3.1. Introduction

Quantum Computation and Quantum Information is the study of the information processing tasks that can be accomplished using quantum mechanical systems. Quantum mechanics are a mathematical framework or set of rules for the construction of physical theories [5, 6]. Quantum computation taught us to think physically about computation, and this approach yields many new and exciting capabilities for information processing and communication. One of the advantages of quantum computation and information is that new tools are available for those problems that are relatively more difficult or impossible to solve on classical computers. Quantum computing believes that what is computable and what is not computable is limited by the laws of physics.

Traditional computer science is based on Boolean logic and algorithms. Its basic variable is a bit with two possible values, 0 or 1. These values are represented in the computer as stable saturated states off or on. Quantum mechanics offer a new set of rules that go beyond this classical paradigm. The basic variable is now a qubit, represented as a vector in a two dimensional complex Hilbert space. $|0\rangle$ and $|1\rangle$ forms a basis in this space [9]. The logic that can be implemented with such qubits is quite distinct from Boolean logic, and this is what has made quantum computing exciting by opening new possibilities [19].

Despite this intense interest, efforts to build quantum information processing systems have resulted in modest success to date. Small quantum computers, capable of doing dozens of operations on a few qubits represent the state of the art in quantum
computation. However, it remains a great challenge to physicists and engineers to develop techniques for making large scale quantum information processing a reality.

3.2. Quantum bits

The bit is the fundamental concept of classical computation and classical information. Quantum computation and quantum information are built upon an analogous concept, the quantum bit, or qubit for short. We can describe qubits as mathematical object with certain specific properties. Qubits, like bits, can be realized as actual physical systems [22]. The beauty of treating qubits as mathematical objects is that it gives us the freedom to construct a general theory of quantum computation and quantum information, which does not depend upon a specific system for its realization.

3.2.1. Single Qubit

Just as a classical bit has a state either 0 or 1 a qubit also has a state. Two possible states for a qubit are the states $|0\rangle$ and $|1\rangle$, which correspond to the states 0 and 1 for a classical bit. Notation like ‘$| >$’ is called the Dirac Notation, and it is a standard notation for states in quantum mechanics. The difference between bits and qubits is that a qubit can be in a state other than $|0\rangle$ or $|1\rangle$. It is also possible to form linear combination of states, often called superposition

$$|\Psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

$$|\alpha|^2 + |\beta|^2 = 1 \quad (3.1)$$

The numbers $\alpha$ and $\beta$ are complex numbers, although for many purposes not much is lost by thinking of them as real numbers. Put another way, the state of a qubit is a
vector in a two dimensional complex vector space. The special states $|0\rangle$ and $|1\rangle$ are known as computational basis states, and form an orthonormal basis for this vector space.

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

(3.2)

We can examine a bit to determine whether it is in the state 0 or 1. For example, computers do this all the time when they retrieve the contents of their memory. Rather remarkably, we cannot examine a qubit to determine its quantum state, that is, the values of $\alpha$ and $\beta$. Instead, quantum mechanics tells us that we can only acquire much more restricted information about the quantum state. When we measure a qubit we get either the result 0, with probability $|\alpha|^2$, or the result 1, with probability $|\beta|^2$. Naturally, $|\alpha|^2+|\beta|^2=1$, since the probabilities must sum to one [13, 16]. Geometrically, we can interpret this as the condition that the qubit states be normalized to length 1. Thus, in general a qubit state is a unit vector in a two-dimensional complex vector space.

