this is the only constraint necessary for quantum gates[3]. This means that any unitary matrix is a valid quantum logic gate! That is, the matrix must satisfy $U^\dagger U = I$ where U is the unitary matrix and $U^\dagger$ denotes the conjugate transpose.

The Hadamard gate mentioned above is a gate that acts on a single qubit. It can be used to take a qubit from one of its computational basis states to a superposition of all the possible computational basis states. So for a two-qubit system the Hadamard gate will take a qubit in the $|0\rangle$ state halfway to the $|1\rangle$ state and vice versa. The Hadamard gate is also its own inverse so applying the gate twice has no effect on a qubit. The matrix representation of the Hadamard

gate is $\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$[3].

How can quantum computers be utilised?

Now that we know the distinction between a classical bit and a qubit as well as how we can build and manipulate basic multiple-qubit systems, we need to know how we can use this difference to our advantage. Quantum computing algorithms had been developed long before the first quantum computation was ever achieved, in the same way that classical algorithms had been around centuries before the first classical computer was built. All quantum algorithms work with the following basic framework:

The system will start with the qubits in a particular classical state.

From there the system is put into a superposition of many states.

This is followed by acting on this superposition with several unitary operations.

And finally, a measurement of the qubits[2].

Obviously algorithms will vary from this scheme but it is useful as a starting point.
The simplest of these algorithms is the Deutsch Algorithm. The Deutsch algorithm solves the following problem: Given a function

$$f : \{0, 1\} \to \{0, 1\}$$

determine if the function is balanced or constant, where a function is balanced if $f(0) \neq f(1)$ i.e. it is one to one, and a function is constant if $f(0) = f(1)$ [2].

If a classical computer were to solve this problem, it would have to evaluate f for each input and then compare the outputs. Typically one would start by evaluating f at the beginning of the domain, $f(0)$, and then working their way down the rest of the inputs. Once the entire domain has been evaluated we can then work out whether the function is balanced or constant. This classical way of doing things can be summarised with the decision tree in Figure 2

In contrast a quantum computer can be in a superposition of all the basic states at the same time. We can utilise this quantum parallelism to evaluate the inputs simultaneously. We can think of the function as a unitary (and therefore reversible) matrix which means that given the output we should be able to deduce the input. As we saw earlier any unitary matrix can be thought of as a quantum logic gate, so we can create a circuit as shown in Figure 3.

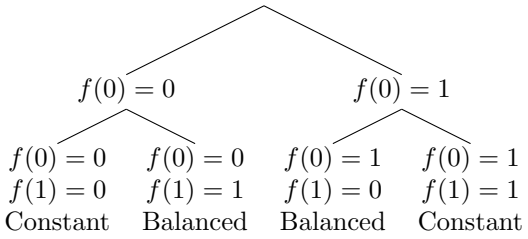

Figure 2: Classical decision tree of Deutsch's algorithm.

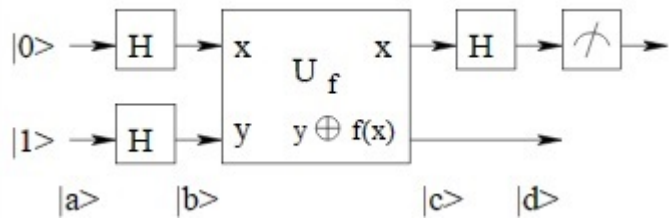

Figure 3: Circuit diagram of Deutsch's algorithm[5].

