STUDIA INFORMATICA
Nr 1(8) Systemy i technologie informacyjne 2007

The Quantum Computer - What Does It Means?

Janusz Kosiński
Academy of Podlasie, Institute of Computer Science

Abstract. In a classical measurement the Shannon information is a natural measure of our ignorance about properties of a system. There, observation removes that ignorance in revealing properties of the system which can be considered to preexist prior to and independent of observation. Because of the completely different root of a quantum measurement as compared to a classical measurement, conceptual difficulties arise when we try to define the information gain in a quantum measurement using the notion of Shannon information. In contrast to classical measurements, quantum measurements, with very few exceptions, cannot be claimed to reveal a property of the individual quantum system existing before the measurement is performed. A mathematical theory of computation that is based on quantum physics is bound to be different. They are the analogues for quantum computers to classical logic gates for conventional digital computers. Although quantum gates work on qubits in a much different fashion from standard electronic circuits, they only differ in their basic effects in one sense: reversibility.

Keywords: qubit, quantum gate, quantum algorithm, quantum computation

1 Introduction

In 1947, American computer engineer Howard Aiken[1] said that just six electronic digital computers would satisfy the computing needs of the United States[2]. Others have made similar errant predictions about the amount of computing power that would support our growing technological needs.

The large amounts of data generated by scientific research, the proliferation of personal computers or the emergence of the Internet, are growing larger need for more computing power. If, as Moore's Law states, the number of transistors on a microprocessor continues to double every 18 months, the year 2020 or 2030 will find the circuits on a microprocessor measured on an atomic scale. And the logical next step will be to create quantum computers, which will harness the power of atoms and molecules to perform memory and processing tasks.

In a classical measurement the Shannon information is a natural measure of our ignorance about properties of a system. There, observation removes that ignorance in revealing properties of the system which can be considered to preexist

[1] Howard Aiken and Grace Hopper designed the MARK series of computers at Harvard University. The MARK series of computers began with the Mark I in 1944.
[2] This remark is also attributed to Thomas J. Watson, the president of IBM

prior to and independent of observation. Today's computers, like a Turing machine, work by manipulating bits that exist in one of two states: a 0 or a 1.

Because of the completely different root of a quantum measurement as compared to a classical measurement, conceptual difficulties arise when we try to define the information gain in a quantum measurement using the notion of Shannon information. The reason is that, in contrast to classical measurements, quantum measurements, with very few exceptions, cannot be claimed to reveal a property of the individual quantum system existing before the measurement is performed.

Benjamin Schumacher discovered a way of interpreting quantum states as information. A theorem is proven for quantum information theory that is analogous to the noiseless coding theorem of classical information theory (*Schumacher, 1995*). In the quantum result, the von Neumann entropy of the density operator describing an ensemble of pure quantum signal states is equal to the number of spin - 1/2 systems. (‘‘quantum bits’’ or ‘‘qubits[3]’’) necessary to represent the signal faithfully. The theorem holds whether or not the signal states are orthogonal. Related results are also presented about the fidelity of quantum coding and about representing entangled[4] quantum states. Because a quantum computer can contain these multiple states simultaneously, it has the potential to be millions of times more powerful than today's most powerful supercomputers.

This superposition of qubits is what gives quantum computers their inherent parallelism. According to physicist David Deutsch, this parallelism allows a quantum computer to work on a million computations at once, while your desktop PC works on one. A 30-qubit quantum computer would equal the processing power of a conventional computer that could run at 10 teraflops (trillions of floating-point operations per second). Today's typical desktop computers run at speeds measured in gigaflops (billions of floating-point operations per second).

2 Defining the Quantum Computer

A quantum bit, or qubit ['kju.bit] is a unit of quantum information. That information is described by a state vector in a two-level quantum mechanical system which is formally equivalent to a two-dimensional vector space over the complex numbers. A qubit has some similarities to a classical bit, but is overall very different. Like a bit, a qubit can have only two possible values - normally a 0 or a 1. The difference is that whereas a bit must be either 0 or 1, a qubit can be 0, 1, or a superposition of both.

A qubit is often represented graphically by a sphere (Bloch sphere) with an arrow in it[5].

[3] Schumacher states that the term *qubit* was invented in jest, during his conversations with Bill Wootters.

[4] Two particles are called entangled if they share the same fuzzy quantum state, meaning neither of them begins with definite properties such as location or polarization (which can be thought of as a particle’s spatial orientation).

[5] In quantum mechanics, the Bloch sphere is a geometrical representation of the pure state space of a two-level quantum mechanical system named after the physicist Felix Bloch. Alternately, it is the pure state space of a 1 qubit quantum register.

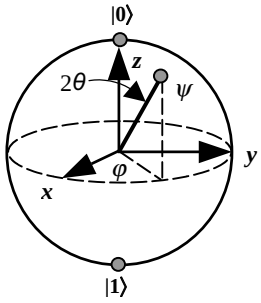


Figure 1. The Bloch sphere

The states a qubit may be measured in are known as basis states (or vectors). As is the tradition with any sort of quantum states, Dirac, or bra-ket[6] notation is used to represent them.

This means that the two computational basis states are conventionally written as $|0\rangle$ and $|1\rangle$ (pronounced: 'ket 0' and 'ket 1').

Arrow up corresponds to a classical 1. Arrow down corresponds to a classical 0. Arrow in between corresponds to a superposition of 1 and 0. Additionally the arrow may be rotated about the vertical axis.

