A Gentle Introduction to Quantum Computing

Abdullah Khalid
2012-10-0168

School of Science and Engineering
Lahore University of Management Sciences

Friday 3rd June, 2011

Contents

1 Introduction to Quantum Computing

The field of quantum computing was pioneered in 1985 by Daved Deutsch [2]. Building upon a suggestion by Feynman [1] and the work of other scientists, he generalized the concept of the Turing Machine as postulated by Turing [3]. He invoked quantum mechanics, our most accurate description of reality till now—realizing that the Turing Machine had the implicit assumption of classical mechanics—and reworked the Turing Machine.

The concept behind quantum computing is simple. Use quantum mechanical systems with the full use of their quantum mechanical properties to do computations. Why is this useful? As was shown over the years, quantum mechanical dynamics can be used to compute the answers to certain problems much faster than any classical system or

computer can[1]. For instance numbers can be factored in polynomial time by quantum computers, compared to the exponential time of classical ones. This has brought about a fundamental revolution in the field of complexity theory and cryptography, among others.

The problem, however, is that no large scale quantum computer has ever been built. The largest such system was just powerful enough to factor 15 into 3×5 with high probability! Needless to say, we are a long way off from the construction of sufficiently powerful quantum computers. By sufficiently powerful, I mean powerful enough to solve problems that current classical computers can do and preferably much more.

At the heart of quantum computing lies the concept of quantum superposition. A classical system can only be in one state at a time. A quantum system can be in a superposition of multiple states at a time. Hence, the idea is to perform computation in parallel. Though it's much trickier than the parrallel processing implemented in networked classical computers today.

There are two reasons for this. First, one can set the initial state of a quantum computers into a superposition of states. But when you evolve that state—or closer to the language of computing, you perform computation on it—the different states do not remain separate. If two different states evolve into the same final state, there is no way to distinguish how much amplitude is due to each of those initial states. This puts a huge limit on the type of quantum algorithms we can design, and explains why we can't just import classical network computing algorithms to the field of quantum computing.

The second reason is that of measurement. This is, in fact, even a more fundamental restriction on what sort of computations we can do. Recall that whatever the state of a quantum system, a measurement on it only gives us one of the possible eigenvalues[2]. Therefore, even if we evolve a system into a superposition of final states, we only get to find out one of them. Repeating the computation or experiment over and over again is of not much use either, because it defeats the purpose of quantum computers solving problems faster. Nevertheless, clever algorithms have been designed that give the answer to a problem in just one measurement. Though this is not the only useful construction of quantum algorithms.

Nevertheless, quantum computing holds enormous promise. It has changed fundamentally how we think about physics and computer science. Moreover, it promises great technological innovations.

In the following sections, I present an introductory level explanation of quantum computing.

[1]The more precise statement is that there are no known 'classical' algorithms for these problems which can compute the answers as fast as the corresponding 'quantum' algorithms. However, we have as yet no reason to believe that such classical algorithms are not possible.

[2]I would desist from using the word 'collapse'. As Deutsch has consistently argued over the years, the idea that a sufficiently large quantum computer when built would be able to do simultaneously perform much more than 10^{80} operations - as an example 10^{400}. Recall that there are an estimated 10^{80} atoms in the universe. This would be overwhelming evidence for Everett's Relative State Formulation of Quantum Mechanics as opposed to the standard Copenhagen interpretation which postulates a collapse of the wave function.

2 Modelling Quantum Computers

There are two fundamental ways that we can look at the theoretical construction of quantum computers.

2.1 Turing Model

The quantum computer is fundamentally the same theoretical object as a classical computer as far as computation is concerned. This means that a quantum computer can compute nothing a classical computer cannot and vice versa. Since, a classical computer is equivalent to a Universal Turing Machine, so is a quantum computer.

In other words, quantum computers may make certain problems tractable i.e. make it possible to compute them in polynomial time compared to relatively slower algorithms for classical computers.

The Turing model might be great for a core theoretical understanding of a computer. However, for practical implementation a different model is needed.

2.2 The Circuit Model of Quantum Computation

The circuit model is essentially similar to the circuit model for classical computers. It allows us to (relatively) easily visualize the state of the system at any point in the computation given a definite input. An example of such a circuit model is given in 1. This is the circuit for the quantum Fourier transform, which we will encounter many times in these notes.