A classical bit is like a coin either heads or tails up. For imperfect coins, there may be intermediate states like having it balanced on an edge, but those can be disregarded in the ideal case. By contrast, a qubit can exist in a continuum of states between $|0\rangle$ and $|1\rangle$ until it is observed. Let us emphasize again that when a qubit is measured, it only ever gives 0 or 1 as the measurement result probabilistically. For example, a qubit can be in the state

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

(3.3)

which, when measured, gives the result 0 fifty percent ($\frac{1}{\sqrt{2}}$) of the time, and the result 1 fifty percent of the time. This state is sometimes denoted $|+\rangle$. 69
3.2.2. Multiple Qubit

Suppose we have two qubits. If these were two classical bits, then there would be four possible states, 00, 01, 10 and 11. Correspondingly, a two qubit system has four computational basis states denoted \( |00> \), \( |01> \), \( |10> \) and \( |11> \). A pair of qubits can also exist in superposition of these four states, so the quantum state of two qubits involves associating a complex coefficient sometimes called amplitude with each computational basis state, such that the state vector describing the two qubits is

\[
|\Psi\rangle = \alpha_{00}|00> + \alpha_{01}|01> + \alpha_{10}|10> + \alpha_{11}|11>
\]

\[
|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \otimes \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} \alpha_1 \beta_1 \\ \alpha_1 \beta_2 \\ \alpha_2 \beta_1 \\ \alpha_2 \beta_2 \end{pmatrix}
\]

(3.4)

Similar to the case for a single qubit, the measurement result \( x (=00, 01, 10 \text{ or } 11) \) occurs with the probability \( |\alpha_x|^2 \), with the state of the qubits after the measurement being \( |x> \). The condition that \( \Sigma_{x\in\{0,1\}}^2 |\alpha_x|^2 = 1 \), where the notation ‘\( \{0,1\}^2 \)’ means ‘the set of strings of length two with each letter being either zero or one’ [13,27]. For a two qubit system, we could measure just a subset of the qubits, say the first qubit, and we can probably guess how this works to measuring the first qubit alone gives 0 with probability \( |\alpha_{00}|^2 + |\alpha_{01}|^2 \), leaving the post measurement state [20].

3.3. Quantum Gates

Classical circuits consist of wires and logic gates. The wires are used to carry information around the circuit, while the logic gates perform manipulations of the information, converting it from one form to another.
3.3.1. Single Qubit gates

Consider, for example, classical single bit logic gate. The only non-trivial gate of this family is the NOT gate, whose operation is defined by its truth table, in which 0→1 and 1→0, that is, the states are interchanged. Can an analogous quantum NOT gate for qubits be defined? Imagine that we had some process, which took the state $|0\rangle$ to the state $|1\rangle$, and vice versa. Such a process would obviously be a good candidate for a quantum analogous to the NOT gate. However, specifying the action of the gate on the states $|0\rangle$ and $|1\rangle$ does not tell us what happens to superposition of the states $|0\rangle$ and $|1\rangle$, without further knowledge about the properties of quantum gates. In fact, the quantum NOT gate acts linearly, that is, it takes the state $\alpha|0\rangle+\beta|1\rangle$ to the corresponding state in which the roles of $|0\rangle$ and $|1\rangle$ have been interchanged, $\alpha|1\rangle+\beta|0\rangle$. There is a convenient way of representing the quantum NOT gate in matrix form, which follows directly from the linearity of quantum gates.

![One-qubit quantum circuit](image)

*Figure 3.1 One-qubit quantum circuit*

Notice that the action of the NOT gate is to take the state $|0\rangle$ and replace it by the state corresponding to the first column of the matrix $X$. Similarly, the state $|1\rangle$ is replaced by the state corresponding to the second column of the matrix $X$. 
Two important ones are the Z and H gates. Z gate leaves $|0\rangle$ unchanged, and flips the sign of $|1\rangle$ to give $-|1\rangle$. The H gate is the Hadamard gate [12, 13]. H gate is sometimes described as being like a square-root of NOT gate, in that it turns a $|0\rangle$ into $(|0\rangle + |1\rangle)/\sqrt{2}$ (first column of H), ‘halfway’ between $|0\rangle$ and $|1\rangle$ and turns $|1\rangle$ into $(|0\rangle - |1\rangle)/\sqrt{2}$ (second column of H), which is also ‘halfway’ between $|0\rangle$ and $|1\rangle$. Note, however, that $H^2$ is not a NOT gate, as simple algebra shows that $H^2=I$.