Any state ψ can be written as a complex superposition of the ket vectors $|0\rangle$ and $|1\rangle$; moreover since phase factors do not affect physical state, we can take the representation so that the coefficient of $|0\rangle$ is real and non-negative. Thus ψ has a representation as:

$$|\psi\rangle = \cos\theta\,|0\rangle + e^{i\varphi}\sin\theta\,|1\rangle = \cos\theta\,|0\rangle + (\cos\varphi + i\sin\varphi)\sin\theta\,|1\rangle \qquad (1)$$

Except in the case ψ is one of the ket vectors $|0\rangle$ or $|1\rangle$, the representation is unique, i.e. the parameters φ and θ uniquely specify a point on the unit sphere of Euclidean space R^3 - the point whose coordinates (x, y, z) are:

$$\begin{aligned} x &= \sin 2\theta \times \cos\varphi \\ y &= \sin 2\theta \times \sin\varphi \\ z &= \cos 2\theta \end{aligned} \qquad (2)$$

In this representation $|0\rangle$ is mapped into (0, 0, 1) and $|1\rangle$ is mapped into (0, 0, -1).

The State Space Postulate[7] tells us that we can describe the most general state $|\psi\rangle$ of a single qubit by a vector of the form:

$$|\psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle \qquad (3)$$

1. Consider the analogous situation for a deterministic classical bit. The state of a classical bit can be described by a single binary value ψ, which can be equal to either 0 or 1.
2. Next consider the slightly more complicated situation of a classical bit whose value is not known exactly, but is known to be either 0 or 1 with corresponding

[6] Bra-ket notation is the standard notation for describing quantum states in the theory of quantum mechanics. It can also be used to denote abstract vectors and linear functionals in pure mathematics. It is so called because the inner product of two states is denoted by a bracket, $\langle\varphi|\psi\rangle$, consisting of a left part, $\langle\varphi|$ called the bra, and a right part, $|\psi\rangle$, called the ket. The notation was invented by Paul Dirac, and is also known as Dirac notation.

[7] The state of a system is described by a unit vector in a Hilbert space H.

probabilities p_0 and p_1. We might call this a probabilistic classical bit. The state of such a probabilistic bit is described by the probabilities p_0 and p_1, which satisfy $p_0 + p_1 = 1$ (reflecting the fact that we know the bit has to be either 0 or 1).

3. Now return to the state of a quantum bit, which is described by a complex unit vector $|\psi\rangle$ in a 2-dimensional Hilbert space. Such a state vector is often depicted as a point on the surface of a 3-dimensional sphere, known as the Bloch sphere.

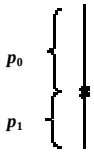
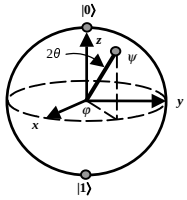
		
The state of a deterministic classical bit can be represented as one of two points, labeled “0” and “1”.	The probabilistic classical bit – Here the probabilities p_0 and p_1 of the bit being 0 and 1, are represented by the position of a point on the line segment between the points representing 0 and 1.	State of a qubit on the Bloch sphere

Figure 2. The representation of classical, probabilistic and quantum bits

The Turing machine, developed by Alan Turing in the 1930s, is a theoretical device that consists of tape of unlimited length that is divided into little squares. Each square can either hold a symbol (1 or 0) or be left blank. A read-write device reads these symbols and blanks, which gives the machine its instructions to perform a certain program. Does this sound familiar? Well, in a quantum Turing machine, the difference is that the tape exists in a quantum state, as does the read-write head. This means that the symbols on the tape can be either 0 or 1 or a superposition of 0 and 1; in other words the symbols are both 0 and 1 (and all points in between) at the same time. While a normal Turing machine can only perform one calculation at a time, a quantum Turing machine can perform many calculations at once.

The original Turing machine was deterministic (DTM): the head would be always in a single state, which would uniquely determine which direction it would go into and how far. There is a variant of the Turing machine, which is not deterministic. The head may be in a state, which gives the machine certain choices as to the direction and length of the next traverse. The choices are then made by throwing dice and possibly applying some weights to the outcome. A machine like that is called a probabilistic Turing machine (PTM), and it turns out that it is more powerful than the deterministic Turing machine in the sense that anything computable with DTM is also computable with PTM and usually faster. But both PTM and DTM are based on classical physics: the states of the tape and of the head are always readable and writable, data can be always copied, everything is uniquely defined.

A mathematical theory of computation that is based on quantum physics is bound to be different. As you move from classical physics to quantum physics there is a qualitative change in concepts that has profound ramifications.

So here's a brief history of how quantum Turing machine came about.

1973 - Bennett demonstrates that a reversible Turing machine is possible (*Bennett, 1973*)
1980 - Benioff observes that since quantum mechanics is reversible a computer based on quantum mechanical principles should be reversible too (*Benioff, 1980*).
1982 - Richard Feynman shows that no classical Turing machine can simulate quantum phenomena without an exponential slow down, and then observes that a universal quantum simulator can (*Feynman, 1982*).
1985 - David Deutsch of Oxford University, UK, describes the first true quantum Turing machine (*Deutsch, 1985*).

In the quantum Turing machine read, write, and shift operations are all accomplished by quantum interactions. The tape itself exists in a quantum state as does the head. In particular in place of the Turing cell on the tape that could hold either 0 or 1, in quantum Turing machine there is a qubit, which can hold a quantum superposition of 0 and 1. The quantum Turing machine can encode many inputs to a problem simultaneously, and then it can perform calculations on all the inputs at the same time. This is called quantum parallelism. The tape of the quantum Turing machine can now be drawn as shown in figure on the next page (*Meglicki, 1995*):

a) initial conditions

b) as the machine evolves, the head moves simultaneously in three different directions - the state of the machine becomes a superposition of the three states

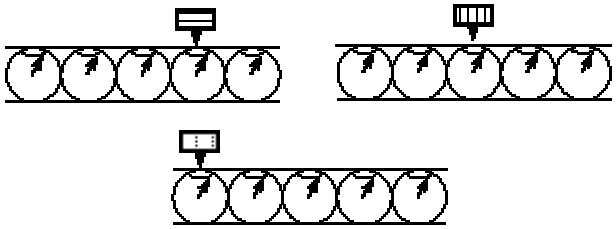


Figure 3. The tape of the quantum Turing machine.