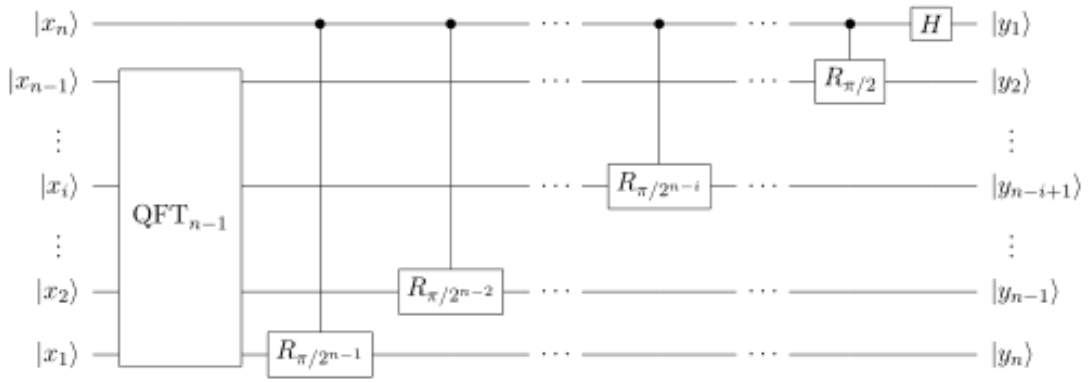

Figure 1: The circuit for the Quantum Fourier Transform

It shows all the major features of the quantum circuit model: quantum wires, quantum gates and qubits. We will discuss all these features in greater detail.

2.3 Qubits

The basic information resource in quantum computation is the qubit, which is derived from"quantum bit".

Essentially, all the information being that is manipulated during the course of a quantum computation is stored in registers of qubits. A single qubit is a two state quantum system. A register is a set of qubits.

We know from quantum mechanics that a two state system in quantum mechanics can be in any superposition of the two basis states. Conventionally, the two basis states in quantum computating literature are represented by $|0\rangle$ and $|1\rangle$. This allows us to write the state of a single qubit as

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

. α and β are in general complex, and due to the normalization condition of state vectors $\alpha * \alpha + \beta * \beta = 1$. Here we have

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

and

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

. This means that

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

In other words, generic quantum states can be represented by column matrices.

To represent a register of qubits we use the following scheme. For a two qubit system, the possible states are $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. This means that an arbitrary state fo a two qubit system can be represented by

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

.

In column matrix formulation, the basis states are

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$|10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

This simple scheme can be generalized for a N qubit system to give us

$$|\psi\rangle = \sum_{i,j,\ldots,m \in \{0,1\}^N} \alpha_{ij\ldots m}\,|ij\ldots m\rangle$$

3 Quantum Gates

To perform operations on qubits we require quantum gates. Quantum gates are essentially evolutions of quantum states. In the circuit model they can be treated as blackboxes, meaning that for the time being we don't care how they are physically implemented. All we care about the inputs they take and the corresponding outputs.

Quantum gates have the property that they have an equivalent number of inputs and outputs. This stems specifically from the fact that quantum mechanics is a theory that obeys time symmetery. A quantum mechanical system that is evolved from a given initial state to a final state using a definite series of operations can be evolved backwards from the final state to the initial state using the inverse of the series of operations on it.

A quantum computer is also a reversible machine. A gate operating on an input to turn it into an output will not be able to do the reverse if the number of outputs are smaller than the number of inputs because information would be lost in the process. Hence, the equivalence of the number of inputs and output. We can divide quantum gates into categories based on the number of inputs/outputs.

To represent the operation of particular gates, two schemes are used. The first one is the more abstract one. It involves giving the operation of the gate on the basis states. Since, any quantum state can be decomposed as a linear combination of the basis states, giving the operation on the basis states is sufficient to describe the operation of the gate on any state. We will use this scheme later on for large qubit systems.

The second representation involves matrices. Since, the state of a qubit register can be given by kets or column matrices, operations on them can be represented by square matrices.Multiply an input state column matrix with the matrix for the quantum gate, we obtain the output state. This scheme makes conceptualizing quantum computing easier. However, it is not practical for large systems, where the matrices would be too large to yield on paper or the computer screen.

3.1 Single Qubit Gates

The simplest quantum gates are single qubits gates. We look at a couple of such gates.