The Hadamard gate is one of the most useful quantum gates. It turns out that single qubit gates correspond to rotations and reflections of the sphere. The Hadamard operation is just a rotation of the sphere about the y-axis by $90^\circ$, followed by a reflection through the x-y plane. Some important single qubit gates are shown in figure.

There are infinitely many two by two unitary matrices, and thus infinitely many single qubit gates.
3.3.2. Multiple qubit gates

Now let us generalize from one to multiple qubits. Five notable multiple bit classical gates are AND, OR, XOR (exclusive-OR), NAND and NOR gates. An important theoretical result is that any function on bits can be computed from the composition of NAND gates alone, which is thus known as a universal gate. By contrast, the XOR alone or even together with NOT is universal. One way of seeing this is to note that applying an XOR gate does not change the total parity of the bit [36]. As a result, any circuit involving only NOT and XOR gates will, if two inputs $x$ and $y$ have the same parity, give outputs with the same parity, resulting the class of functions which may be computed, and thus precluding universality.

The prototypical multi-qubit quantum logic gate is controlled-NOT or CNOT gate. This gate has two input qubits, known as the control qubit and the target qubit, respectively. The circuit representation for the CNOT is shown in the Figure 3.4. The top line represents the control qubit, while the bottom line represents the target qubit. The action of the gate may be described as follows

$$U_{\text{CN}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

*Figure 3.4 Controlled-NOT gate and its matrix representation*
If the control qubit is set to 0, then the target qubit is left alone. If the control qubit is set to 1, then the target qubit is flipped. Shown in equations (3.5)

\[
\begin{align*}
|00\rangle &\rightarrow |00\rangle; \quad |01\rangle \rightarrow |01\rangle; \\
|10\rangle &\rightarrow |11\rangle; \quad |11\rangle \rightarrow |10\rangle.
\end{align*}
\tag{3.5}
\]

Another way of describing the CNOT is as a generalization of the classical XOR gate, since the action of the gate may be summarized as \(|A, B\rangle \rightarrow |A, B\oplus A\rangle\), where \(\oplus\) is addition modulo two, which is exactly what the XOR gate does. That is, the control qubit and the target qubit are XOR and stored in the target qubit. The CNOT can be regarded as a type of generalized-XOR gate. Any multiple qubit logic gates may be composed from CNOT and single qubit gates.

3.4. Quantum Circuits

We’ve already seen a few simple quantum circuits in previous Figures. Let’s look in a little more detail at the elements of a quantum circuit. A simple quantum circuit containing three quantum gates is shown in Figure 3.5. The circuit is to be read from left to right. Each line in the circuit represents a wire in the quantum circuit [19]. This wire does not necessarily correspond to a physical wire it may correspond to the passage of time or perhaps to a physical particle such as photon a particle of light moving from one location to another through space. It is convenient to assume that the state input to the circuit is a computational basis state, usually the state \(|0\rangle\) from Eq. 3.6.

The circuit in the figure accomplishes a simple but useful task it swaps the states of the two qubits. To see that this circuit accomplishes the swap operation, note that the
sequence of gates has the following sequence of effects on a computational basis state $|a, b>$,

$$|a, b> \rightarrow |a, a\oplus b>$$

$$\rightarrow |a \oplus (a\oplus b), a\oplus b>=|b, a\oplus b>$$

$$\rightarrow |b, (a\oplus b)\oplus b>=|b, a>,$$

where all additions are done modulo 2. The effect of the circuit, therefore, is to interchange the state of the two qubits.