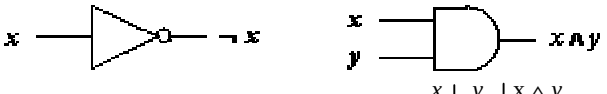
The machine evolves in many different directions simultaneously. After some time t its state is a superposition of all states that can be reached from the initial condition in that time. In the quantum Turing machine read, write, and shift operations are all accomplished by quantum interactions. The tape itself exists in a quantum state as does the head. In particular in place of the Turing cell on the tape that could hold either 0 or 1, in quantum Turing machine there is a qubit, which can hold a quantum superposition of 0 and 1. The quantum Turing machine can encode many inputs to a problem simultaneously, and then it can perform calculations on all the inputs at the same time. This is called quantum parallelism.

Quantum Turing machine can be used to simulate the classical Turing machine and the probabilistic Turing machine too. But quantum Turing machine can do more than that. For example it can generate truly random numbers, something that classical Turing machines cannot do.

Quantum parallelism is not easy to harness though. On measurement of final results the wave function of the computer must collapse, so that only a single result is delivered. On the other hand, it turns out that it is possible to measure certain joint properties of all the outputs.

3 Quantum gates

The theory of quantum computing is related to a theory of reversible computing. A computation is reversible if it is always possible to uniquely recover the input, given the output. For example, the NOT operation is reversible, because if the output bit is 0, you know the input bit must have been 1, and vice versa. On the other hand, the AND operation is not reversible (see figure below).



x	$\neg x$
0	1
1	0

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Figure 4. The standard gates

A quantum gate or a quantum logic gate is a basic quantum circuit operating on a small number of qubits. They are the analogues for quantum computers to classical logic gates for conventional digital computers. Although quantum gates work on qubits in a much different fashion from standard electronic circuits, they only differ in their basic effects in one sense: reversibility. Both types of gates take a bit, alter it, and give an output bit state. Quantum gates, however, have the additional property of reversibility[8]. The reversible AND gate keeps a copy of the inputs and adds the and of x_0 and x_1 (denoted $x_1 \wedge x_2$) to the value in the additional input bit. And note that by fixing the additional input bit to 0 and discarding the copies of the x_0 and x_1 we can simulate the non-reversible AND gate.

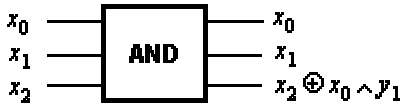


Figure 5. The reversible AND gate

[8] The reason that quantum gates are reversible is that their mechanism of action on qubits is through Schroedinger evolution (which is reversible by virtue of being unitary).

In classical computation, one could choose to be more environmentally friendly and uncompute redundant or junk information, and reuse the cleared-up memory for another computation. However, simply discarding the redundant information does not actually affect the outcome of the computation. In quantum computation however, discarding information that is correlated to the bits we keep can drastically change the outcome of a computation. For this reason, the theory of reversible computation plays an important role in the development of quantum algorithms. In a manner very similar to the classical case, reversible quantum operations can efficiently simulate non-reversible quantum operations (and sometimes vice versa) so we generally focus attention on reversible quantum gates. However, for the purposes of implementation or algorithm design, this is not always necessary (e.g. one can cleverly configure special families of non-reversible gates to efficiently simulate reversible ones).

Some universal classical logic gates, such as the Toffoli gate[9], provide reversibility and can be directly mapped onto quantum logic gates. Quantum logic gates are represented by unitary matrices[10]. The most common quantum gates operate on spaces of one or two qubits. This means that as matrices, quantum gates can be described by 2 × 2 or 4 × 4 matrices with orthonormal rows. Because the qubit is expressed as a vector, single qubit operators, or quantum gates, are expressed as 2×2 matrices. In order to be a valid operator, the result must still conform to $\alpha^2 + \beta^2 = 1$. It turns out that any matrix which transforms a source vector with that property to a result vector with that same property is said to be unitary, that is $H^tH = I$, where:

$$H = \begin{bmatrix} a+bi & c+di \\ e+fi & g+hi \end{bmatrix} \quad \text{and} \quad H^t = \begin{bmatrix} a-bi & e-fi \\ c-di & g-hi \end{bmatrix} \tag{4}$$

The requirement that the operator be unitary greatly restricts what can be done in quantum computing. Most classical operations are not unitary; because we can not get back the original value once the operation is performed. In fact, any operation that takes two bits and produces one is not unitary. The ability to reverse computation has practical implications. This creates an explosion in the number of bits required to perform computation, because the information necessary to reverse all operations performed must be part of the computation.

The quantum equivalents of such classical logic gates are achieved as simple unitary operations on qubits - *e.g.* see tables below.

[9] This gate has a 3-bit input and output. If the first two bits are set, it flips the third bit.

[10] A square matrix U is a unitary matrix if $U^H = U^{-1}$, where U^H denotes the conjugate transpose and U^{-1} is the matrix inverse. For example:

$$A = \begin{bmatrix} 2^{-1/2} & 2^{-1/2} & 0 \\ -2^{-1/2}i & 2^{-1/2}i & 0 \\ 0 & 0 & i \end{bmatrix}$$ is a unitary matrix

Table 1. The NOT gate

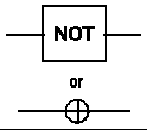
Symbol		The NOT gate is probably the most familiar to those used to dealing with classical circuit diagrams. It simply inverts the values of the qubit. It also has the effect of exchanging a and b in superpositioned states, exchanging the probability that the qubit will collapse to $\|0\rangle$ with the probability of collapse to $\|1\rangle$ and vice versa (that is $a\|0\rangle + b\|1\rangle$ becomes $b\|0\rangle + a\|1\rangle$).
Matrix	$NOT = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	
Truth table in Dirac notation	Input | Output $\|0\rangle$ | $\|1\rangle$ $\|1\rangle$ | $\|0\rangle$	