Identity Gate Of these the simplest gate is the identity gate. This identically outputs the input. It's typically represented by $\mathbf{I}$. In the first scheme, we have

$$\begin{aligned}\mathbf{I}\left|0\right\rangle &= \left|0\right\rangle \\ \mathbf{I}\left|1\right\rangle &= \left|1\right\rangle .\end{aligned}$$

This completely specifies the operation of the identity gate. Given the generic state $\left|\psi\right\rangle = \alpha\left|0\right\rangle + \beta\left|1\right\rangle$, we can input it into the identity gate to get out

$$\begin{aligned}\mathbf{I}\left|\psi\right\rangle &= \mathbf{I}(\alpha\left|0\right\rangle + \beta\left|1\right\rangle) \\ &= \alpha\left|0\right\rangle + \beta\left|1\right\rangle .\end{aligned}$$

In the matrix representation, the $\mathbf{I}$ is given by the identity matrix

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Multiplyin any state with $\mathbf{I}$ will just return the original state.

NOT Gate The NOT gate switches between the two basis states.

$$\mathbf{X}\,|0\rangle = |1\rangle$$
$$\mathbf{X}\,|1\rangle = |0\rangle$$

The reason for representing the NOT gate by $\mathbf{X}$ is because it's matrix is given by

$$\mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

This is just one of the Pauli matrices, one which is usually represented by $\mathbf{X}$. We can check it's operation on the two basis states.

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

and

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

On an arbitrary state, we get

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}.$$

The other two Pauli matrices are gates in their own right, though their operation is slightly more complicated. We have

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

and

$$\mathbf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Hadamard Gate This is a special type of gate which takes a single basis state to a superposition of states.

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

It's matrix is given by

$$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}.$$

Notable about this gate is that it is self inverse, i.e. $H = H^{-1}$.

This means that

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

and

$$\mathbf{H}\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = |1\rangle\,.$$

Phase Gate This applies a phase difference to the qubit to only the $|1\rangle$ state, while leaving the $|0\rangle$ state unchanged.

$$\mathbf{R}\,|0\rangle = |1\rangle$$
$$\mathbf{R}\,|1\rangle = e^{\iota\phi}\,|1\rangle$$

It's matrix is given by

$$\begin{bmatrix}1 & 0\\ 0 & e^{i\phi}\end{bmatrix}.$$

3.2 Two Qubit and Higher Dimensional Gates

Two (and higher) qubit gates are essentially are of two types. One type is the analog of classical gates, where the output depends on all the inputs. The other type is what are termed 'controlled' gates. These have the feature that one of the input/output pair—called the target qubit—has an action done on it if and only if the other input/output pairs—called the control qubits—have a certain value. The control qubits are unaffected by the gate in either case. We look at a couple of such cases.

C-NOT Gate Here the C-NOT stands for 'Controlled NOT'. It applies the NOT gate to the target qubit if and only if the control qubit is $|1\rangle$. It's matrix representation is given by

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

This gate can equivalently be seen as one which XORs the value of the second qubit with the value of the first. Essentially, under the action of this gate,

$$|x\rangle |y\rangle \rightarrow |x\rangle |x \oplus y\rangle .$$

Controlled Phase Shift On the same lines as the C-NOT, this applies a phase shift to the target qubit if the controlled qubit has state $|1\rangle$.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & e^{i\phi} \end{bmatrix}.$$

Hadamard Gate The single qubit Hadamard Gate can be generalized to a n-qubit gate, where the single qubit Hadamard Gate is applied separately and individually to all the qubits.

$$\mathbf{H} |x_0\rangle |x_1\rangle |x_2\rangle \cdots |x_{n-1}\rangle \rightarrow (\mathbf{H} |x_0\rangle)(\mathbf{H} |x_1\rangle)(\mathbf{H} |x_2\rangle) \cdots (\mathbf{H} |x_{n-1}\rangle)$$

3.2.1 Function Evaluations

Gates can be stacked together in a quantum network to build up more complicated gates. Some of these gates can become the incarnation of functions. For instance, the C-NOT gate is single gate that allows us to XOR the two input values.

A more complicated example is shown in Fig. 2. This shows a quantum adder, that building on the same principles as the XOR, allows us to fully add two numbers.

Many times, in the course of quantum computation, we are in need of such quantum networks. Typically, we are not interested in how a particular function is implemented. Sometimes, we are interested in generic functions of only a certain type. In those case, we abstract away all details, and treat these functions as blackboxes. In other words, these functions can be expressed as quantum gates, which take in a certain number of inputs and transform them in a definite way to an output.

4 Quantum Algorithms

At the heart of computation lies algorithms, specific sets of instructions to compute the answers to specific problems. The set of instructions for a quantum computation, in

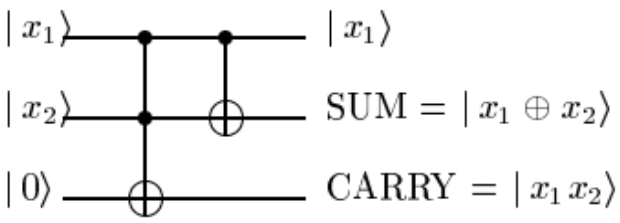

Figure 2: A quantum adder. The first gate is a Toffoli Gate, the second a C-NOT.

analogy to classical computation, involve qubits and quantum gates that are applied to them.

