13 Quantum computing

In this chapter we shall look at the basic principles of quantum computing and its implementation by optical techniques. Since this is a rapidly developing subject that occupies the attention of many research groups worldwide, we shall concentrate on introducing the fundamental ideas and avoid too many details that will inevitably date very quickly. The reader is referred to the bibliography for more rigorous treatments of the subject and more comprehensive discussions of the present state of the art in the experiments.

13.1 Introduction

Present-day computer technology is based on the silicon microprocessor chip. Silicon technology was first introduced in the 1960s, and has developed at a staggering rate that is familiar to everyone. The rapid development of the technology was noticed as early as 1965, when Gordon Moore, co-founder of the Intel Corporation, enunciated the law which now bears his name. **Moore's law** states that the number of transistors on a chip doubles every 18–24 months. The exponential growth that Moore's law predicts has held true for 30 years. Figure 13.1 shows the exponential progression of chip technology from the Intel 4004 introduced in 1971, which had 2250 transistors, through to the Pentium 4 introduced in 2000, which has 42 000 000.

The optimism that Moore's law engenders seems to hold no bounds. However, a closer look at the underlying principles reveals that the law must eventually break down at some time in the not-too-distant future. The progress in the chip technology has followed developments in the

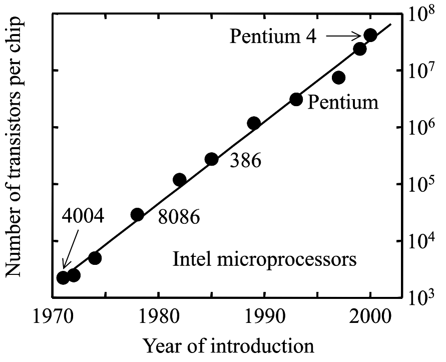

Fig. 13.1 Evolution of Intel microprocessors from the introduction of the 4004 chip in 1971. The graph shows the number of transistors per microprocessor against year of introduction on a log-linear scale. The straight line fit establishes the exponential growth predicted by Moore's Law.
(*Source*: Intel. See www.intel.com/technology/mooreslaw)

fabrication techniques that make it possible to produce transistors of ever-diminishing size. The transistors used in modern desktop computers are already less than 1 μm in dimension, and to maintain the progress, the size will have to continue to shrink. This makes it more and more difficult to produce the chips, leading to a similar exponential rise in the cost of the fabrication plants, a fact which is sometimes known as **Moore's second law**.

At the more fundamental level, an even more serious problem is going to be encountered soon because quantum effects will begin to become important when the size of the transistors becomes comparable to the de Broglie wavelength of the electrons that carry the signals. On these length-scales the physical laws that govern the circuit design such as Ohm's law no longer hold, and the circuits will no longer operate in the normal way. Even if this problem could be overcome by designing the circuits using quantum transport theory rather than the classical laws, we shall eventually hit another barrier when the size of the transistor becomes comparable to the size of the individual atoms. At this point the progression must stop, because we cannot realistically divide matter into smaller units than its constituent atoms.

Nobody knows for sure when Moore's law will break down. Moore himself has predicted that the end of the road will come around 2020. What is clear is that the law must eventually break down, and this will impose limits on the computational power that can be obtained by improving the existing technologies. For certain types of task, the failure of Moore's law will not lead to particular difficulties. The word-processors of tomorrow will continue to function even though the processing power of the chips will not be improving at the kind of rate that we are used to. However, for number-crunching tasks, the scale of the problems that can be tackled is always ultimately limited by the computer processing power that is available.

Computing tasks are generally classified according to the way in which they scale with the size. If the number of computer operations increases as a polynomial power of the size N, then the problem is said to belong to the **polynomial complexity class**, abbreviated to **P**. If, on the other hand, the number of operations increases faster than a polynomial function, then the problem is said to belong to the **non-polynomial complexity class** (**NP**). This difference is illustrated in Fig. 13.2, which compares the way a polynomial function of N, namely N^4, compares with a non-polynomial function, namely $\exp(N)$. As N gets larger, the non-polynomial functions always win eventually.

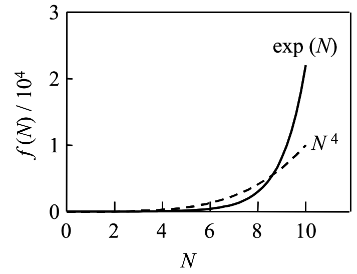

Fig. 13.2 Comparison of the size scaling of a polynomial function (N^4) with a non-polynomial function, namely $\exp(N)$.

Conventional computers are able to handle problems within the **P** class without too much difficulty. If the problem is too hard to solve today, then Moore's law tells us that we should be able to solve it soon, due to the exponential increase in processing power. On the other hand, problems in the **NP** class are always going to prove difficult. We only have to increase the size of the problem by a small amount to need a very large increase in the amount of computing power required. An important example of an **NP** problem is the factorization of large numbers. At present, the only way to find the prime factors of a large integer N is to divide N by all odd integers up to $\sqrt{N}$ to see if there is a remainder or not. Since the process of division takes of order N operations, we need an extra N operations each time we increase N by one. In other words, the number of operations required increases exponentially with N. Thus by increasing N, we enforce an exponentially increasing consumption of computer time for finding the factors. This is the basis of the security of the widely used RSA encryption scheme that we encountered in Section 12.1.

It has not been proven mathematically that an algorithm for efficient factorization does not exist. If such an algorithm were to exist, then factorization would reduce to the **P** class.

The difficulty that computer scientists meet when dealing with problems in the **NP** class stems from the escalating increase in computer time required as N increases, which makes the problem intractable in practice. All these statements presuppose that the computer scientists only have at their disposal a conventional computer which runs according to classical principles. These principles are modelled mathematically according to the operations of **universal Church–Turing machines**, (or **Turing machines** for short). The breakthrough in quantum computation came with the realization that other types of computer might exist that operate on completely different principles to Turing machines. In this case, the Turing machine should only be seen as the limiting case of more general types of computers that operate on the principles of quantum physics rather than classical physics.

See R. P. Feynman, *Int. J. Theor. Phys.* **21**, 467 (1982).

The idea of running a computer according to the laws of quantum mechanics was initially proposed by Richard Feynman in 1982. He pointed out that it gets progressively more difficult to simulate quantum systems with a conventional computer due to the exponential increase in processing power required as the system size increases. He therefore made the radical proposal that we ought to install quantum hardware in the computer, so that the computer's computational power would scale at the same rate as the complexity of the system that was being investigated. Three years later, David Deutsch wrote a theoretical paper which outlined the basic principles of quantum computation. In analogy with the Turing machine, he introduced the notion of a **universal quantum computer**, and showed that it could, in principle, solve problems that are not *efficiently* solvable with a classical computer.