![Quantum circuit for swapping two qubits.](image)

**Figure 3.5 Quantum circuit for swapping two qubits.**

There are a few features allowed in classical circuits that are not usually present in quantum circuits. First of all, we don’t allow loops, that is, feedback from one part of the quantum circuit to another; we say the circuit is acyclic. Second, classical circuits allow wires to be *joined* together, an operation known as *FANIN*, with the resulting single wire containing the bitwise OR of the inputs. Obviously this operation is not reversible and therefore not unitary, so we do allow FANIN in our quantum circuits. Third, the inverse operation, *FANOUT*, whereby several copies of bits are produced is also not allowed in quantum circuits.
3.5. Classical computations on a Quantum computer

This section illustrates of simulating classical logic circuit using a quantum circuit. Not surprisingly, the answer to this question turns out to be yes. It would be very surprising if this were not the case, as physicists believe that all aspects of the world around us, including classical logic circuits can ultimately be explained using quantum mechanics. The reason quantum circuits can not be directly used to simulate classical circuits is because unitary quantum logic gates are inherently reversible, whereas many classical logic gates such as the NAND gate are inherently irreversible [23].

Any classical circuit can be replaced by an equivalent circuit containing only reversible elements, by making use of a reversible gate known as the Toffoli gate. The Toffoli gate has three input bits and three output bits, as illustrated in the Figure 3.6. Two of the bits are control bits that are unaffected by the action of the Toffoli gate. The third bit is a target bit that is flipped if both control bits are set to 1, and otherwise is left alone.

<table>
<thead>
<tr>
<th>Inputs a</th>
<th>Inputs b</th>
<th>Inputs c</th>
<th>Outputs a'</th>
<th>Outputs b'</th>
<th>Outputs c'</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

![Figure 3.6 Truth table for Toffoli gate and its circuit representation](image)

Figure 3.6 Truth table for Toffoli gate and its circuit representation

Note that applying the Toffoli gate twice to a set of bits has the effect \((a,b,c)\rightarrow (a,b,c\oplus ab)\rightarrow(a,b,c)\), and thus the Toffoli gate is a reversible gate, since it has an inverse itself. The Toffoli gate can be used to simulate NAND gates, as shown. It is possible to
simulate all other elements in a classical circuit, and thus an equivalent reversible circuit can simulate an arbitrary classical circuit.

The Toffoli gate has been described as a classical gate, but it can also be implemented as a quantum logic gate. By definition, the quantum logic implementation of the Toffoli gate simply permutes computational basis states in the same way as the classical Toffoli gate [35]. For example, the quantum Toffoli gate acting on the state $|110>$ flips the third qubit because the first two are set, resulting in the state $|111>$. It is tedious but not difficult to write this transformation out as an 8 by 8 matrix, $U$, and verify explicitly that $U$ is a unitary matrix, and thus the Toffoli gate is a legitimate gate. The quantum Toffoli gate can be used to simulate irreversible classical logic gates.

3.6. Quantum Algorithms

The spectacular promise of quantum computers is to enable new algorithms (called quantum algorithms), that give solutions to problems requiring exorbitant resources for their solution on a classical computer. Two broad classes of quantum algorithms are known which fulfill this promise. The first class of algorithms is based upon Shor’s quantum Fourier transforms and includes remarkable algorithms for solving the factoring and discrete logarithm problem [35]. These algorithms provide a striking exponential speed up over the best known classical algorithms. The second class of algorithm is based upon Grover’s algorithm for performing quantum searching [23]. These provide a remarkable quadratic speed up over the best possible classical algorithms. The quantum search algorithms derive its importance from the widespread use of search based techniques in classical algorithms which in many instances allow a straightforward adaptation of the search for the given problem.

Most modern theoretical quantum algorithms rely on quantum property called quantum parallelism. Quantum parallelism arises from the ability of a quantum memory
register to exist in a superposition of base states. A quantum memory register can exist in a superposition of states, each component of this superposition may be thought of as a single argument to a function. A function performed on the register in a superposition of states is thus performed on each of the components of the superposition, but this function is only applied one time. Since the number of possible states is $2^n$ where $n$ is the number of qubits in the quantum register, we can perform in one operation on a quantum computer what would take an exponential number of operations on a classical computer. This is fantastic, but as the number of superposed states increases in the quantum register, the probability of measuring any particular one will decrease [19].