To be a little more specific, we can describe the scheme of things in the following way. A quantum network is defined to be a collection of qubits and gates in a specific order. For instance, Fig. 1 shows the network for the Quantum Fourier Transform. A network is the manifestation of an algorithm, in the sense that if it is fed with a certain input state, it outputs the answer to the problem the algorithm is designed to solve. To be useful, a network—quantum or otherwise—must have a well defined operation for all permissible input states.

A quantum computer can always be used to implement any classical algorithm[3]. However, that is not why we are interested in quantum computers. We would like to work with algorithms that can only be run on quantum computers, and that provide speedups over the corresponding classical algorithms.

Not a large number of such algorithms have been discovered or designed till now. The field of quantum computing is still young. However, a few very notable algorithms have been designed that have spearheaded much of the initial interest in quantum computing. The most important of these algorithms is Shor's factoring algorithm [8]. Designed by Peter Shor in 1994, it provides an exponential speedup over the best known classical algorithm for factoring a general large number. Another algorithm of interest is Groover's algorithm [11], a search algorithm that provides a quadratic speedup over the best known classical algorithms.

However, the very first quantum algorithm ever designed, a simple one qubit algorithm, was by David Deutsch [12]. We present here an improved version of the algorithm [13] and it's generalization to more qubits, before moving on to more complicate algorithms.

4.1 The Deutsch Algorithm

This quantum algorithm purports to solve the following problem.

We have a black box for computing an unknown function, $f : \{0,1\} \to \{0,1\}$. We determine the value of $f(0) \oplus f(1)$ by making one query to the black box. This allows

[3]Any algorithm at least that does not involve cloning states, since quantum states cannot be cloned.

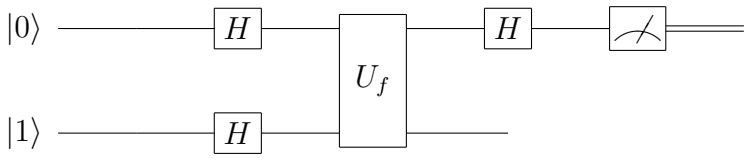

Figure 3: The Deutsch-Jozsa algorithm, a quantum circuit approach

us to distinguish whether the function f is balanced (i.e. zero and one are output with equal frequency) or constant (i.e. output is either one or zero always).

We can implement function evaluations as unitary transformations U_f acting on the states that represent the information.

$$\mathbf{U_f} \left|x\right\rangle = \left|f(x)\right\rangle$$

Input the quantum circuit given in Fig. 3 with input qubits 0 and 1 i.e. $\left|\psi_1\right\rangle = \left|0\right\rangle \left|1\right\rangle$. Apply the Hadamard gate to both bits one by one to obtain,

$$\begin{aligned}
\left|\psi_2\right\rangle &= \mathbf{H_1 H_2} \left|0\right\rangle \left|1\right\rangle \\
&= \frac{\left(\left|0\right\rangle + \left|1\right\rangle\right)}{\sqrt{2}} \frac{\left(\left|0\right\rangle - \left|1\right\rangle\right)}{\sqrt{2}} \\
&= \frac{1}{\sqrt{2}} \left|0\right\rangle \frac{\left(\left|0\right\rangle - \left|1\right\rangle\right)}{\sqrt{2}} + \frac{1}{\sqrt{2}} \left|1\right\rangle \frac{\left(\left|0\right\rangle - \left|1\right\rangle\right)}{\sqrt{2}}.
\end{aligned}$$

Now, apply the function operator U_f to this last state to give,

$$\begin{aligned}
\left|\psi_3\right\rangle &= U_f \left|\psi_2\right\rangle \\
&= \frac{(-1)^{f(0)}}{\sqrt{2}} \left|0\right\rangle \frac{\left(\left|0\right\rangle - \left|1\right\rangle\right)}{\sqrt{2}} + \frac{(-1)^{f(1)}}{\sqrt{2}} \left|1\right\rangle \frac{\left(\left|0\right\rangle - \left|1\right\rangle\right)}{\sqrt{2}}.
\end{aligned}$$