Table 2. The Hadamard gate

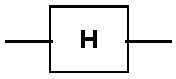
Symbol		Arguably the most important gate in quantum computation is the Walsh-Hadamard transformation gate (or Hadamard gate for short). Its function as a one-bit gate is to put the unsuperpositioned qubit into a superposition of the, $\|1\rangle$ and $\|0\rangle$ states. As the quantum computer derives much of its power from superposition-based activities, this gate is crucial.
Matrix	$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	
Truth table in Dirac notation	Input | Output $\|0\rangle$ | $\frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$ $\|1\rangle$ | $\frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$	

Table 3. Phase shift gate

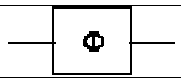
Symbol		Gates in this class operate on a single qubit. They are represented by 2 × 2 matrices of the form $R(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i\theta} \end{pmatrix}$ This has the computational effect of making two qubits that are in phase with each other move into another phase together.
Matrix	$Shift = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	
Truth table in Dirac notation	Input | Output $\|0\rangle$ | $\|0\rangle$ $\|1\rangle$ | $-\|1\rangle$	

Table 4. Square-root NOT gate

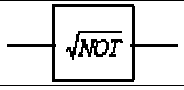
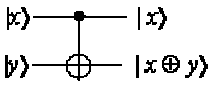
Symbol		The root NOT gate is exactly what it sounds like: the input is sent through two root NOT gates to NOT the input as the output. One advantage it has is that, unlike the NOT and CNOT gates, two root NOT gates can invert a superpositioned qubit (that is, the probabilities that the qubit will collapse to 1 is changed to the probability that that the qubit will collapse to 0).
Matrix	$\sqrt{NOT} = \begin{pmatrix} 1+i & 0 \\ 1-i & -1 \end{pmatrix}$	
Truth table in Dirac notation	Input | Output $\|0\rangle$ | $\frac{1}{2}(1+i)\|0\rangle + \frac{1}{2}(1-i)\|1\rangle$ $\|1\rangle$ | $\frac{1}{2}(1-i)\|0\rangle + \frac{1}{2}(1+i)\|1\rangle$	

Table 5. Controlled NOT gate

Symbol		Also known as the XOR or measurement gate, the controlled NOT (or CNOT) gate takes two qubits as input, $\|x\rangle$ and $\|y\rangle$. The result is the $\|x\rangle$ qubit and an XOR (⊗) of $\|x\rangle$ and $\|y\rangle$. If $\|x\rangle$ value is 0, $\|y\rangle$ remains the same; otherwise, $\|y\rangle$ value is flipped to its opposite

Matrix	$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	
Truth table	Control | Input | Output 1 | Output 2 0 | 0 | 0 | 0 0 | 1 | 0 | 1 1 | 0 | 1 | 1 1 | 1 | 1 | 0	

4 Quantum algorithms

Since quantum algorithms share some features with classical probabilistic algorithms, we will start with a comparison of the two algorithmic paradigms (*Kaye, et al., 2007*).

A classical probabilistic computation acting on a register that can be in one of four states labeled 0, 1, 2, 3. The $p_{0,j}$ are the probabilities for the computation proceeding from state 0 to state *j* in the first step. The $q_{j,k}$ represent the probabilities for the computation proceeding from state *j* to state *k* in the second step.

Suppose we want to find the total probability that the computation ends up in state 3 after the second step. (Fig. 6)

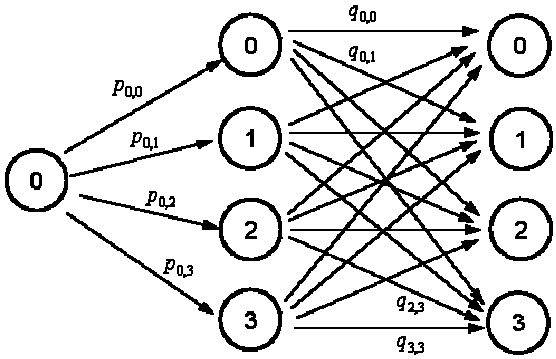


Figure 6. Finding the total probability in classical computation

This is calculated by first determining the probability associated with each computation 'path' that could end up at the state 3, and then by adding the

probabilities for all such paths. There are four computation paths that can leave the computation in state 3 after the first step. The computation can proceed from state 0 to state *j* and then from state *j* to state 3, for any of the four $j \in \{0, 1, 2, 3\}$. The probability associated with any one of these paths is obtained by multiplying the probability $p_{0,j}$ of the transition from state 0 to state *j*, with the probability $q_{j,3}$ of the transition from state j to state 3. The total probability of the computation ending up in state 3 is given by adding these four possibilities. So we have:

$$prob(\text{final outcome is } 3) = \sum_j p_{0,j} q_{j,3} \qquad (5)$$

Another way of looking at this computation is to suppose the register consists of two qubits, and let the labels 0, 1, 2, 3 refer to the four basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, respectively. Then view each of the transition probabilities as a squared norm of a quantum probability amplitude, so that $p_{0,j} = |\alpha_{0,j}|^2$ and $q_{j,k} = |\beta_{j,k}|^2$. This approach is shown in figure below, which can be viewed as a quantum computation in which the state is measured after each step.