See D. Deutsch, *Proc. R. Soc. London A*, **400**, 97 (1985). The phrase 'information is physical' is usually attributed to Rolf Landauer, and the idea can be traced back to his paper on 'Irreversibility and heat generation in the computing process' in *IBM J. Res. Dev.* **5**, 183 (1961). For a discussion of this concept in the context of quantum computing, see, for example, D. P. DiVincenzo and D. Loss *Superlattices and Microstructures* **23**, 419 (1998).

The revolutionary ideas of quantum computation involve a radical rethink about the way computers work. We have to realize that 'information is physical' in the sense that classical computers encode the bits of information in a variety of physical ways, such as the voltages on a transistor, the magnetization of a ferromagnetic material, or the intensity of a pulse of light. Although the underlying physics of transistors, ferromagnets, and light pulses are governed by quantum mechanics, the way the data is encoded is purely classical. Thus, for example, the voltage on the transistor has a well-defined value that can be uniquely determined according to the laws of classical electromagnetism. Deutsch's idea was to take a leap ahead and encode the information itself as quantum states which have no classical analogue. In doing so, we achieve an exponential increase in the computing power as the system gets larger. This comes about from exploiting the complexity of quantum systems to our advantage.

In the years that immediately followed Deutsch's landmark paper, the subject was mainly restricted to theoretical groups, who worked hard to understand the basic principles and advantages of quantum computing over conventional computational methods. Some groups concentrated on finding specific examples that would establish the general principle that quantum computers can outperform their classical counterparts, at least on paper. Others devoted their attention to designing experiments to prove the principles and establish that the ideas are more than just a theoretical dream.

A key breakthrough was made in 1994 when Peter Shor showed that a quantum computer can factorize a large number in polynomial time rather than exponential time. In this way, he reduced the factorization problem from the **NP** to the **P** complexity class. Since then, more examples have been found where quantum computers have an essential speed advantage over their classical counterparts. Meanwhile, the first generation of experiments has been completed, and several groups have now demonstrated baby quantum computers. Everyone realizes that it will take a long time for these baby quantum computers to grow to maturity and reach the point where they can really outperform their classical counterparts. At the same time, the potential benefits are enormous, and this prompts a forward-looking attitude in which new ideas are explored and developed, both experimentally and theoretically.

See P. W. Shor: Algorithms for quantum computation: Discrete logarithms and factoring, in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (ed. S. Goldwasser). IEEE Computer Society Press (1994), Los Alamitos, California, p. 124.

It is interesting to realize that quantum computation is based on the quantum 'weirdness' of superposition states. The superposition principle frequently causes conceptual difficulties when it is first encountered, and could be seen as an obstacle to information processing because it leads to probabilistic outcomes in measurements. In the subject of quantum computation we side-step the conceptual questions and take a pragmatic approach to exploit the parallelism of quantum states in a very practical way. In this way we bypass the philosophical questions and turn quantum mechanics into a practical subject that is used to enhance the possibilities of information science.

In the following sections we shall first introduce the basic concepts of quantum bits (qubits) and quantum logic gates. We shall then look at the problem of decoherence, and discuss some of the potential applications of quantum computers. Finally, we shall give a brief survey of some of the experimental work that has been performed so far, with particular stress on ion-trap systems.

13.2 Quantum bits (qubits)

13.2.1 The concept of qubits

Classical computers store information as binary bits that can take the value of logical 0 or 1. In analogy with their classical counterparts, quantum computers store the information as **quantum bits**, or **qubits** for short. These are quantum-mechanical states of individual particles such as atoms, photons, or nuclei. The key difference between classical and quantum bits is that qubits can not only represent pure 0 and 1 states, but they can also take on superposition states, in which the system is in both the 0 and 1 state at the same time. This is a consequence of the superposition principle of quantum mechanics, and contrasts with classical systems which can only ever be in one of the two possible states at a given moment. (See Section 9.2.2.)

The properties of qubits are governed by their quantum-mechanical wave function ψ. We choose physical systems that have two readily distinguishable quantum states that can be used to represent binary 0 and 1. If we use Dirac notation to label the quantum states corresponding to 0 and 1 as $|0\rangle$ and $|1\rangle$ respectively, then the general state of the qubit can be written in the following form:

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle, \tag{13.1}$$

where the normalization condition on $|\psi\rangle$ requires that

$$|c_0|^2 + |c_1|^2 = 1. \tag{13.2}$$

Equation 13.1 explicitly expresses the fact that the system is in a superposition of both $|0\rangle$ and $|1\rangle$ states at the same time. The relative proportion of each of the binary states is governed by the amplitude coefficients c_0 and c_1.

In order to clarify what we understand by qubits, it is helpful to discuss the kinds of physical system that might comprise the quantum hardware. Table 13.1 lists some of the most important systems that have been considered in this context. In each case we have an individual quantum system with two clearly distinguishable states. In order for the system to be usable, we require that the chosen property should be easily measurable, and that the two states are orthogonal to each other, such that:

$$\langle 0|1\rangle = 0. \tag{13.3}$$

The examples given in Table 13.1 all satisfy these criteria.

Table 13.1 Some physical realizations of qubits. In the case of the superconducting loop, the direction of the magnetic flux quantum is determined by the direction of the persistent current.

Quantum system	Physical property	$\|0\rangle$	$\|1\rangle$
Photon	Linear polarization	Horizontal	Vertical
Photon	Circular polarization	Left	Right
Nucleus	Spin	Up	Down
Electron	Spin	Up	Down
Two-level atom	Excitation state	Ground state	Excited state
Josephson junction	Electric charge	N Cooper pairs	$N+1$ Cooper pairs
Superconducting loop	Magnetic flux	Up	Down

Let us suppose that we choose to use the linear polarization of an individual photon as the basis for the qubit states. In this case, we could define the $|0\rangle$ and $|1\rangle$ states to correspond to the horizontal and vertical polarization states, respectively. An arbitrary state of the qubit would then be given by the wave function $|\psi\rangle$ with

We have already considered photon qubits of this type in the discussion of quantum cryptography in Chapter 12.

$$\begin{aligned}|\psi\rangle &= c_0|0\rangle + c_1|1\rangle \\ &\equiv c_0|\leftrightarrow\rangle + c_1|\updownarrow\rangle,\end{aligned} \tag{13.4}$$

where we used the same notation for the polarization states as in Table 12.1. The quantum information of the qubit is stored in the amplitude coefficients c_0 and c_1. These coefficients can be calculated precisely, but cannot be measured directly. Thus, for example, measurements using the apparatus shown in Fig. 12.2 give the result 0 with probability $|c_0|^2$ and 1 with probability $|c_1|^2$, so that repeated measurement permit the determination of $|c_i|^2$, but not c_i. It therefore seems that the quantum information is hidden, and that we gain little by moving over to the quantum technology. This is indeed the case if we only have one qubit: the advantages of the quantum technology only emerge when we have several qubits.