This can be simplified to,

$$= (-1)^{f(0)} \frac{\left(\left|0\right\rangle + (-1)^{f(0)\oplus f(1)} \left|1\right\rangle\right)}{\sqrt{2}} \frac{\left(\left|0\right\rangle - \left|1\right\rangle\right)}{\sqrt{2}}.$$

Now, there are two cases. If f is a constant function, we have $f(0) \oplus f(1) = 0$, which leads to after applying another Hamadard gate to the first quibit,

$$\psi_4 = (-1)^{f(0)} \left|0\right\rangle \frac{\left(\left|0\right\rangle - \left|1\right\rangle\right)}{\sqrt{2}}.$$

When f is balanced, we have $f(0) \oplus f(1) = 1$, which leads to,

$$\psi_4 = (-1)^{f(0)} \left|1\right\rangle \frac{\left(\left|0\right\rangle - \left|1\right\rangle\right)}{\sqrt{2}}.$$

So, now all we have to do is to measure the state of the first qubit. If it comes out as $\left|0\right\rangle$, f is constant. If it comes out as $\left|1\right\rangle$, f is balanced.

The measurement problem is not really a problem here. The final state can only be exactly one of the two basis states. It can't be in the superposition of states whatever the unknown function might be. Therefore, the measurement of the final state tells us with certainity what type of function it is.

The same problem solved on a classical computer would require at least two queries to the blackbox of the unknown function. Here we achieve the same task in just one query. This is because of the superposition property of quantum mechanics. The unknown function is being fed with the superposition of all possible input states, and so we can determine the output in just one query.

4.2 The Deutsch-Jozsa Algorithm

This is a generalization of the Deutsch algorithm given above. In this we deal with a slightly more complicated function, defined as,

$$f : \{0,1\}^n \to \{0,1\}.$$

As before, we try to determine if this function is balanced or constant. For the function to be balanced we mean that exactly half of the inputs yield 1 and the other half 0. For the function to be constant, we mean that they all take on the same value, either 0 or 1. We are guaranteed that the function is not any other type.

As before we implement the given function as a a unitary transformation, given by,

$$\hat{\mathbf{U}}_f : |\mathbf{x}\rangle\,|y\rangle \to |\mathbf{x}\rangle\,|y \oplus f(x)\rangle\,.$$

The $\mathbf{x}$ here represents the n-bit string. We can compute that $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ is an eigenstate of $\hat{\mathbf{U}}_f$ with eigenvalue $(-1)^f$. The circuit for this algorithm is given in Fig. 4.

We can represent the tensor product of n states in $|0\rangle$ by $|0\rangle^{\otimes n}$. Similarly, the tensor product of n 1-qubit Hadamard gates acting in parallel can be represented by $H^{\otimes n}$.

We start off with a the state $|\psi_1\rangle = |0\rangle\,|1\rangle$. We first appy a Hadamard gate to the last state and then $H^{\otimes n}$ to all the rest. This gives us,

$$\begin{aligned}
|\psi_2\rangle &= H^{\otimes n} H\,|\psi_1\rangle \\
&= H^{\otimes n}\,|0\rangle^{\otimes n} \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) \\
&= \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\epsilon\{0,1\}^n} |\mathbf{x}\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right).
\end{aligned}$$

Applying our function, we get,

$$|\psi_2\rangle = \frac{1}{\sqrt{2^n}} U_f \left(\sum_{\mathbf{x}\epsilon\{0,1\}^n} |\mathbf{x}\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)\right)$$

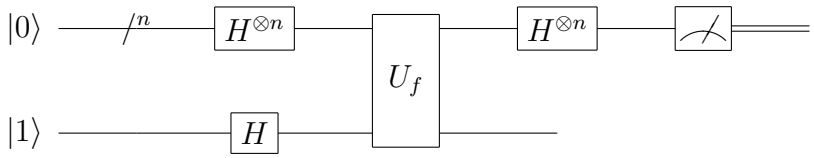

Figure 4: Deutsch-Jozsa Algorithm

$$= \frac{1}{\sqrt{2^n}} \sum_{\mathbf{x}\epsilon\{0,1\}^n} (-1)^{f(\mathbf{x})} |\mathbf{x}\rangle \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}} \right).$$

Now applying $H^{\otimes n}$ again,

$$|\psi_3\rangle = \frac{1}{\sqrt{2^N}} \sum_{\mathbf{x}\epsilon\{0,1\}^n} (-1)^{f(\mathbf{x})} \frac{1}{\sqrt{2^n}} \sum_{\mathbf{z}\epsilon\{0,1\}^n} (-1)^{\mathbf{x}\cdot\mathbf{z}} |\mathbf{z}\rangle .$$

Now, depending on whether, the value of the ground state, we can decide if the function is constant or balanced i.e. we look at,

$$2^{-n} \sum (-1)^{f(x)}$$

and decide if it's value is ± 1 for the case of f constant, or 0 for the case of f balanced.