Quantum search algorithms have many potential applications. Specific aspects of quantum search algorithms can be applied to many problems in computer science to speed up algorithms for some problems in NP specifically, those problems for which a straight search for a solution is the best algorithm known. Quantum mechanics can greatly speed up searching process over unstructured database or randomly distributed data. In a quantum search process, the inspection of search space need not be carried out picking one item at a time from the search space instead the search can be applied directly to groups of items superposed together. Quantum computing requires to write programs in a completely new way some of which are completely meaningless with a classical computer e.g. superposition. A major breakthrough happened with Lov Grover’s very fast algorithm that is proven to be the fastest possible for searching through unstructured databases. The algorithm is so efficient that it requires roughly $\sqrt{N}$ (where $N$ is the total number of elements in the search space) searches to find the desired item, as opposed to search in classical computing, which on average requires $N/2$ searches. The quantum search algorithm offers only a quadratic speedup, as opposed to the more impressive exponential speedup offered by algorithms based on the quantum Fourier transform.
Grover's algorithm has another very useful application, in the field of cracking encrypted data. Quantum computers can crack DES (Data Encryption Standard) widely used system to protect data. An exhaustive search by conventional means would take a long time (even more than a year). But Grover's algorithm could find the DES enciphering key in just after few searches as compared to classical algorithm. Grover states that “A search engine could examine every nook and cranny of the Internet in half an hour, a brute force decoder could unscramble the DES transmission in five minutes”. These numbers are staggering compared to the fastest classical algorithm’s number of today.

Algorithm design for quantum computers is not easy. Designers face problems, not faced in the construction of algorithms for classical computers. The problem is that our human intuition is rooted in the classical world. If we use that intuition as an aid to the construction of algorithms, then the algorithmic ideas we come up with will be classical ideas. To design good quantum algorithms one must turn off one’s classical intuition for at least part of the design process. Only then we can achieve the desired algorithmic end [44].

3.6.1 Quantum Parallelism Hadamard Transformation

Quantum parallelism is fundamental feature of many quantum algorithms. Quantum parallelism allows quantum computers to evaluate a function f (x) for many different values of x simultaneously. Let us explain how quantum parallelism works, and some of its limitations.

Suppose f (x): {0,1}→{0,1} is a function with a one bit domain and range. A convenient way of computing this function on a quantum computer is to consider a two qubit quantum computer which starts in the state |x, y>. With an appropriate sequence of logic gates it is possible to transform this state into |x, y ⊕ f (x)> , where ⊕ indicates
addition modulo 2; the first register is called the ‘data’ register, and the second register the ‘target’ register [15]. It give the transformation defined by the map $|x, y\rangle \rightarrow |x, y \oplus f(x)\rangle$ a name $U_f$, and note that it is easily shown to be unitary. If $y=0$, then the final state of the second qubit is just the value $f(x)$.

Consider the function shown in Figure 3.7, which applies $U_f$ to input not in the computational basis. Instead, the data register is prepared in the superposition $(|0\rangle + |1\rangle)/\sqrt{2}$, which can be created with a Hadamard gate acting on $|0\rangle$. Then apply $U_f$ resulting in the state

$$\frac{|0, f(0)\rangle + |1, f(1)\rangle}{\sqrt{2}}$$

(3.7)

\[ 
\begin{array}{c}
|0\rangle + |1\rangle \\
\sqrt{2} \\
|0\rangle
\end{array}
\]

![Quantum circuit for evaluating f(0) and f(1) simultaneously](image)