A collection of N qubits is called a **quantum register** of size N. Consider a two-qubit register. The wave function for an arbitrary state is specified as a superposition of the four possible combinations of states of the individual qubits:

The easiest way to make an N-qubit system (at least, conceptually) is to couple together N two-level particles. It is also possible to use different energy levels of a single particle as different qubits. Note that some qubits (e.g. excitons, flux qubits) are collective quantum excitations rather than real particles.

$$|\psi\rangle = c_{00}|00\rangle + c_{01}|01\rangle + c_{10}|10\rangle + c_{11}|11\rangle, \tag{13.5}$$

where the notation $|ij\rangle$ implies that qubit 1 is in state i and qubit 2 is in state j. This can be generalized to any number of qubits. Thus a three-qubit register would have a general wave function of the form:

$$\begin{aligned}|\psi\rangle = {} & c_{000}|000\rangle + c_{001}|001\rangle + c_{010}|010\rangle + c_{011}|011\rangle \\ & + c_{100}|100\rangle + c_{101}|101\rangle + c_{110}|110\rangle + c_{111}|111\rangle.\end{aligned} \tag{13.6}$$

It is apparent that an N-qubit register is described by 2^N wave function amplitudes $c_{ijk\ldots}$. The quantum information is stored in these amplitudes, which are complex numbers with a modulus between 0 and 1. The amount of information clearly grows exponentially with the register size, but the information is hidden and a large amount of it is lost when measurements are made. However, provided we only manipulate the qubits and let them interact with each other coherently without making measurements, then all the information is preserved. This is the basis of the huge quantum parallelism that underlies quantum computation. The clever part of the subject is to devise methods to harness the parallelism. We shall give some examples of how this is done in Section 13.5.

13.2.2 Bloch vector representation of single qubits

The normalization condition written in eqn 13.2 suggests that we can represent the state of a single qubit as a vector. This vector is called the

Bloch vector and has been discussed previously in Section 9.6 in the context of two-level atoms. The Bloch vector maps out a sphere of unit radius called the **Bloch sphere**, as illustrated in Fig. 13.3. Points on the Bloch sphere are specified by their polar angles (θ, φ). The North pole $(\theta = 0)$ and South pole $(\theta = \pi)$ of the sphere are defined to correspond to the pure $|1\rangle$ and $|0\rangle$ states, respectively. All other values of θ correspond to superposition states of the type given in eqn 13.1.

Note that there has been a change of notation here compared to Section 9.6. In the treatment of two-level atoms it is customary to label the lower and upper levels as 1 and 2, respectively, whereas here we are using the labels 0 and 1 instead in order to make the link with binary logic. Note also that some authors put $|1\rangle$ at the South pole and $|0\rangle$ at the North pole, interchanging c_0 and c_1 in eqn 13.7. This choice is purely a matter of convention, and has no physical significance.

The correspondence between the amplitude coefficients and polar angles can be made explicit by setting (cf. eqn 9.64)

$$\begin{aligned} c_0 &= \sin(\theta/2), \\ c_1 &= \mathrm{e}^{\mathrm{i}\varphi}\cos(\theta/2). \end{aligned} \tag{13.7}$$

We shall see in Section 13.3 below that the Bloch sphere model is very helpful in understanding the effect of quantum operations on qubits.

13.2.3 Column vector representation of qubits

Another useful way to describe the state of a single qubit with a wave function given by eqn 13.1 is as a column vector of the form:

The one- and two- qubit wave functions can also be conveniently represented as the row vectors (c_0, c_1) and $(c_{00}, c_{01}, c_{10}, c_{11})$, respectively.

$$|\psi\rangle = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}. \tag{13.8}$$

This column vector notation allows us to use 2×2 matrices to represent the operations that are performed on the qubits, which simplifies the formal treatment. Furthermore, it provides a convenient way to handle multiple qubits. For example, we can represent a two-qubit system with a wave function of the type given in eqn 13.5 as a column vector of the form:

$$|\psi\rangle = \begin{pmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{pmatrix}, \tag{13.9}$$

We are then able to use 4×4 matrices to represent the operations that manipulate the two-dimensional qubits, as we shall see in Section 13.3.3.

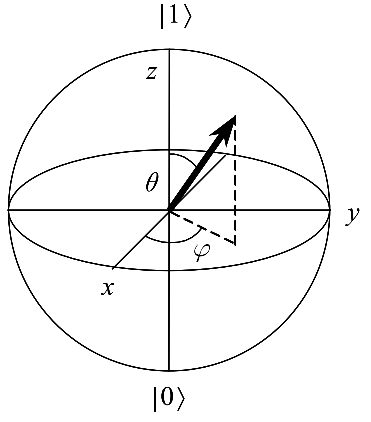

Fig. 13.3 The Bloch sphere representation of qubits. Qubit states correspond to points on the surface of the sphere, with $|0\rangle$ at the South pole, $|1\rangle$ at the North pole, and superposition states everywhere else.

13.3 Quantum logic gates and circuits

13.3.1 Preliminary concepts

A classical computer consists of a memory and a processor. The processor carries out operations on the bits of information stored in the memory according to a program, and outputs the results as a new set of bits. The processing operations are performed by millions of simple **binary logic gates** such as the NOT or NAND gates. These perform operations on either one or two bits at a time. For example, the NOT gate operates on one bit at a time, while the NAND gate operates on

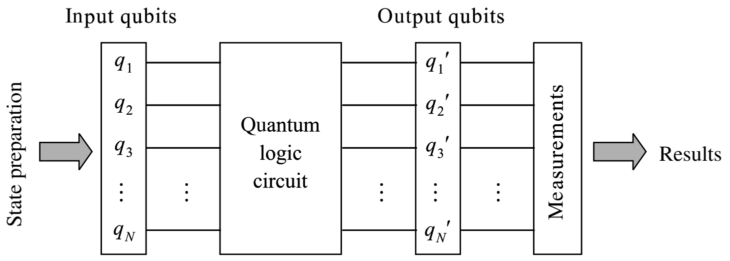

Fig. 13.4 Schematic block diagram of the workings of a quantum computer. The qubits $\{q_1, q_2, q_3, \ldots, q_N\}$ from the input register are set up in the correct initial states and are fed into the quantum logic circuit. The quantum logic circuit performs the processing tasks and outputs a new set of qubits $\{q_1', q_2', q_3', \ldots, q_N'\}$. Measurements are made on the output register and the results are then read out.

two bits. The truth tables for these classical operations are given in Tables 13.2 and 13.3. The program determines how the binary gates are linked together in a logical circuit in order to perform the required task.