4.3 Shor's Algorithm

Shor's algorithm is an integer factorization algorithm designed by Perter Shor in 1994 [8]. The fastest known classical factoring algorithm, the general number sieve works in subexponential time, sppecifically, $\bigcirc(e^{\log N}(\log\log N)^{2/3})$. Shor's algorithm achieves the same task in $\bigcirc((\log N)^3)$. This should be motivation enough to study the algorithm. We will shortly see how Shor's algorithm can be used to break RSA, the widely used public-key cryptographic scheme.

4.3.1 RSA

Let's go over RSA so that there are no notational confusions.

Suppose Bob picks a number $N = pq$ where p and q are two large primes. He then picks a random c such that $gcd(c, (p-1)(q-1)) = 1$. He also computes $d = c^{-1} \pmod{N}$. (N, c) is made available to the public.

Alice wants to send the message a to Bob. She computes $b \equiv a^c \pmod{()\ N}$, and sends it to Bob.

Bob computes $a' = b^d = a^{cd} \equiv a \pmod{N}$. In this way Bob gets a without anyone else finding out. The reason is as follows. The obvious method of recovering a from b is to compute d. However, to compute d one needs to know $(p-1)(q-1)$. This in turn requires factoring N, for which there is no known polynomial time algorithm.

Other attacks can also be made, some more and some less clever. However, there is no known classical polynomial algorithm for recovering the message. However, for a moment consider the following attack.

4.3.2 Breaking RSA through period finding

Suppose that one can somehow compute the order of b—the message that Alice sent—in the modular group N. Recall that the order of an element b mod N, is the smallest number r such that $b^r \equiv 1 \pmod{N}$. From number theory we know that r divides the order of the group $(p-1)(q-1)$.

We can also assert that the order of b in mod N is the same as the order of a. This is because $b^d \equiv a \pmod{N}$ and $a^c \equiv a \pmod{N}$. So, the subgroup generated by a contains b and the subgroup generated by b contains a. This can only happen if the two subgroups are equivalent. If they are equivalent then they have the same number of elements and so the order of a and b are the same. We have already computed the order of b. Now we know this is also the order of a.

We also know that c does not have any common factors with $(p-1)(q-1)$. Also, r divides $(p-1)(q-1)$. This implies that c and r have no common factors. We can do the following computation,

$$c \equiv c' \pmod{r}$$

There is a member d' in mod r such that,

$$c'd' \equiv 1 \pmod{r} .$$

But this means that,

$$\begin{aligned} cd' &\equiv 1 \pmod{r} \\ cd' &= 1 + mr \end{aligned}$$

The d' can be easily computed using euclidean algorithm. Then

$$\begin{aligned} b^{d'} &\equiv a^{cd'} \\ &\equiv a^{1+mr} \\ &\equiv a(a^r)^m \\ &\equiv a. \end{aligned}$$

In this way we have recovered a, the encrypted message. This of course depends on being able to find the period of b. However, there is no known polynomial time algorithm for doing this. This is where Shor's algorithm comes in.

4.3.3 Period finding through Shor's algorithm

We will need two registers of qubits to work with. They will be called the input and output registers respectively. The output register will be made of $n_0 = \log N$ qubits. The input register will contain twice this number, $n = 2n_0$. We start with all our registers in the state $|0\rangle$,.

$$|\psi_0\rangle = |0\rangle_{\otimes n} |0\rangle_{\otimes n_0} .$$

We apply the Quantum Fourier Transform(QFT) to the first register. QFT is in some senses similar to the classical fourier transform. Essentially, it transforms a given vector from one representation to another. The QFT is applied in the following way,

$$\hat{\mathbf{F}} |x\rangle_{\otimes n} = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i \frac{x}{2^n} y} |y\rangle_{\otimes n} .$$

There is also an inverse transform QFT^{-1} that gives,

$$\hat{\mathbf{F^{-1}}} \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i \frac{x}{2^n} y} |y\rangle = |x\rangle .$$

In our case, all the $x = 0$. So the exponential factor is always 1.

$$\begin{aligned} |\psi_1\rangle &= (\hat{\mathbf{F}} |0\rangle_{\otimes n}) |0\rangle_{\otimes n_0} \\ &= \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_{\otimes n} |0\rangle_{\otimes n_0} . \end{aligned}$$