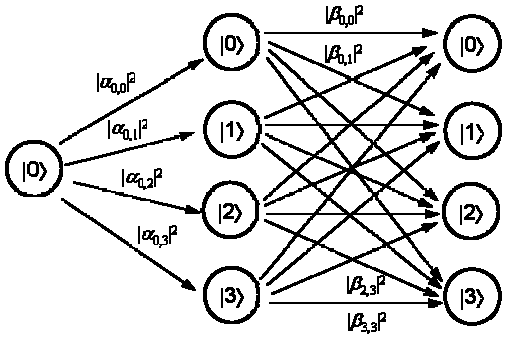


Figure 7. Finding the total probability in quantum computation

If we measured the state (in the computational basis) immediately after the first step of the computation, the probability associated with outcome 2 would be:

$$prob(\text{measurement after first step gives } 3) = \left|\alpha_{0,2}\right|^2 = p_{0,2} \qquad (6)$$

Since we assume that the state is measured after each step, we would know the intermediate state *j*, and thus we would know which computation path leading to the final state 3 was taken. The total probability of arriving at the final state 3 is determined by adding the squared norm of the probability amplitude $\alpha_{0,j}\beta_{j,3}$ associated with each path (*i.e.* we add the probabilities for the four paths, and not the probability amplitudes). As before, the total probability of measuring outcome 3 after the second step is:

$$prob(\text{final outcome is } 3) = \left|\alpha_{0,j}\right|^2 \left|\beta_{j,3}\right|^2 = \left|\alpha_{0,j}\beta_{j,3}\right|^2 \qquad (7)$$

In a fully quantum algorithm, we would not measure the state immediately after the first step. This way the quantum probability amplitudes will have a chance

to interfere. For example, some negative amplitude could cancel with some positive amplitude, significantly affecting the final probabilities associated with a given outcome. A quantum version of the algorithm above is illustrated in figure below.

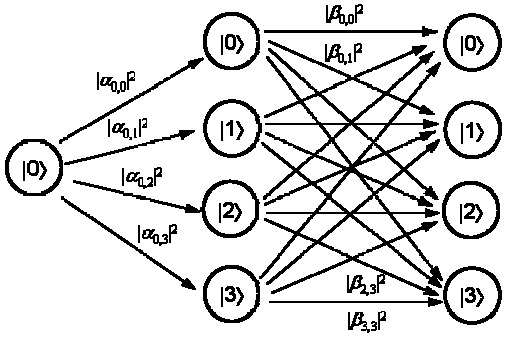


Figure 8. Fully quantum algorithm

This time the calculation of the total probability associated with outcome 3 in the measurement after the second step is different. Since there is no measurement after the first step of the computation, we do not learn the path taken by the computation to the final state 3. That is, when we obtain the output 3, we will have no information telling us which of the four paths was taken. In this case, instead of adding the probabilities associated with each of these four paths, we must add the probability amplitudes. The probability of a measurement after the second step giving the result 3 is obtained by taking the squared norm of the total probability amplitude. Classical probabilistic algorithms can be easily simulated by quantum algorithms. However, we know that naively replacing each quantum gate with a probabilistic classical gate can give drastically different outcomes, and thus will not work in general. And, there is no known general purpose classical algorithm for simulating quantum systems (and, in particular, quantum computers). As an example we look at the Deutsch algorithm. The Deutsch algorithm is a very simple example of a quantum algorithm based on the Quantum Fourier Transform[11]. The problem solved by the Deutsch algorithm is the following.

Suppose we are given a reversible circuit for computing an unknown 1-bit function $f : \{0, 1\} \rightarrow \{0, 1\}$. We treat this reversible circuit as a 'black box' or 'oracle'. This means that we can apply the circuit to obtain values of $f(x)$ for given inputs x, but we cannot gain any information about the inner workings of the circuit to learn about the function f. So determining $f(0) \oplus f(1)$ is equivalent to determining whether the function f is constant or balanced. How many queries to the oracle for f must be made classically to determine $f(0) \oplus f(1)$? Clearly the answer is 2.

Suppose we compute $f(0)$ using one (classical) query. Then the value of $f(1)$ could be 0, making $f(0) \oplus f(1) = 0$, or the value of $f(1)$ could be 1, making $f(0) \oplus f(1) = 1$. Without making a second query to the oracle to determine the value of $f(1)$, we

[11] The quantum Fourier transform is the discrete Fourier transform with a particular decomposition into a product of simpler unitary matrices. Using this decomposition, the discrete Fourier transform can be implemented as a quantum circuit consisting of Hadamard gates and controlled phase shift gates.

can make no conclusion about the value of $f(0) \oplus f(1)$. The Deutsch algorithm is a quantum algorithm capable of determining the value of $f(0) \oplus f(1)$ by making only a single query to a quantum oracle for *f*. The given reversible circuit for *f* can be made into a quantum circuit, by replacing every reversible classical gate in the given circuit with the analogous unitary quantum gate. This quantum circuit can be expressed as a unitary operator:

$$U_f : |x\rangle\,|y\rangle \rightarrow |x\rangle\,|y \oplus f(x)\rangle \qquad (8)$$

Having created a quantum version of the circuit for *f*, we can supply quantum bits as inputs. We define U_f, so that if we set the second input qubit to be in the state $|y\rangle = |0\rangle$, then $|x\rangle = |0\rangle$ in the first input qubit will give $|0 \oplus f(0)\rangle = |f(0)\rangle$ in the second output bit, and $|x\rangle = |1\rangle$ in the first input qubit will give $|f(1)\rangle$. So we can think of $|x\rangle = |0\rangle$ as a quantum version of the (classical) input bit 0, and $|x\rangle = |1\rangle$ as a quantum version of the input bit 1. Of course, the state of the input qubit can be some superposition of $|0\rangle$ and $|1\rangle$. A circuit, implementing the Deutsch algorithm is shown below.

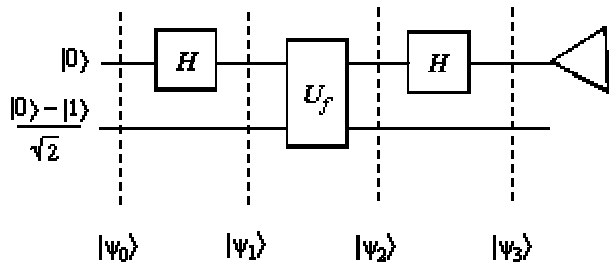


Figure 9. A circuit implementing the Deutsch algorithm

For the Deutsch problem we are ultimately not interested in individual values of $f(x)$, but wish to determine the value of $f(0) \oplus f(1)$.