Figure 3.7 Quantum circuit for evaluating $f(0)$ and $f(1)$ simultaneously

The different terms contain information about $f(0)$ and $f(1)$ it is almost as if we have evaluated $f(x)$ for two values of $x$ simultaneously, a feature known as quantum parallelism. Unlike classical parallelism, where multiple circuits each built to compute $f(x)$ are executed simultaneously, here a single $f(x)$ circuit is employed to evaluate the function for multiple values of $x$ simultaneously, by exploiting the ability of a quantum computer to be in superposition’s of different states.
This procedure can easily be generalized to functions on an arbitrary number of bits, by using a general operation known as the Hadamard transform, or sometimes the Walsh-Hadamard transform. This operation is just $n$ Hadamard gates acting in parallel on $n$ qubits. For example, shown in the figure 3.7 is the case $n=2$ with qubits initially prepared as $|0>$, which gives as output.

$$\frac{1}{\sqrt{2}}(|00>+|01>+|10>+|11>)/2.$$  \hspace{1cm} (3.8)

We write $H^\otimes 2$ to denote the parallel action of two Hadamard gates, and read ‘$\otimes$’ as tensor. More generally, the result of performing the Hadamard transform from “Eq. 3.8” on $n$ qubits initially in the $|0>$ state is

$$\frac{1}{\sqrt{2^n}}\sum_0^n |x>f(x)>$$ \hspace{1cm} (3.9)

where the sum is over all possible values of $x$, and we write $H^\otimes 2$ to denote this action. That is, the Hadamard transform produces an equal superposition of all computational basis states. Moreover, it does this extremely efficiently, producing a superposition of $2^n$ states using just $n$ gates.

Quantum parallel evaluation of a function with $n$ bit input $x$ and 1 bit output, $f(x)$, can thus be performed in the following manner. Prepare the $n+1$ qubit states $|0>^n|0>$, then apply the Hadamard transform to the first $n$ qubits, followed by the quantum circuit implementing $U_f$. This produces the state

$$\frac{1}{\sqrt{2^n}}\sum_0^n |x>f(x)>$$ \hspace{1cm} (3.10)

In some sense, quantum parallelism enables all possible values of the function $f(x)$ to be evaluated simultaneously, even though we apparently only evaluated $f(x)$ once.
However, this parallelism is not immediately useful. In single qubit example, measurement of the states gives only either $|0, f(0)>$ or $|1, f(1)>$. Similarly, in the general case, measurement of the state $\sum_x |x, f(x)>$ would give only $f(x)$ for a single value of $x$. Quantum computations requires something more than just quantum parallelism to be useful; it requires the ability to extract information about more than one value of $f(x)$ from superposition states like $\sum_x |x, f(x)>$.

3.6.2 The Deutsch-Jozsa Algorithm

This oracle generalizes the Deutsch oracle to a function $f: \{0,1\}^n \rightarrow \{0,1\}$. The function is constant or balanced in all the stages. Here balanced means that the function is 0 on half of its arguments and 1 on the other half [7, 8]. Of course in this case the function may be neither constant nor balanced. In this case the oracle doesn't work. It may say yes or no but the answer will be meaningless. Also here the algorithm allows one to evaluate a global property of the function in one measurement because the output state is a superposition of balanced and constant states such that the balanced states all lie in a subspace orthogonal to the constant states and can therefore be distinguished from the latter in a single measurement. In contrast, the best deterministic classical algorithm would require $2^n/2+1$ queries to the oracle in order to solve this problem [8].

3.6.3 Heisenberg Uncertainty Principle Algorithm

In 1984 Charles Bennett and Gilles Brassard published the first QKD protocol. It was based on Heisenberg's Uncertainty Principle and is simply known as the BB84 protocol after the author’s names and the year in which it was published. It is still one of the most prominent protocols and one could argue that all of the other HUP based protocols are essentially variants of the BB84 idea [13, 19]. The basic idea for all of these
protocols then is that Alice can transmit a random secret key to Bob by sending a string of photons where the secret key's bits are encoded in the polarization of the photons. Heisenberg’s Uncertainty Principle can be used to guarantee that an Eavesdropper cannot measure these photons and transmit them on to Bob without disturbing the photon's state in a detectable way thus revealing their presence.