Table 13.2 Truth table for the classical single-bit NOT gate.

Input bit	Output bit
0	1
1	0

The basic idea of a quantum computer is much the same. The information is stored in a register of qubits and the processing tasks are carried out by **quantum logic gates**. These quantum logic gates are connected together in a **quantum circuit** in order to carry out specified processing tasks. Figure 13.4 shows a schematic block diagram of a quantum computer. We start with an N-qubit register $\{q_1, q_2, q_3, \ldots, q_N\}$, in which the qubits have previously been prepared in the required initial states. These input qubits are fed into the quantum logic circuit which then performs the processing task according to the program of the quantum computer. The output of the quantum logic circuit is a new set of qubits $\{q_1', q_2', q_3', \ldots, q_N'\}$. The final results of the computational task are obtained by making measurements on these output qubits, which return a set of N classical bits.

Table 13.3 Truth table for the classical two-bit NAND gate.

Input bits		Output bit
0	0	1
1	0	1
0	1	1
1	1	0

At this stage it appears that we have gained nothing from the quantum calculation. We started with a 'data set' of N qubits and ended up with a result consisting of N classical bits. However, the key point to understand is that with N input qubits we are effectively entering 2^N data points into the computer. If we program the quantum processor intelligently, we can obtain information from the input data set more efficiently than we would with a classical machine. We shall see examples of this in Section 13.5. Thus the benefit of the quantum computer over its classical counterpart comes from the manipulation of the 2^N amplitude coefficients of the N input qubits within the quantum logic circuit before the final measurements are made.

It is clear from the above that the heart of a quantum computer is the quantum logic circuit that performs the information-processing task. The quantum logic circuit consists of a programmed sequence of simple quantum logic gates. Just as with classical computers, it turns out that we only need a very small number of quantum logic gates to perform all the possible computing tasks. We first need a series of **single-qubit**

gates which perform operations on one qubit at a time. Then we need one **two-qubit gate** which operates on two qubits at a time. With these basic building blocks we can implement any quantum logic circuit that we may require. Our task therefore is to understand both single- and two-qubit gates, beginning with the single-qubit gates.

See Nielsen and Chuang (2000, §4.5).

13.3.2 Single-qubit gates

The operation of a single-qubit gate is shown schematically in Fig. 13.5. A single qubit q is fed into the gate, and the gate outputs another qubit q'. If we write the wave functions of q and q' as $|\psi\rangle$ and $|\psi'\rangle$, respectively, with

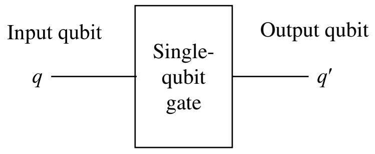

Fig. 13.5 Schematic diagram of a single-qubit gate. The gate transforms an input qubit q to an output qubit q'.

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle, \tag{13.10}$$

and

$$|\psi'\rangle = c_0'|0\rangle + c_1'|1\rangle, \tag{13.11}$$

then we see that the effect of the gate is to change the amplitude coefficients of the qubit in a determined way. By making use of the column vector notation defined in eqn 13.8, we can describe the gate by a 2×2 matrix $\mathbf{M}$ as follows:

$$\begin{pmatrix} c_0' \\ c_1' \end{pmatrix} = \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix}, \tag{13.12}$$

with

$$\begin{aligned} c_0' &= M_{11}c_0 + M_{12}c_1 \\ c_1' &= M_{21}c_0 + M_{22}c_1. \end{aligned} \tag{13.13}$$

It turns out that the only requirement on the gate matrix $\mathbf{M}$ is that it should be **unitary**:

$$\mathbf{M}\mathbf{M}^\dagger = \mathbf{I}, \tag{13.14}$$

where $\mathbf{M}^\dagger$ is the **adjoint** matrix of $\mathbf{M}$, and $\mathbf{I}$ is the identity matrix. This condition can be written explicitly as:

$$\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} \begin{pmatrix} M_{11}^* & M_{21}^* \\ M_{12}^* & M_{22}^* \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{13.15}$$

The unitarity requirement implies that all quantum gates must be *reversible.* (See Exercise 13.2.)

Table 13.4 Single-qubit gates. Note that the X and Z gate matrices are identical to their respective Pauli spin matrices, which is one of the reasons why the gates are labelled 'X' and 'Z' in the first place. The other reason relates to their geometric interpretation as rotation operators about the x- and z-axes, respectively.

Quantum gate	Matrix representation
NOT (X)	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Hadamard (H)	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$

Three of the most important single-qubit gates are listed in Table 13.4. The **NOT gate**, which is represented by the 'X' symbol, switches the amplitude coefficients around:

$$X \cdot q = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_0 \end{pmatrix}. \tag{13.16}$$

The Z **gate** flips the sign of $|1\rangle$, while leaving $|0\rangle$ unchanged:

$$Z \cdot q = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} c_0 \\ -c_1 \end{pmatrix}. \tag{13.17}$$

Finally, the **Hadamard gate** (H gate) turns basis states into superposition states, and vice versa:

$$H \cdot q = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} (c_0 + c_1)/\sqrt{2} \\ (c_0 - c_1)/\sqrt{2} \end{pmatrix}. \qquad (13.18)$$

Thus, for example, the H gate maps the $|0\rangle$ and $|1\rangle$ states onto the $(|0\rangle + |1\rangle)/\sqrt{2}$ and $(|0\rangle - |1\rangle)/\sqrt{2}$ superposition states, respectively, whereas it turns superposition states like $(|0\rangle + |1\rangle)/\sqrt{2}$ into basis states:

$$H \cdot \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \qquad (13.19)$$

We explained in Section 13.2.2 that the state of a qubit can be mapped onto a point on the surface of the Bloch sphere. By changing the amplitude coefficients of the qubit, a single-qubit gate alters the position of the qubit on the Bloch sphere. The quantum gates can therefore be given a geometrical interpretation. For example, the X gate is equivalent to a rotation of π radians about the x-axis. This can be seen by considering a few examples of the effect of the X gate:

$$\begin{aligned} |0\rangle &\rightarrow |1\rangle, \\ |1\rangle &\rightarrow |0\rangle, \\ (1/\sqrt{2})(|0\rangle + |1\rangle) &\rightarrow (1/\sqrt{2})(|0\rangle + |1\rangle), \\ (1/\sqrt{2})(|0\rangle + \mathrm{i}|1\rangle) &\rightarrow (1/\sqrt{2})\, \mathrm{e}^{\mathrm{i}\pi/2}(|0\rangle - \mathrm{i}|1\rangle). \end{aligned}$$

In a single-qubit system, phase factors like the one in the output qubit of the fourth example are unmeasurable. Note, however, that *relative* phase shifts are significant in multiple qubit systems.