We initially apply Hadamard gates on the states $|0\rangle$ and $|1\rangle$ to get the input qubits into a superposition. The unitary matrix that represents the function U_f then takes these input qubits in the state $|x, y\rangle$ and outputs the state $|x, y \oplus f(x)\rangle$ where $\oplus$ is addition modulo 2. After applying another Hadamard gate to the top qubit we take a measurement and receive an output in the form

$$\pm\,|f(0) \oplus f(1)\rangle\,(\frac{|0\rangle - |1\rangle}{\sqrt{2}})$$

Since $f(0) \oplus f(1)$ is 0 if the function is constant and 1 if balanced then we only need one measurement to distinguish between a constant or balanced function[1]. With this quantum circuit we were able to obtain a global property of $f(x)$ in a single evaluation[3]! Although we have answered the problem in a single operation we have not gained more information out of that operation that in the classical case. We have simply restructured the order in which we receive the information. Instead of evaluating the value of the function at all points in its domain and inferring the answer at the end, we have observed a global property of the function without explicitly evaluating the value at any point in the domain[2]. In order to obtain all the information of the system, we would have to do one more operation to evaluate the value of the function at one point in the domain and hence infer the value at the other point. This decision tree is shown in Figure 4; contrasting it with the classical decision tree in Figure 2 we see clearly that the order in which we receive the information has simply been restructured.

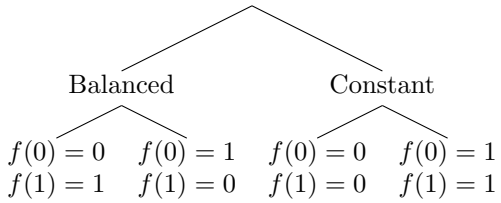

Figure 4: Quantum decision tree of Deutsch's algorithm.

Is quantum computing the future?

So far we've seen the power that quantum computing can bring us, but only in rather contrived and ideal conditions. We have seen that in theory a qubit can hold a potentially infinite amount of 'information', however we can only retrieve a single bit upon measurement. We have seen that quantum circuits can restructure the classical decision tree, but ultimately we are not getting any hidden information. The ability to hold all possible function values at once and then allow the results to interfere with each other, as we saw in Deutsch's algorithm, can allow us to provide exponential speed-up times in niche problems such as integer factorisation[6]. The solutions to problems like these can have staggering consequences. For example the problem of factoring integers is essential to breaking modern encryption methods largely employed by the web[7]. Quantum computing doesn't just pose a large threat to the encryption of the future but also the present and the past. A quantum computer built years into the future could decrypt this recorded communication. So internet communication recorded today could be broken perhaps a decade into the future from now and there is a lot of information that needs to remain secure for decades. Thus it is important to consider encryption methods that remain secure against quantum computers even now. Companies are beginning to recognise this; recently Google announced an experiment in Chrome, where a small fraction of connections between the browser and Google's servers would use a post-quantum key-exchange algorithm in addition to the ordinary algorithm[8]. So although these computers are not likely to replace our home computers for everyday tasks; I think the potential application of quantum computing in computationally intensive fields such as scientific computing and Artificial Intelligence will be prodigious.

References

[1] A. Hagar, M. Cuffaro. Quantum computing. 2006 [accessed September 26 2016]. Available from: http://plato.stanford.edu/entries/qt-quantcomp/

[2] A. Mannucci Mirco, S. Yanofsky Noson Quantum computing for computer scientists. Cambridge University Press, 2008.

[3] L. Chuang Isaac, A. Nielsen Michael Quantum computation and quantum information. Cambridge University Press, 2000.

[4] Wikimedia Commons. Bloch sphere, 2012 [accessed 26 September 2016]. Available from: https://upload.wikimedia.org/wikipedia/commons/f/f4/Bloch_Sphere.svg

[5] Wikimedia Commons. Deutsch algorithm circuit, 2007 [accessed 26 September 2016]. Available from: https://upload.wikimedia.org/wikipedia/commons/8/8a/Deutsch_algorithm_circuit.svg

[6] S. Bone, M. Castro. A brief history of quantum computing [accessed 26 September 2016]. Available from: http://www.doc.ic.ac.uk/ nd/surprise_97/journal/vol4/spb3/

[7] A primer on public-key encryption [accessed 26 September 2016]. Available from: https://math.berkeley.edu/ kpmann/encryption.pdf

[8] M. Braithwaite. Experimenting with post-quantum cryptography. 2016 [accessed 26 September 2016]. Available from: https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html