The Deutsch algorithm illustrates how we can use quantum interference to obtain such global information about the function *f*, and how this can be done more efficiently than is possible classically.

Table 6. Some problems solved with quantum algorithms

The Deutsch–Jozsa Problem	Input: A black-box for computing an unknown function f : $\{0, 1\}^n \rightarrow \{0, 1\}$. Promise: f is either a constant or a balanced function. Problem: Determine whether f is constant or balanced by making queries to f.
Simon's Problem	Input: A black-box for computing an unknown function f : $\{0, 1\}^n \rightarrow X$, where X is some finite set. Promise: There exists a string $s = s_1 s_2 \ldots s_n$ so that $f(x) = f(y)$ if and only if $x = y$ or $x = y \oplus s$. Problem: Determine the string s by making queries to f

Phase Estimation Problem	Input: The state $\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1} e^{2\pi i \omega y}\left\|y\right\rangle$
	Problem: Obtain a good estimate of the phase parameter ω.
Eigenvalue Estimation Problem	Input: A quantum circuit implementing an operator U, and an eigenstate $\|\psi\rangle$ with corresponding eigenvalue $e^{2\pi i\omega}$. Problem: Obtain a good estimate for ω.
Order-Finding Problem	Input: Integers a and N such that GCD(*a*,N) = 1 *(i.e. a* is relatively prime to N). Problem: Find the order of a modulo N.
Integer Factorization Problem	Input: An integer N. Problem: Output positive integers $p_1, p_2, \ldots, p_l, r_1, r_2, \ldots, r_l$ where the p_i are distinct primes and $N = p_1^{r_1}, p_2^{r_2}, ..., p_l^{r_1}$
The Discrete Logarithm Problem	Input: Elements *b* and a = b^t in Z_p^*, where *t* is an integer from {0, 1, . . . , *r*−1} and *r* is the order of *a*. Problem: Find t. (The number t is called the discrete logarithm of *b* with respect to the base *a*.)
The Search Problem	Input: A black box U_f for computing an unknown function $f: \{0, 1\}n \rightarrow \{0, 1\}$. Problem: Find an input $x \in \{0, 1\}$ n such that $f(x) = 1$.

5 Qubit Control

Quantum computers also utilize another aspect of quantum mechanics known as entanglement. One problem with the idea of quantum computers is that if you try to look at the sub-atomic particles, you could bump them, and thereby change their value. If you look at a qubit in superposition to determine its value, the qubit will assume the value of either 0 or 1, but not both (effectively turning your spiffy quantum computer into a mundane digital computer).

To make a practical quantum computer, scientists have to devise ways of making measurements indirectly to preserve the system's integrity. Entanglement provides a potential answer. In quantum physics, if you apply an outside force to two atoms, it can cause them to become en-tangled, and the second atom can take on the properties of the first atom. So if left alone, an atom will spin in all directions. The instant it is disturbed it chooses one spin, or one value; and at the same time, the second entangled atom will choose an opposite spin, or value. Qubits represent atoms, ions, photons or electrons and their respective control devices that are working together to act as computer memory and a processor. All quantum computation exploits the quantum nature of an electron's spin or a photon's polarity. Quantum theory dictates that until these properties are actually observed, they are indeterminate: the spin of an electron, for instance, can be "up" or "down," or a combination of the two.

We can control the microscopic particles that act as qubits in quantum computers by using some control devices, *e.g.*:

- Ion traps - using optical or magnetic fields (or a combination of both) to trap ions.
- Optical traps - using light waves to trap and control particles.
- Quantum dots are made of semiconductor material and are used to contain and manipulate electrons.

5.1. Ion-trap

This technology uses electric and magnetic fields to isolate a charged particle from its environment - a prerequisite for exploiting the temperamental quantum properties of electrons. Although ion traps are just one technology for building a quantum computer, they have the longest history - the first trap was built in Monroe's lab[12] in 1995 - and they've advanced the furthest. But most ion traps are difficult to fabricate, consisting of a ceramic insulator and gold contacts for conducting an electrical current. Monroe's team built their chip out of insulating layers of alloys of aluminum, gallium, and arsenide, with semiconducting layers of gallium and arsenide - all easy to deposit on a chip using a conventional process called molecular beam epitaxy.[13]

The chip is placed in a vacuum, which then gets injected with a vapor of cadmium ions. When the appropriate voltages are applied to the electrodes, a cadmium ion with a free electron becomes trapped, floating between the cantilevers above the etched hole. In order to actually use the atom's free electron for computation, the ion must be probed by a laser beam that reads the electron's spin state. Putting multiple traps on one chip presents difficulties, because as the number of traps increase, it becomes more difficult for a laser to read the state of an individual electron without interfering with the state of other electrons.

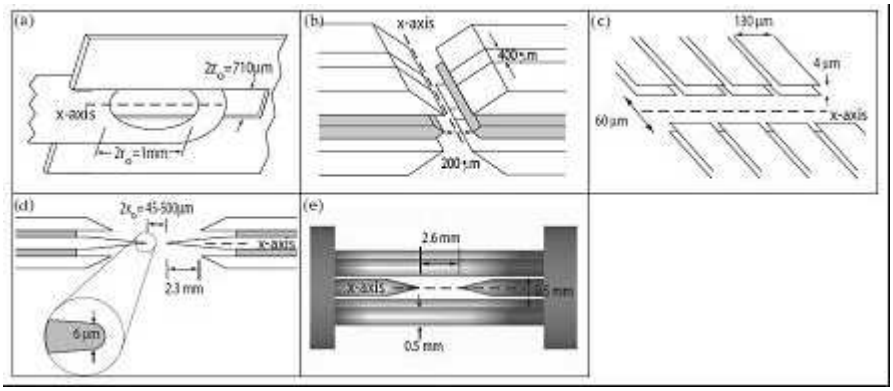


Figure 10. Examples of ion traps (*FOCUS Centre, 2006*): a) ring-fork quadrupole trap, b) three-layer linear trap, c) GaAs chip linear trap, d.) two-needle quadrupole trap, e) four-rod linear trap.

