Quantum Computing

The field of quantum computing was pioneered in 1985 by Daved Deutsch. Building upon a suggestion by Feynman and the work of other scientists, he generalized the concept of the Turing Machine as postulated by Turing. 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

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 n 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 an 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. 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.

**Modelling Quantum Computers**

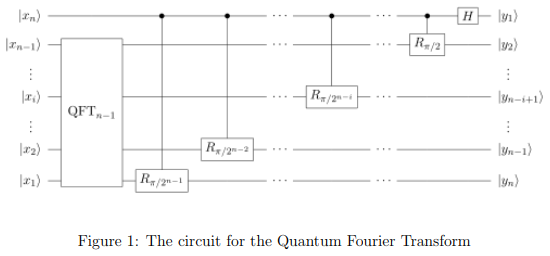
There are two fundamental ways that we can look at the theoretical construction of quantum computers.

**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.

**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.



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.

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> and |1>. This allows us to write the state of a single qubit as



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



And



This means that :



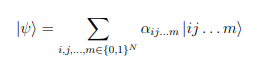
In other words, generic quantum states can be represented by column matrices. To represent a register of qubits we use the following scheme. For a two qubit system, the possible states are |00>, |01>, |10>, |11>. This means that an arbitrary state fo a two qubit system can be represented by



In column matrix formulation, the basis states are



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



**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 intocategories 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.

**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 I. In the first scheme, we have



This completely specifies the operation of the identity gate. Given the generic state

|ψ> = α |0> + β |1>, we can input it into the identity gate to get out



In the matrix representation, the I is given by the identity matrix



Multiply in any state with I will just return the original state.

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



The reason for representing the NOT gate by X is because it’s matrix is given by



This is just one of the Pauli matrices, one which is usually represented by X. We can check it’s operation on the two basis states



And



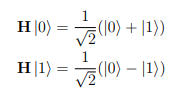
The other two Pauli matrices are gates in their own right, though their operation is slightly more complicated. We have



And



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

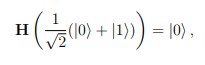


It’s matrix is given by



Notable about this gate is that it is self inverse, i.e. H = H−1.

This means that



And



**Phase Gate** This applies a phase difference to the qubit to only the |1i state, while

leaving the |0> state unchanged.



It’s matrix is given by



**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.

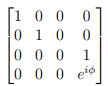
**C-NOT Gate** Here the C-NOT stands for ‘Controlled NOT’. It applies the NOT gate to the target qubit if and only if the control qubit is |1i. It’s matrix representation is given by



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,



**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>.



**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.



**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.

**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.

**DiVincenzo Criteria**

In 1997 David P. DiVincenzo 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 definablei.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”. 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.