The equivalent operations in terms of the Bloch vectors are as follows:

$$\begin{aligned} (0,0,-1) &\rightarrow (0,0,1), \\ (0,0,1) &\rightarrow (0,0,-1), \\ (1,0,0) &\rightarrow (1,0,0), \\ (0,1,0) &\rightarrow (0,-1,0). \end{aligned}$$

The first two of these operations are illustrated in Figs 13.6(a) and (b).

In the same way, the Z gate is equivalent to a rotation of π about the z-axis, while the H gate is equivalent to a rotation of π about the z-axis

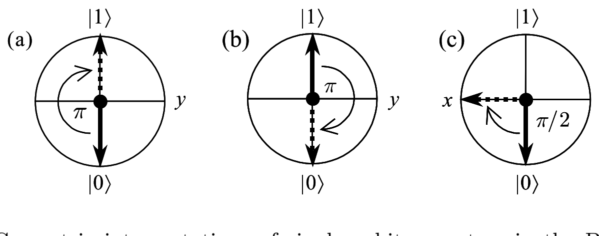

Fig. 13.6 Geometric interpretations of single-qubit operators in the Bloch sphere representation. (a) X operator on the $|0\rangle$ state observed in the y-z plane. (b) X operator on the $|1\rangle$ state observed in the y-z plane. (c) H operator on the $|0\rangle$ state observed in the x-z plane. In (c) the first rotation about the z-axis has no effect in this particular example.

followed by a rotation of $\pi/2$ about the y-axis. Figure 13.6(c) shows the effect of an H gate on the $|0\rangle$ qubit.

Example 13.1 A qubit in the $|0\rangle$ state is input to an H gate followed by a Z gate. What is the output qubit?

Solution
The output qubit is calculated by applying the operation matrices in the correct order to the input qubit:

$$q' = Z \cdot H \cdot q,$$

with $q = (1, 0)$. Written explicitly, we have:

$$q' = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}.$$

The output qubit is thus:

$$q' = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.$$

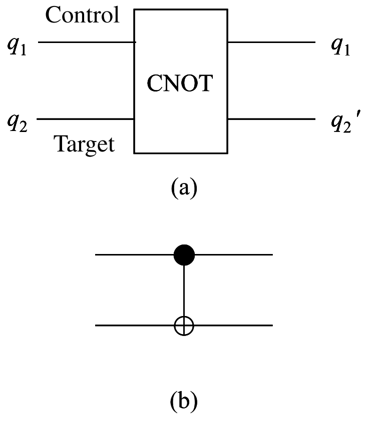

Fig. 13.7 The controlled-NOT (C-NOT) gate. (a) The gate changes the target qubit q_2 depending on the state of the control qubit q_1. (b) Symbol used to represent C-NOT gates in quantum circuits.

13.3.3 Two-qubit gates

We mentioned in Section 13.3.1 that any arbitrary qubit gate can be built up from a sequence of single-qubit gates and one type of two-qubit gate. A particularly useful type of two-qubit gate is the **controlled unitary operator** (C-U) gate. C-U gates have two input qubits, which are designated as the **control** and **target** qubits, respectively. The gate has no effect on the control qubit, but performs a unitary operation on the target qubit conditionally on the state of the control qubit. Since we only need to develop one type of two-qubit gate to build a quantum computer, we can learn all the basic principles by restricting our discussion to the simplest one, namely the **controlled-NOT gate** (C-NOT gate).

Other common examples of two-qubit gates are the controlled rotation gate (C-ROT) and the controlled phase shift gate (C-PHASE).

Figure 13.7 shows a schematic diagram of a C-NOT gate, together with the symbol that represents the gate in quantum circuits. In a C-NOT gate, the controlled unitary operation is the NOT gate. The control and target qubits are designated q_1 and q_2, respectively, and the gate carries out the NOT operation on q_2 if $q_1 = |1\rangle$. The truth table for the C-NOT gate is given in Table 13.5.

Table 13.5 Truth table for the controlled-NOT (C-NOT) operation.

Input qubits		Output qubits	
Control	Target	Control	Target
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

By making use of the column vector notation for a two-qubit wave function defined by eqn 13.9, we can write down a 4×4 unitary matrix to represent the C-NOT operation (see Exercise 13.6):

$$\hat{U}_{\text{CNOT}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{13.20}$$

The effect of the C-NOT operator on an arbitrary two qubit state can then be found as follows:

$$\hat{U}_{\mathrm{CNOT}} \cdot |\psi\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{pmatrix} = \begin{pmatrix} c_{00} \\ c_{01} \\ c_{11} \\ c_{10} \end{pmatrix}. \qquad (13.21)$$

It thus becomes apparent that the C-NOT operator has the effect of switching round the amplitude coefficients of the $|10\rangle$ and $|11\rangle$ states.

Example 13.2 What is the output of the quantum circuit shown in Figure 13.8 when both input qubits are in the $|0\rangle$ state?

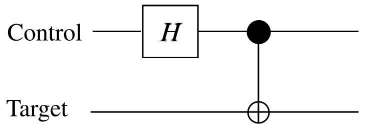

Fig. 13.8 Quantum circuit composed of an H gate and a C-NOT gate.

Solution
The circuit consists of an H gate and a C-NOT gate. We first compute the effect of the H gate on the control bit using eqn 13.18:

$$H \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}.$$

We then write the input to the C-NOT gate in the form given in eqn 13.5:

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle)\, |0\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle.$$

Finally we compute the output of the C-NOT gate using the C-NOT operator given in eqn 13.20:

$$|\psi'\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \cdot \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

The output is therefore:

$$|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

As we shall see in the next chapter, this output state is called a **Bell state** and is very important in the discussion of entangled states.

13.3.4 Practical implementations of qubit operations

Up to this point our treatment of quantum logic gates has been purely formal. We must now see how these operations can be implemented in the laboratory. As explained above, we can form an arbitrary quantum gate by combining C-NOT gates with single-qubit operations. Our task thus reduces to learning how to implement single-qubit operations and then the C-NOT operation. We begin by considering the single-qubit gates.