3.6.4 Adiabatic Algorithms

More than a decade has passed since the discovery of the first quantum algorithm, but so far little progress has been made with respect to the “Holy Grail” of solving an NP-complete problem with a quantum circuit model [11]. As stressed above, Shor’s algorithm stands alone in its exponential “speed up”, yet while no efficient classical algorithm for factoring is known to exist, there is also no proof that such an algorithm doesn’t or cannot exist. In 2000 a group of physicists from MIT and Northeastern University proposed a novel paradigm for quantum computing that differs from the circuit model in several interesting ways [9, 11]. Their goal was to try to solve with this algorithm an instance of satisfiability deciding whether a proposition in the propositional calculus has a satisfying truth assignment which is one of the most famous NP complete problems [9, 10 & 11].

3.6.5 BB84 Six-State Protocol (SSP) Algorithm

Another variant of BB84 is the Six-State Protocol (SSP) proposed by Pasquinucci and Gisin in 1999. SSP is identical to BB84 except, as its name implies, rather than using two or four states, SSP uses six states on three orthogonal bases by which to encode the bits sent. This means that an eavesdropper would have to choose the right basis from among 3 possibilities. This extra choice causes the eavesdropper to produce a higher rate of error thus becoming easier to detect. Brus and Micchiavello proved in 2002 that such
higher-dimensional systems offer increased security. While there are a number of other BB84 variants, one of the more recent was proposed in 2004 by Scarani, Acin, Ribordy, and Gisin.

The SARG04 protocol shares the exact same first phase as BB84 [21]. In the second phase, when Alice and Bob determine for which bits their bases matched, Alice does not directly announce her bases. Rather she announces a pair of non orthogonal states, one of which she used to encode her bit. If Bob used the correct basis, he will measure the correct state. If he chose incorrectly, he will not measure either of Alice's states and he will not be able to determine the bit. BB84 was the first proposed QKD protocol and it was based on Heisenberg's Uncertainty Principle. A whole series of protocols followed which built on the ideas of BB84. Some of the most notable of these were B92, SSP, and Sarg04 [13, 21].

3.6.6 Shor's Algorithm

The oracle just described, although demonstrating the potential superiority of quantum computers over their classical counterparts, nevertheless deal with apparently unimportant computational problems [9, 10]. Indeed, it is doubtful whether the research field of quantum computing would have attracted so much attention and would have evolved to its current status if its merit could be demonstrated only with these problems. But in 1994, after realizing that Simon's oracle can be harnessed to solve a much more interesting and crucial problem, namely factoring, which lies at the heart of current cryptographic protocols such as the RSA, Peter Shor has turned quantum computing into one of the most exciting research domains in quantum mechanics [10].

Shor's algorithm exploits the ingenious number theoretic argument that two prime factors $p, q$ of a positive integer $N=pq$ can be found by determining the period of a function $f(x) = y^x \mod N$, for any $y < N$ which has no common factors with $N$ other than 1.
The period \( r \) of \( f(x) \) depends on \( y \) and \( N \). Once one knows the period, one can factor \( N \) if \( r \) is even and \( y^{r/2} \neq -1 \mod N \), which will be jointly the case with probability greater than 1/2 for any \( y \) chosen randomly (if not, one chooses another value of \( y \) and tries again). The factors of \( N \) are the greatest common divisors of \( y^{r/2} \pm 1 \) and \( N \), which can be found in polynomial time using the well known Euclidean algorithm [15]. In other words, Shor's remarkable result rests on the discovery that the problem of factoring reduces to the problem of finding the period of a certain periodic function \( f: Z_n \to Z_N \), where \( Z_n \) is the additive group of integers mod \( n \) (Note that \( f(x) = y^x \mod N \) so that \( f(x+r) = f(x) \) if \( x+r \leq n \). The function is periodic if \( r \) divides \( n \) exactly, otherwise it is almost periodic).