Now we define $f(x) = b^x \pmod N$ and implement a quantum gate that implements this. More specifically, if we have a state like $|x\rangle |0\rangle$ then the quantum gate, $\hat{\mathbf{f}}$ acts on this state to give us $\hat{\mathbf{f}} |x\rangle |0\rangle = |x\rangle |f(x)\rangle$. This applied to our registers yields,

$$\begin{aligned} |\psi_2\rangle &= \mathbf{f}(\hat{\mathbf{x}}) |\psi_1\rangle \\ &= \frac{1}{2^{n/2}} \sum_{x=0}^{2^n-1} |x\rangle_{\otimes n} |f(x)\rangle_{\otimes n_0} . \end{aligned}$$

At this point we make a measurement of the output register. We get out some $f_0(x_r)$ where $x_r = x_0 + kr$ Here x_0 is the smallest value of x for which $f(x_0) = f(x_r)$. This is useful because of the measurement postulate the input register too is affected. It's state too changes and our overall state becomes,

$$|\psi_3\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle_{\otimes n} |f(x_r)\rangle_{\otimes n_0} .$$

Here m is the smallest integer such that $x_0 + mr \geq 2^n$. At this point we don't really care about the output state, so from this point on we will only write down the input state.

$$|\psi_4\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle_{\otimes n} .$$

We now apply QFT again to our input register.

$$\begin{aligned} |\psi_5\rangle &= \hat{\mathbf{F}} |\psi_4\rangle \\ &= \sum_{k=0}^{m-1} \frac{1}{\sqrt{m}} \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i (x_0+kr) y/2^n} |y\rangle_{\otimes n} . \end{aligned}$$

We can make a few rearrangments and break up the exponential,

$$= \sum_{y=0}^{2^n-1} \left(e^{2\pi i x_0 y/2^n} \frac{1}{\sqrt{m2^n}} \sum_{k=0}^{m-1} e^{2\pi i k r y/2^n} \right) |y\rangle_{\otimes n} .$$

Notice here is a case of an overall phase factor. Every $|y\rangle_{\otimes n}$ has the same factor associated with it, and that factor has it's modulus squared equal to 1 i.e. $|e^{2\pi i x_0 y/2^n}|^2 = 1$. Consequently, we can drop it from our state,

$$|\psi_6\rangle = \sum_{y=0}^{2^n-1} \left(\frac{1}{\sqrt{m2^n}} \sum_{k=0}^{m-1} e^{2\pi \iota k r y/2^n} \right) |y\rangle_{\otimes n} .$$

We make a measurement of the input register. This yields one of the values of y with probability give by the squared modulus of it's coefficient.

$$P(y) = \frac{1}{m2^n} \left| \sum_{k=0}^{m-1} e^{2\pi \iota k r y/2^n} \right|^2 .$$

We can do some calculus based analysis of the above expression to figure out when it's maximum [6]. However, in the interest of brevity we will skip it and only state the general argument. The exponential has it's maximum when it's argument is of the form $2\pi l$ where l is some integer. Our exponential will have maximums when y is close to an integer multiple of $2^n/r$. It can be shown that for integer j if $|y - \frac{j2^n}{r}| < \frac{1}{2}$, then the probability of obtaining such a y is at least 40%. We can repeat the quantum part effeciently and with near certainity obtain such a y.

Why is this useful? Notice the above expression can be rewritten as

$$|\frac{y}{2^n} - \frac{j}{r}| < \frac{1}{2^{n+1}}$$

Since n is large, and we know y and can compute 2^n, then $y/2^n$ is a good estimate of j/r. This is not as good as finding out r itself, but it's very close. There is a low chance that

any two random numbers would have a common factor. If j and r don't have a common factor than r is simply the numerator. If not we can either use a classical computer to compute multiples of the numerator and find out which one of them is r. Or we can run the quantum part all over again and get a new j/r.

To check if we have the right r we can simply compute $b^r \pmod{N}$. If it's equal to 1, we have our period.

We have broken the RSA!

4.3.4 Factoring with Shor's algorithm

Shor's algorithm is commonly described as a factoring algorithm. We now explain how knowing the period of a number mod N can be used to factor N.