In physical terms, a single-qubit gate operates on an input qubit, and returns an output as a new qubit. As explained in Section 13.3.2, the operation of the gate can be given a geometric interpretation in terms of the Bloch vector representing the qubit. In general, it is possible to decompose an arbitrary single-qubit operator $\hat{U}$ into a series of Bloch vector rotations about the y- and z-axes together with multiplication by a phase shift:

See Nielsen and Chuang (2000, §4.2). By symmetry, the operator can equally well be decomposed into a series of rotations about the x- and y-axes.

$$\hat{U} = \mathrm{e}^{\mathrm{i}\alpha} R_z(\theta_3) R_y(\theta_2) R_z(\theta_1), \tag{13.22}$$

where α is a real number and $R_i(\theta)$ is the operator representing the rotation through an angle θ about Cartesian axis i. This result can be given an intuitive geometric interpretation in terms of an arbitrary mapping of the Bloch vector angles $(\theta, \varphi) \to (\theta', \varphi')$. (See Exercise 13.5.)

The absolute phase of a wave function is unobservable, and hence the global phase shift angle α in eqn 13.22 is not significant. The implementation of single-qubit operations thus reduces to carrying out Bloch vector rotations through arbitrary angles about the y- and z-axes. We consider here the specific case where the qubit is based on a two-level atom system. The Bloch vector rotations can then be performed by using the techniques of resonant light–atom interactions developed in Chapter 9.

As explained in Section 9.6, the application of a short electromagnetic pulse at the resonant frequency of the system causes a rotation of the Bloch vector about an axis in the x-y plane. (See Fig. 13.9.) The azimuthal angle φ of the rotation plane is set by the optical phase of the pulse, while the rotation angle Θ is equal to the **pulse area** defined by (cf. eqn 9.51):

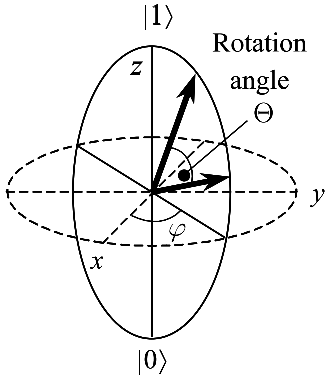

Fig. 13.9 The application of a short resonant electromagnetic pulse produces a rotation of the Bloch vector by an angle Θ about an axis within the x-y plane. The azimuthal angle φ of the rotation plane is determined by the phase of the pulse, while the rotation angle is governed by the pulse area given in eqn 13.23.

$$\Theta = \left| \frac{\mu_{01}}{\hbar} \int_{-\infty}^{+\infty} \mathcal{E}_0(t)\,\mathrm{d}t \right|, \tag{13.23}$$

where μ_{01} is the dipole moment for the $|0\rangle \to |1\rangle$ transition, and $\mathcal{E}_0(t)$ is the time-dependent electric field amplitude of the pulse. Pulses that produce rotation angles of 180° and 90° are called π- and $\pi/2$-pulses, respectively. These are especially important since they are part of the X and H operators. (See Fig. 13.6.)

The fact that the azimuthal angle of the rotation axis is determined by the phase of the pulse means that no explicit pulses are required for the z rotations. Keeping track of the z rotations is in fact effectively a book-keeping exercise. As an example, consider the result of two arbitrary operations $\hat{U}_1$ and $\hat{U}_2$. The combined operation is given from eqn 13.22 as:

$$\begin{aligned}\hat{U} &= \hat{U}_2 \cdot \hat{U}_1 \\ &= \mathrm{e}^{\mathrm{i}\alpha'} R_z(\theta'_3) R_y(\theta'_2) R_z(\theta'_1) \cdot \mathrm{e}^{\mathrm{i}\alpha} R_z(\theta_3) R_y(\theta_2) R_z(\theta_1) \\ &= \mathrm{e}^{\mathrm{i}(\alpha'+\alpha)} R_z(\theta'_3) R_y(\theta'_2) R_z(\theta'_1+\theta_3) R_y(\theta_2) R_z(\theta_1).\end{aligned} \tag{13.24}$$

Now a rotation about the z-axis of θ_1 followed by a rotation of θ_2 about the y-axis is equivalent to a single rotation by θ_2 about the axis in the x-y plane with an azimuthal angle of $(\pi/2 - \theta_1)$. The first two rotations can thus be performed by a single pulse with a phase of $(\pi/2 - \theta_1)$ and pulse area of θ_2. Similarly the next two rotations can be performed by a pulse of phase $(\pi/2 - \theta'_1 - \theta_3)$ and area θ'_2. The final z-axis rotation is simply recorded as a phase shift to be implemented when the next operation is performed. We can therefore perform rotations through arbitrary angles about arbitrary rotation axes by careful choice of the pulse phase and amplitude. This allows us to perform arbitrary single-qubit operations.

The implementation of single-qubit operations therefore presents no fundamental issues. We merely need to irradiate the atoms with short light pulses at the transition frequency with the correct pulse energy and phase to produce the required Bloch vector rotation. The essential physics for these operations has been known and understood for many years now. The key to the practical implementation of quantum computation thus becomes the demonstration of the C-NOT gate, which we now discuss.

Single-qubit operations on spin systems are similarly performed by applying resonant electromagnetic pulses of the required phase, amplitude, and duration. (See Appendix E.)

C-NOT gates act on two qubits according to the truth table given in Table 13.5. The key point is that we have to flip the target qubit depending on the state of the control qubit. The simplest way to see how this works is to consider the level scheme shown in Fig. 13.10. We have two qubits, q_A and q_B, each with their own resonant angular frequencies ω_A and ω_B. The two qubits interact with each other so that when we put both of them in the $|1\rangle$ state at the same time, the angular frequency of the system is not just equal to $(\omega_A + \omega_B)$, but is shifted to:

$$\omega_{AB} = \omega_A + \omega_B + \Delta \tag{13.25}$$

where $\hbar\Delta$ is the interaction energy. Δ can be either positive or negative depending on whether we have an attractive or repulsive interaction between the qubits. The interaction term has the effect that the resonant frequency of each qubit depends on the state of the other. If $q_A = |0\rangle$, we can perform the NOT operation on q_B by applying a π-pulse at angular frequency ω_B. However, if $q_A = |1\rangle$, the frequency of the π-pulse must be shifted to $\omega_B' = \omega_B + \Delta$. Similarly, q_A can be manipulated with pulses at ω_A if $q_B = |0\rangle$, but the frequency must be shifted to ω_A' when $q_B = |1\rangle$.