[12] Christopher Monroe's research group at the University of Michigan Department of Physics and FOCUS Physics Frontier Center.

[13] This first ion-trap chip builds on a quantum computing roadmap that Monroe, David Kielpinski at MIT, and David J. Wineland at NIST published in Nature in 2002.

5.2. Optical traps

Optical traps exploit the momentum of laser light to manipulate individual objects – not just beads, but also cells, organelles, and even atoms. After years of work, the Bell Labs group found they could hold a clear bead in place with a single, tightly focused laser beam, a surprisingly simple configuration now known as optical tweezers. If the bead starts to drift away, the laser light is deflected, and the particle is pushed in the opposite direction, back towards the focus. The result is that the bead is held gently in place, as if by tiny springs. When the experimenter moves the light beam, the bead follows, so optical tweezers can move objects around like their nonoptical namesake.

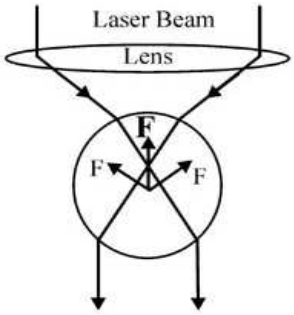


Figure 11. The model of optical tweezers. The incoming light is focused down to a tight spot and the forces that act on the sphere are such that the centre moves towards this focus. The 'F's in the diagram indicate the forces on the sphere

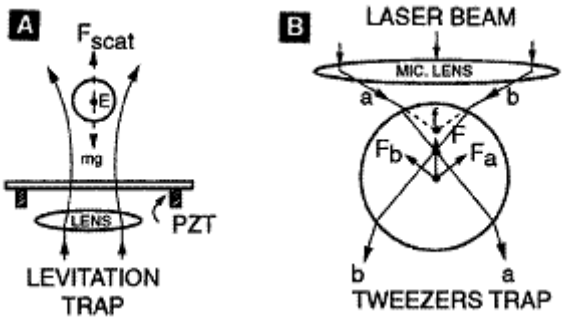


Figure 12. The example of optical trap (*Ashkin, 1997*): a) Geometry of levitation trap. b) Origin of backward restoring force F for sphere located below tweezers focus f

Optical tweezers use light to manipulate microscopic objects as small as a single atom. The radiation pressure from a focused laser beam is able to trap small particles. In the biological sciences, these instruments have been used to apply forces in the pN-range and to measure displacements in the nm range of objects ranging in size from 10 nm to over 100 mm.

The techniques of optical trapping and manipulation of neutral particles by lasers provide unique means to control the dynamics of small particles.

5.3. Quantum dots

Quantum dots are tiny nanocrystals that glow when stimulated by an external source such as ultraviolet (UV) light. They are unique class of semiconductor because they are so small, ranging from 2-10 nanometers (10-50 atoms) in diameter. In natural bulk semiconductor material, an extremely small percentage of electrons occupy the conduction band the overwhelming majority of electrons occupy the valence band, filling it almost completely.

The only way for an electron in the valence band to jump to the conduction band is to acquire enough energy to cross the band gap[14], and most electrons in bulk simply do not have enough energy to do so. It is also established that electrons in natural semiconductor bulk that have been raised into the conduction band will stay there only momentarily before falling back across the bandgap to their natural, valence energy levels. As the electron falls back down across the band gap, electromagnetic radiation with a wavelength corresponding to the energy it loses in the transition is emitted.

Because quantum dots' electron energy levels are discrete rather than continuous, the addition or subtraction of just a few atoms to the quantum dot has the effect of altering the boundaries of the bandgap. Changing the geometry of the surface of the quantum dot also changes the bandgap energy, owing again to the small size of the dot, and the effects of quantum confinement. As with bulk semiconductor material, electrons tend to make transitions near the edges of the bandgap. However, with quantum dots, the size of the bandgap is controlled simply by adjusting the size of the dot. (*Evident Technologies, 2007*)

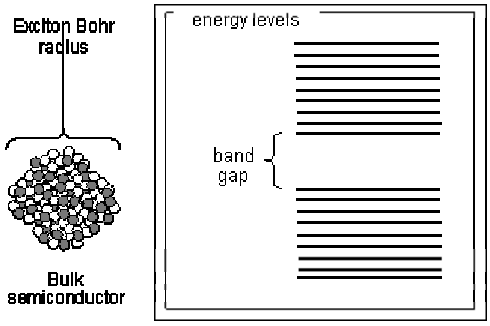


Figure 13. A tunable bandgap

[14] A sufficiently strong stimulus will cause a valence band electron to take residence in the conduction band, causing the creation of a positively charged hole in the valence band. The raised electron and the hole taken as a pair are called an *exciton.* Excitons have an average physical separation between the electron and hole, referred to as the Exciton Bohr Radius this physical distance is different for each material.

Quantum dots are small semi-conductor or metal islands with a diameter that is small enough to make their charging energy greater than k_BT where k_B is Boltzmann's constant and T is the operating temperature. The charging energy is the potential energy needed to overcome the electrostatic repulsion from the other electrons in the dot – or in other words, the energy required to add an electron to a dot. If this energy is greater than the thermal energy of the environment (k_BT), dots can trap individual charges.