The idea is simple. Suppose we want to factor $N = pq$. We simply pick a random a which is coprime to N. We then use Shor to compute the order of this $a \pmod{N}$. If this r is odd, we pick a new a. If it's even then we can calculate,

$$x = a^{r/2} \pmod{N} .$$

Then we have,

$$x^2 \equiv 1 \pmod{N} .$$

This gives us,

$$(x-1)(x+1) \equiv 0 \pmod{N} .$$

We know $(x-1) = a^{r/2} - 1 \neq 0$ since, r is the smallest power of a for which $a^r \equiv 1 \pmod{N}$. We check if $x + 1 = a^{r/2} + 1$ is equal to zero or not $\pmod{N}$. If it is, we have been unlucky and we start over with a new a. If it is not, then we can continue.

Now, since $N \nmid (x+1)$ and $N \nmid (x-1)$ but $N|(x-1)(x+1)$, then it must mean that $p|(x-1)$ and $q|(x+1)$. Then it only remains to compute $gcd(N, x+1)$ and $gcd(N, x-1)$, which will yield p and q.

5 Implementing Quantum Computers

Implementing practical quantum computation is no mean feat. It involves technological feats that have not been imvented yet. However, the great thing is that workers in the field have decided on a sufficiently general framework for the theoretical construction of quantum computers that it allows wide scope for the implementation of such systems. This has driven innovations in various directions. However, as yet no practical model has been decided as the preferred way of implementing quantum computers.

5.1 DiVincenzo Criteria

In 1997 David P. DiVincenzo [5] set up five essential criteria that any quantum computer must fulfill, so as to make the connection with the theoretical circuit model complete.

1. **Hilbert space control.** The Hilbert space we work with must be precisely definable i.e. we control the exact space we are working with. It must also be possible to scale the system. This means that more qubits can be added by a simple tensor product scheme.

2. **State preparation.** It must be possible to prepare qubits in some some simple known state from which computation can proceed.

3. **Low decoherence.** It must be possible to isolate our quantum computer or quantum mechanical system from the environment so that decoherence is low. There is no known analog of Shannon's noisy channel coding theorem. Consequently, our knowledge of acceptable decoherence level is dependant on the most effecient error correcting codes for quantum computation known.

4. **Implementation of quantum gates.** Quantum gates form a crucial part of quantum computers. Basic quantum gates are simply"controlled sequences of precisely defined unitary transformation" (DiVincenzo pg 5). They must be implemented with high precision and it must be possible to target specific groups of qubits to perform these translations on.

5. **State specific quantum measurements.** At the end of any quantum computations, and sometimes in between, measurements are made to read out the state of the system. It must be possible to be able to perform qubit specific measurements as well as on group of subsystems. Moreover, it must be possible to perform such measurements in whatever basis we choose.

References

[1] Feynman, R. (1982). “Simulating physics with computer”. International Journal of Theoretical Physics 21 (67): 467.

[2] Deutsch, D. (1985). “Quantum theory, the Church Turing principle, and the universal quantum computer”. Proc. Roy. Soc. Lond., A 400: 97117.

[3] Turing, A. M. (1936).“On Computable Numbers, with an Application to the Entscheidungsproble”. Proceedings of the London Mathematical Society. 2 42: 23065. 193637.

[4] Shannon, E.,Weaver, W. (1949). “The Mathematical Theory of Communication”. The University of Illinois Press, Urbana, Illinois.

[5] DiVincenzo, D., Loss, D. (1947). “Quantum Information is Physica”. Superlattices and Microstructures 23, 419 (1998).

[6] Nielsen, Michael A., Chuang, Isaac L. (2000). “Quantum Computation and Quantum Information”. Cambridge University Press.

[7] Kaye, P., Laflamme, R., Mosca M. (2007). “An introduction to quantum computing”, Oxford University Press..

[8] Shor, Peter W. (1997). “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Compute”. SIAM J. Comput. 26 (5): 14841509.

[9] Aarson, Scott. February 24th, 2007. ”Shor, Ill do it”. `http://www.scottaaronson.com/blog/?p=208`

[10] Breaking RSA Encryption with a Quantum Computer: Shor’s Factoring Algorithm, Lecture notes on Quantum computation, Cornell University, Physics 481-681, CS 483; Spring, 2006 by N. David Mermin.

[11] Grover L.K.: A fast quantum mechanical algorithm for database search, Proceedings, 28th Annual ACM Symposium on the Theory of Computing, (May 1996) p. 212

[12] Deutsch, D.,Jozsa, R. (1992).“Rapid solutions of problems by quantum computatio”. Proceedings of the Royal Society of London A 439: 553.

[13] Cleve R., Ekert A., Macchiavello C., Mosca M. (1998).“Quantum algorithms revisite”. Proceedings of the Royal Society of London A 454: 339354.