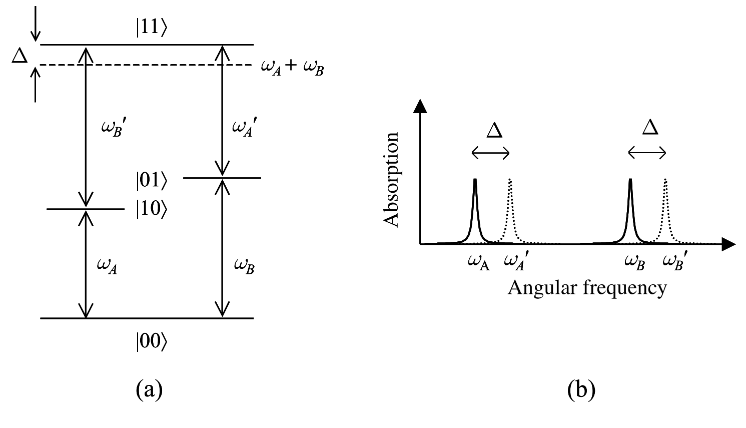

Fig. 13.10 (a) A possible level scheme for the implementation of the C-NOT gate using two qubits q_A and q_B with angular frequencies ω_A and ω_B, respectively. $\hbar\Delta$ is the interaction energy between the two qubits. (b) Absorption spectrum corresponding to the level scheme in part (a). The system only responds at angular frequency ω_B' $(\equiv \omega_B + \Delta)$ if $q_A = |1\rangle$. Similarly, the resonant frequency of q_A shifts to ω_A' if $q_B = |1\rangle$.

This is illustrated by the dotted lines in Fig. 13.10(b), which represent the response of the system when the other qubit is in the $|1\rangle$ state.

The argument works equally well the other way round.

Let us suppose that we have a system with the level scheme shown in Fig. 13.10 and we designate q_A as the control and q_B as the target. Starting from the ground state $|00\rangle$, we can demonstrate the four lines of the truth table of the C-NOT operation given in Table 13.5 as follows.

1. $|00\rangle \to |00\rangle$: this is trivially performed by doing nothing.
2. $|01\rangle \to |01\rangle$: we apply a π-pulse at ω_B to go from $|00\rangle \to |01\rangle$ and then follow it with a π-pulse at ω_B'. Nothing happens after the second pulse because the system only responds to frequency ω_B' if $q_A = |1\rangle$.
3. $|10\rangle \to |11\rangle$: we apply a π-pulse at ω_A to go from $|00\rangle \to |10\rangle$ and then follow it with a π-pulse at ω_B'. The system can now respond to the second π-pulse and goes into the $|11\rangle$ state.
4. $|11\rangle \to |10\rangle$: starting with the $|11\rangle$ state prepared in the previous line, we apply a second π-pulse at ω_B'. This causes q_B to flip from $|1\rangle$ to $|0\rangle$, leaving us with the $|10\rangle$ state, as required.

The key point of the demonstration is the response of the system to the π-pulse at ω_B'. This flips the target qubit if $q_A = |1\rangle$, but does nothing if $q_A = |0\rangle$.

It is, of course, apparent that we need to be able to measure the states of both qubits at the end of the operations in order to verify that the required output states have been produced. Hence the fifth item in the DiVincenzo check list. (See Section 13.6.)

Two qubit gates have been demonstrated experimentally in a number of physical systems, and we briefly list here some of the most important ones.

All-optical schemes One- and two-qubit gates can be implemented by encoding the quantum information onto the mode occupied by a single photon, and then manipulating the mode by means of linear optical components such as beam splitters. A single-photon source is required. (See Section 6.7.) This approach differs to the ones described below in that measurements form an intrinsic part of the computational process, instead of being just a method to read out the quantum states at the end of the calculation.

NMR systems The qubits correspond to spin states of specified nuclei within a molecule or crystal, and operations are carried out by RF pulses. The nuclei are in different environments and so have slightly different resonance frequencies. The spins on nearby nuclei interact with each other through the spin–spin interaction. (See Exercise 13.13.)

Ion traps The qubits correspond to the excitation states of a row of single ions held in an ion trap. The ions are all identical and therefore have the same resonance frequency, but can be addressed individually by laser pulses because they are physically separated from each other. The ions interact through the repulsive forces associated with vibrational displacements from the equilibrium positions.

Cavity QED systems The qubits correspond to opposite circular polarization states of two photons interacting with a single atom inside a resonant cavity. The photons interact with each other through their mutual interaction with the atom, which is strongly enhanced by the resonant cavity.

Quantum dots The qubit consists of an exciton confined in a quantum dot. (See Appendix D, especially Section D.3.) The excitons behave like two-level atoms, and the operations are performed by resonant optical pulses. Different types of excitons within an individual dot interact through their Coulomb interaction.

Superconducting systems The quantum information is stored as the charge of a small region of superconducting material called a 'box'. The box is connected to a charge reservoir through a Josephson tunnelling junction. The charge is controlled by the voltage across the junction, and the $|0\rangle$ and $|1\rangle$ states correspond to charges differing by one Cooper pair, with $\Delta q = -2e$. Adjacent boxes are electrostatically coupled via their mutual Coulomb repulsion, and gate operations are performed by sequences of voltage pulses.

Further details of some of these prototype quantum computing systems will be given in Section 13.6. For the other systems, the reader is referred to the bibliography.

13.4 Decoherence and error correction

The operation of a quantum computer relies on the precise manipulation of quantized states of individual quantum systems. We require that the qubits should interact with each other in a controlled way and with nothing else. Unfortunately, this idealistic scenario is impossible to achieve in practice. All quantum systems are fragile because they couple with their environments to a greater or lesser extent. A totally isolated system would in fact be useless for quantum information processing because we would have no means to interact with it and perform the quantum operations that are at the heart of quantum computation.

The 'environment' that we are considering here consists of a very large number of atoms and molecules which obey quantum laws individually, but classical laws collectively. The thermal motion of the particles within the environment acts like a random noise source which can interact with the qubits and introduce uncontrollable random behaviour. For example, the thermal noise could cause a qubit to flip its logical value randomly, and thereby lose its quantum information irretrievably. Since the operation of a quantum computer relies on manipulating *coherent* superposition states, the fragility of qubits with respect to environmental noise is conveniently quantified in terms of **decoherence** rates.

Superposition states are called 'coherent' because they manifest quantum interference effects analogous to those that can occur between coherent light waves. The coupling of simple quantum systems with the noisy environment is now understood to explain why quantum effects such as the Schrödinger cat paradox are not observed in the macroscopic world. An experiment demonstrating that the coherence of a quantum superposition state is controlled by its coupling to a noisy environment is described in C. J. Myatt, *et al.*, *Nature* **403**, 269 (2000).