That this problem can be solved efficiently by a quantum computer is demonstrated with Simon's oracle [9][10].

Shor's result is the most dramatic example so far of quantum “speed-up” of computation, notwithstanding the fact that factoring is believed to be only in NP and not in NP complete [10]. To verify whether \( n \) is prime takes a number of steps which is a polynomial in \( \log_2 n \) (the binary encoding of a natural number \( n \) requires \( \log_2 n \) resources) [9, 10]. But nobody knows how to factor numbers into primes in polynomial time, not even on a probabilistic Turing machine, and the best classical algorithms we have for this problem are sub-exponential. This is yet another open problem in the theory of computational complexity. Modern cryptography and Internet security protocols such public key and electronic signature is based on these facts: It is easy to find large prime numbers fast, and it is hard to factor large composite numbers in any reasonable amount of time [10]. The discovery that quantum computers can solve factoring in polynomial time has had, therefore, a dramatic effect. The implementation of the algorithm on a physical machine would have economic, as well as scientific consequences [9].
3.6.7 Grover's Algorithm

Suppose we have met someone who kept the name secret, but revealed her telephone number. Can we find out her name using her number and a phone directory? In the worst case, if there are \( n \) entries in the directory, the computational resources required will be linear in \( n \). Grover (1996) showed how this task, namely, searching an unstructured database, could be done with a quantum algorithm with complexity of the order \( \sqrt{n} \). Agreed, this “speed-up” is more modest than Shor's since searching an unstructured database belongs to the class \( P \), but contrary to Shor's case, where the classical complexity of factoring is still unknown, here the superiority of the quantum algorithm, however modest, is definitely provable. That this quadratic “speed-up” is also the \textit{optimal} quantum speeds up possible for this problem [9, 11].

Although the purpose of Grover's algorithm is usually described as “searching a database”, it may be more accurate to describe it as “inverting a function”[15]. Grover's algorithm illustrates that in the quantum model searching can be done faster than this; in fact its time complexity \( O(N^{1/2}) \) is asymptotically the fastest possible for searching an unsorted database in the \textit{linear} quantum model. It provides a quadratic speedup, unlike other quantum algorithms, which may provide exponential speedup over their classical counterparts. However, even quadratic speedup is considerable when \( N \) is large. Unsorted search speeds of up to constant time are achievable in the \textit{nonlinear} quantum model if we have a function \( y=f(x) \) that can be evaluated easily on a quantum computer by Grover's algorithm which allows us to calculate \( x \) when given \( y \). Inverting a function is related to searching a database because we could come up with a function that produces a particular value of \( y \) if \( x \) matches a desired entry in a database, and another value of \( y \) for other values of \( x \). The applications of this algorithm are far reaching. For example, it can be used to determine efficiently the number of solutions to an \( N \) item search problem, hence
to perform exhaustive searches on a class of solutions to an NP complete problem and substantially reduce the computational resources required for solving it [9, 12].

3.7 Summary of Chapter

Quantum Computing is relatively new and an emerging area in the field of computing that taught us to think physically about computation. Quantum Computation and Quantum Information is the study of the information processing tasks that can be accomplished using quantum mechanical systems. Quantum Computing enables the development of new and different kind of algorithms, called quantum algorithms that are based on quantum principles. Specific aspects of quantum theory like superposition and quantum parallelism can be applied to the construction of such algorithms. Quantum search algorithms have many potential applications as they can greatly speed up searching process over randomly distributed data. By employing the extraordinary properties of quantum mechanical operations, like superposition and quantum parallelism, data can be searched with tremendous speed and efficiency.