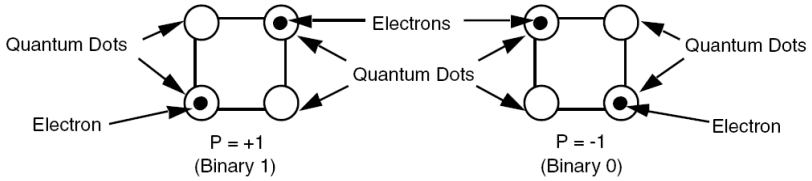


Figure 14. Cell polarizations and representations of binary 1 and binary 0 (*Shukla, 2004*)

Exactly two mobile electrons are loaded into cells and can move to different quantum dots by means of electron tunneling. Tunneling paths are represented by the lines connecting the quantum dots in figure below. Coulombic repulsion will cause "classical" models of the electrons to occupy only the corners of the cell, resulting in two specific polarizations. These polarizations are configurations where electrons are as far apart from one another as possible, in an energetically minimal position, without escaping the confines of the cell.

6 Today's Quantum Computers

Quantum computers could one day replace silicon chips, just like the transistor once replaced the vacuum tube. But for now, the technology required to develop such a quantum computer is beyond our reach. Most research in quantum computing is still very theoretical. The most advanced quantum computers have not gone beyond manipulating more than 16 qubits, meaning that they are a far cry from practical application. Several key advancements have been made in quantum computing in the last few years. See table below:

Table 7. The history of quantum computation growth

1998	Los Alamos and MIT researchers managed to spread a single qubit across three nuclear spins in each molecule of a liquid solution of alanine or trichloroethylene molecules. Spreading out the qubit made it harder to corrupt, allowing researchers to use entanglement to study interactions between states as an indirect method for analyzing the quantum information.
2000	In March, scientists at Los Alamos National Laboratory announced the development of a 7-qubit quantum computer within a single drop of liquid. The quantum computer uses nuclear magnetic resonance (NMR) to manipulate particles in the atomic nuclei of molecules of transcrotonic acid, a simple fluid consisting of molecules made up of

	six hydrogen and four carbon atoms. The NMR is used to apply electromagnetic pulses, which force the particles to line up. These particles in positions parallel or counter to the magnetic field allow the quantum computer to mimic the information-encoding of bits in digital computers.
2001	Scientists from IBM and Stanford University successfully demonstrated Shor's Algorithm on a quantum computer. Shor's Algorithm is a method for finding the prime factors of numbers (which plays an intrinsic role in cryptography). They used a 7-qubit computer to find the factors of 15. The computer correctly deduced that the prime factors were 3 and 5.
2005	The Institute of Quantum Optics and Quantum Information at the University of Innsbruck an-nounced that scientists had created the first qubyte, or series of 8 qubits, using ion traps.
2006	Scientists in Waterloo and Massachusetts devised methods for quantum control on a 12-qubit system. Quantum control becomes more complex as systems employ more qubits.
2007	Canadian startup company D-Wave demonstrated a 16-qubit quantum computer. The computer solved a sudoku puzzle and other pattern matching problems. The company claims it will produce practical systems by 2008. Skeptics believe practical quantum computers are still decades away, that the system D-Wave has created isn't scaleable, and that many of the claims on D-Wave's Web site are simply impossible (or at least impossible to know for certain given our understanding of quantum mechanics).

References

1. Ashkin A.; *Optical trapping and manipulation of neutral particles using lasers*; Research Department, Bell Laboratories, Lucent Technologies, Holmdel, 1997.
2. Benioff P.; *The Computer as a Physical System: A Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines*; Journal of Statistical Physics, Vol. 22, 1980.
3. Bennett C.; *Logical Reversibility of Computation*; IBM Journal of Research and Development, Vol. 17, 1973, pp. 525-532.
4. Bennett C.; *Logical Reversibility of Computation*; IBM Journal of Research and Development, Vol. 17, 1973, pp. 525-532.
5. Deutsch D.; *Quantum Theory, the Church-Turing Principle, and the Universal Quantum Computer*; Proceedings of the Royal Society of London, Vol. A400, 1985.
6. Evident Technologies; *Quantum dots;* Troy, New York, 2007; http://www.evidenttech.com/qdot-definition/quantum-dot-introduction.php
7. Feynman R. P.; *Simulating Physics with Computers*; International Journal of Theoretical Physics, Vol. 21, Nos. 6/7, 1982.
8. Kaye Ph., Laflamme R., Mosca M.; *An Introduction to Quantum Computing*; Oxford University Press Inc., New York, 2007.
9. Meglicki Z.; *Introduction to Quantum Computing*; Computer Based Learning Unit, University of Leeds, April 1995.

10. Monroe's C. - FOCUS Center; *Efficient photoionization loading of trapped ions with ultrafast pulses*;, Optical Physics Interdisciplinary Laboratory and Department of Physics, University of Michigan, 2006.
11. Schumacher B.; *Quantum coding*; Department of Physics, Kenyon College, Gambier, Ohio Phys. Rev. Issue 4 – April 1995.
12. Shukla K.S., Edit.; *Nano, Quantum and Molecular Computing Implications to High Level Design and Validation*; Virginia Polytechnic and State University, Blacksburg, U.S.A., 2004.

Erratum

In the Studia Informatica, Vol. 1/2(7) 2006 the following corrections should be noted. (1) Table of contents as a name of page 3 (instead of polish version), (2) The Tabu Search approach in coherent co-synthesis of multiprocessor systems, second author Czajkowski K. name was misspelled, (3) header on even pages from 36 to 44, both authors Drabowski M., Czajkowski K. names were misspelled, (4) Method of Logical Synthesis of Integrated Circuits in basis K-PLA, Novikov S., the figures from 1 to 10 were pass over – this Studia Informatica, Vol. 1(8) 2007 includes report under the title of Many-valued gates for reducing the chip-area of integrated circuits (by Novikov S.) as extended version of mentioned above with the same figures identically numbered.

We apologize for these mistakes.