We discussed the various types of process that cause decoherence when we considered the damping of coherent superposition states in Section 9.5.2. The key parameter that quantifies the decoherence is the **dephasing time** T_2. In gases the dephasing time is often limited by collisions between the particles, while in solids or liquids we have to

Table 13.6 Decoherence times (T_2) and gate operation times (T_{op}) for some of the physical systems considered for quantum computing. N_{op} is the number of gate operations that could be performed before decoherence occurs. It should be emphasized that many of the values quoted in this table represent optimistic upper limits, and only those labelled with an (e) are based on genuine experimental data. Thus, for example, it is known that quantum dot excitons have dephasing times as long as ~ 1 ns, and that T_{op} can be as short as ~ 1 ps, but, as yet, no-one has managed to demonstrate 10^3 gate operations.

System	T_2 (s)	T_{op} (s)	N_{op}	References
Nuclear spin	10^4	10^{-3}	10^7	DiVincenzo, *Phys. Rev A* **50**, 1015 (1995)
Ion trap	10^0	10^{-6} (e)	10^6	Schmidt-Kaler *et al.*, *J. Phys. B* **36**, 623 (2003)
				Steane *et al.*, *Phys. Rev. A* **62**, 042305 (2000)
Exciton (quantum dot)	10^{-9} (e)	10^{-12} (e)	10^3	Langbein *et al.*, *Phys. Rev. B* **70**, 033301 (2004)
				Li *et al.*, *Science* **301**, 809 (2003)
Electron spin (quantum dot)	10^{-7}	10^{-12}	10^5	Pazy *et al.*, *Europhys. Lett.* **62**, 175 (2003)
Superconducting flux qubit	10^{-8} (e)	10^{-10} (e)	10^2 (e)	Chiorescu, I. *et al.*, *Science* **299**, 1869 (2003)

contend with the randomness introduced by interactions with thermally excited vibrations (i.e. phonons).

The number of quantum operations that can be performed before dephasing sets in is given by:

$$N_{\mathrm{op}} = \frac{T_2}{T_{\mathrm{op}}}, \tag{13.26}$$

where T_{op} is the time required to perform the operation. Some optimistic values of N_{op} are given in Table 13.6. It is apparent that a certain amount of trade-off takes place. For example, NMR systems have very long dephasing times because nuclear spins only interact very weakly with their environment. At the same time, it also difficult to interact with nuclear spins in a controlled way, and hence the quantum operations tend to be rather slow. Less well-isolated systems decohere faster, but they are easier to interact with and the operations can be performed faster. Thus while it is obvious that we must work as hard as we can to reduce the dephasing rate for any particular system, it does not automatically follow that the systems with the longest dephasing times offer the best possibilities.

The original proposals for quantum error correcting may be found in P. W. Shor, *Phys. Rev. A* **52**, R2493 (1995), and A. M. Steane, *Phys. Rev. Lett.* **77**, 793 (1996). An experimental demonstration of quantum error correction using ion traps is described in J. Chiaverini *et al.*, *Nature* **432**, 602 (2004).

Fortunately, the situation is not quite as bad as it might seem at first. In classical data processing, error-checking protocols are used all the time to check and correct for errors. In an analogous way, it is possible to correct for the effects of dephasing on qubits by **quantum error correction** algorithms. The basic principle is essentially the same as for classical error correction, although the details are obviously very different. The idea to use extra qubits to check the fidelity of the data, and then apply quantum algorithms to reconstruct the original states. In this way we can achieve **fault-tolerant quantum computation**: that is, robust quantum computation in the presence of a finite amount of noise from the environment. The relative error rate required is less than about 10^{-5}. The price that is paid is that the processing speed is reduced, since some of the quantum resources are being employed purely for error correction.

A quick glance at Table 13.6 suggests that the ratio of T_2 to T_{op} is quite promising for some systems. However, it should be stressed that many of the values quoted in Table 13.6 are only theoretical limits. For example, the theoretical limit of T_2 for the 729 nm transition of a Ca^+ ion is set by the ~ 1 s radiative lifetime of the upper level, but the coherence time measured experimentally is only ~ 1 ms. Much further work is clearly needed to identify new physical systems and understand the fundamental limits that determine the coherence and gate operation times.

See F. Schmidt-Kaler *et al.*, *J. Phys. B: At. Mol. Opt. Phys.* **36**, 623 (2003).

13.5 Applications of quantum computers

Let us suppose that we had a large quantum computer. What would it be useful for? The general answer to this question has yet to be given. We cannot say for certain whether a quantum computer will always be more powerful than its classical counterpart. On the other hand, there is a growing number of situations where we do know that the quantum computer is more efficient than the classical one, at least in principle.

Before looking at specific examples, it is worth recalling that the fundamental reason why a quantum computer can outperform a classical one is related to the inherent parallelism of quantum systems. (See Section 13.2.1.) A quantum register of size N can hold 2^N numbers simultaneously, whereas a classical register of the same size only contains one number. When we operate on the quantum register, we perform the calculation on many numbers simultaneously, whereas the classical computer only calculates the answer for one given number. Therefore, the quantum computer will eventually beat the classical one provided that we exploit the parallelism effectively.

In the subsections that follow, we shall first illustrate how the benefits of the quantum computer are harnessed in practice for two important quantum algorithms, namely the Deutsch algorithm and the Grover algorithm. We shall then briefly consider a few of the other applications that have been proposed in the literature for quantum computers.

Table 13.7 Possible results for the four possible versions of the one-bit function $f(x)$. The function is described as constant if both outputs are the same, or balanced if the results of 0 and 1 occur with the same frequency.

	$f(0)$	$f(1)$	
f_1	0	0	Constant
f_2	1	0	Balanced
f_3	0	1	Balanced
f_4	1	1	Constant

13.5.1 Deutsch's algorithm

The first algorithm to be proposed that demonstrated that a quantum computer can be more efficient than a classical one is the **Deutsch algorithm**. The algorithm concerns the evaluation of a binary function $f(x)$ that acts on a one-bit binary number. The function has only two possible results: $f(0)$ and $f(1)$, and is defined to be *balanced* or *constant* according to the scheme given in Table 13.7. The task to be performed is to determine whether an unknown function is balanced or constant. A classical computer requires two calls of the function to complete this task, but a quantum computer can do it with just one, as we shall now demonstrate.

The Deutsch algorithm applies specifically to the case of a one-bit function. A more generalized version for an N-bit function is called the **Deutsch–Josza algorithm**. The Deutsch algorithm described here is a slightly improved version of the original one given in Deutsch's paper on the universal quantum computer (*Proc. R. Soc. London A*, **400**, 97 (1985)). See Nielson and Chuang (2000) for further details of the historical development of the algorithm.